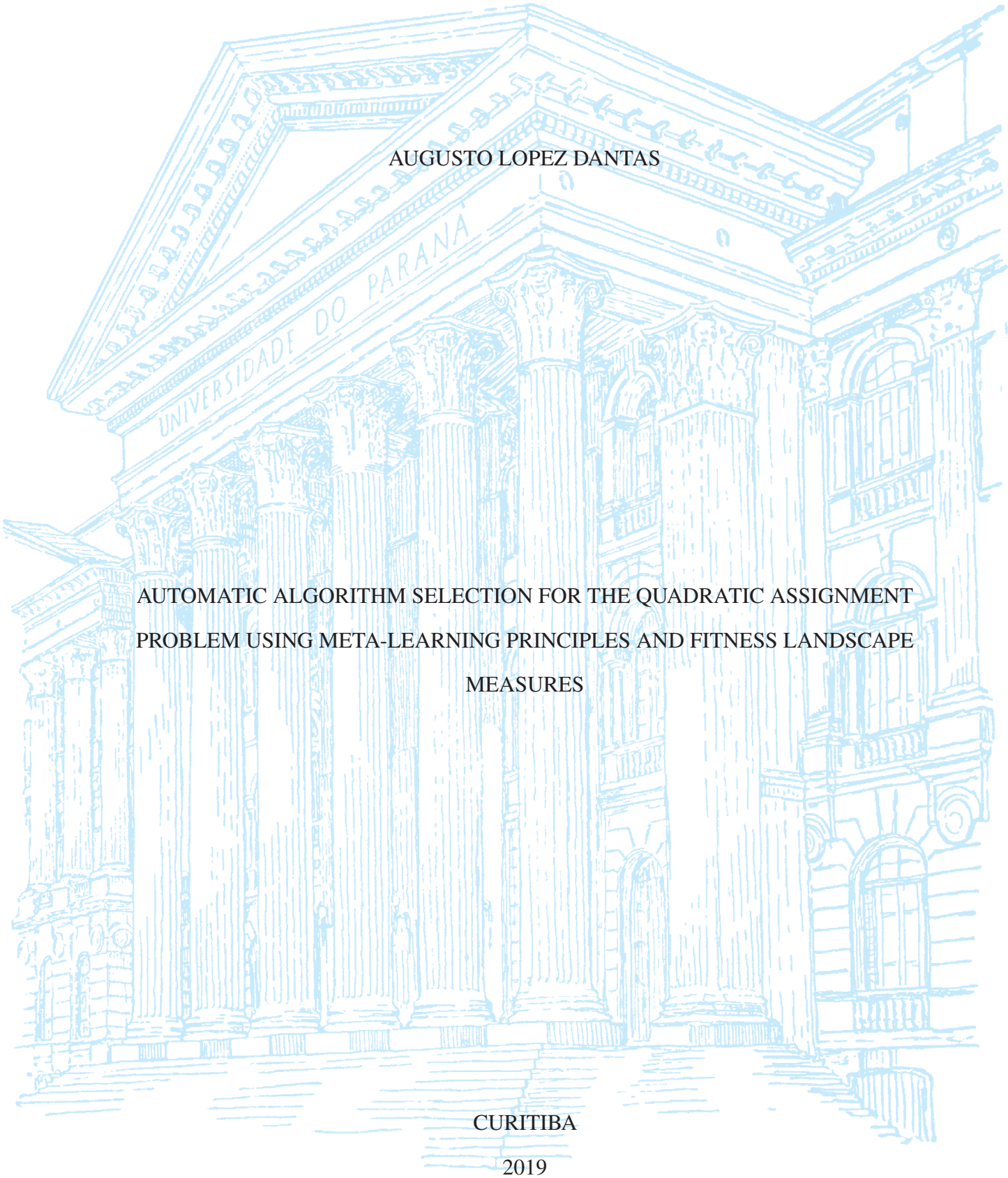
UNIVERSIDADE FEDERAL DO PARANÁ

AUGUSTO LOPEZ DANTAS



AUTOMATIC ALGORITHM SELECTION FOR THE QUADRATIC ASSIGNMENT PROBLEM USING META-LEARNING PRINCIPLES AND FITNESS LANDSCAPE MEASURES

CURITIBA

2019

AUGUSTO LOPEZ DANTAS

AUTOMATIC ALGORITHM SELECTION FOR THE QUADRATIC ASSIGNMENT PROBLEM USING META-LEARNING PRINCIPLES AND FITNESS LANDSCAPE MEASURES

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Orientador: Aurora Trinidad Ramirez Pozo.

CURITIBA

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

D192a Dantas, Augusto Lopez
Automatic algorithm selection for the quadratic assignment problem using meta-learning principles and fitness landscape measures [recurso eletrônico] / Augusto Lopez Dantas. – Curitiba, 2019.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2019.

Orientador: Aurora Trinidad Ramirez Pozo .

1. Algoritmos. 2. Otimização Combinatória. 3. Programação heurística. I. Universidade Federal do Paraná. II. Pozo, Aurora Trinidad Ramirez. III. Título.

CDD: 511.8

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **AUGUSTO LOPEZ DANTAS** intitulada: **Automatic Algorithm Selection for the Quadratic Assignment Problem Using Meta-learning Principles and Fitness Landscape Measures**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua Aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 13 de Fevereiro de 2019.

AURORA TRINIDAD RAMIREZ POZO
Presidente da Banca Examinadora (UFPR)

ROBERTO SANTANA HERMIDA
Avaliador Externo (UPV)

LUIZ EDUARDO SOARES DE OLIVEIRA
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Acknowledgements

I thank my parents, my brother, my advisor, my friends, the university, all my professors so far and the National Council for Scientific and Technological Development (CNPq).

RESUMO

Algoritmos meta-heurísticos são usados para obter boas soluções em tempo factível para vários problems de busca NP-difícil. Entretanto, o desempenho dos algoritmos depende fortemente das características do problema, o que significa que não há um único método capaz de obter as melhores soluções em todos os cenários. Portanto, estudar formas de realizar seleção automática de algoritmo para problemas de otimização é um crescente tópico de pesquisa nos últimos anos. Esse tipo de tarefa pode ser abordado através de conceitos de meta-aprendizagem, oriundos da comunidade de aprendizagem de máquina, cujo objetivo é mapear as características das instâncias do problema ao desempenho de um conjunto de algoritmos. Um aspecto fundamental de meta-aprendizagem é a caracterização das instâncias, o que é feito pelas meta-características e que, por sua vez, devem ser propriedades informativas que afetam o desempenho dos algoritmos. Algumas pesquisas propõem o uso de medidas baseadas em Análise de Superfície de Aptidão (ASA) para representar as instâncias. Esse tipo de métrica caracteriza as formas das superfícies geradas por um operador heurístico, e são tradicionalmente usadas para relacionar a dificuldade que um determinado problema oferece à uma meta-heurística. Neste trabalho, foi conduzida de forma progressiva uma série de estudos relacionados à seleção de algoritmos com meta-aprendizagem e medidas de ASA para o Problema Quadrático de Alocação (PQA). O PQA é um clássico problema de busca NP-difícil e é conhecido por ser um dos mais desafiadores para algoritmos de otimização. As principais contribuições deste trabalho são os experimentos de meta-aprendizagem realizados usando medidas de ASA conhecidas na literatura e alguns dos algoritmos meta-heurísticos do estado-da-arte. Diversos aspectos da abordagem são evidenciadas ao longo dos estudos, como desempenho de classificação, esforço de aplicação e consumo de tempo.

Palavras-chave: Seleção de Algoritmo. Meta-aprendizagem. Otimização Combinatória. Análise de Superfície de Aptidão.

ABSTRACT

Meta-heuristic algorithms have been used to obtain good solutions in feasible time for many NP-hard search problems. However, the performance of the algorithms heavily depends on the features of the problem, meaning that it does not exist a single method able to achieve the best solutions in all scenarios. Therefore, studying ways to perform automatic algorithm selection for optimization problems is an increasing research topic in the past recent years. This type of task can be addressed by using meta-learning concepts, from the machine learning community, which aims at mapping the characteristics of problem instances to the performance of a set of algorithms. A key aspect of meta-learning is the characterization of the instances, which is done by the meta-features that, in turn, must be informative properties that affect the performance of the algorithms. Some researches have proposed the use of features based on Fitness Landscape Analysis (FLA) to characterize the instances. This type of measures characterizes the landscape shapes generated by a heuristic operator, and are traditionally used to relate the difficulty in which a problem imposes to a meta-heuristic. In this work, a series of studies were progressively conducted regarding algorithm selection with meta-learning and FLA features for the Quadratic Assignment Problem (QAP). The QAP is a classical NP-hard search problem that is known for being one of the most challenging for optimization algorithms. The main contributions of this work are the meta-learning experiments performed using well known sets of FLA measures from the literature and some of the state-of-the-art meta-heuristic algorithms. Several aspects of the approach are highlighted along the studies, such as the performance of classification, application effort and time consumption.

Keywords: Algorithm Selection. Meta-learning. Combinatorial Optimization. Fitness Landscape Analysis.

List of Figures

List of Tables

List of Acronyms

ACO	Ant Colony Optimization
ASP	Algorithm Selection Problem
BI	Best Improvement
BLS	Breakout Local Search
BMA	Breakout Memetic Algorithm
FDC	Fitness Distance Correlation
FLA	Fitness Landscape Analysis
ILS	Iterated Local Search
KNN	K-Nearest Neighbors
MH	Metropolis-Hastings
ML	Meta-learning
MMAS	Max-Min Ant System
MMASBI	Max-Min Ant System with Best Improvement
PCA	Principal Component Analysis
QAP	Quadratic Assignment Problem
RF	Random Forest
RO-TS	Robust Taboo Search
UX	Uniform Crossover

List of Symbols

n	instance size
S_n	set of all possible permutations
ϕ	permutation
μ	mean of the values of a matrix
σ	standard deviation of the values of a matrix
δ	movement cost

CONTENTS

1 Introduction

Optimization problems consist on the search for minimizing or maximizing one or more functions through the attribution of a set of variable, which could also be subjected to some constraints (Talbi, 2009). The problem can be classified into two major categories according the nature of the variables, they being the continuous and discrete types. Combinatorial optimization problem are discrete problems with a finite search space and have a great practical importance, making them the object of study of several works (Blum e Roli, 2003).

The Quadratic Assignment Problem (QAP) is one of the most challenging combinatorial optimization problems and was first proposed as a mathematical model in 1957 (Koopmans e Beckmann, 1957). It consists on assigning a set of facilities into a set of locations in a way to obtain the minimal possible flow between the facilities and the minimal distance between the locations. The QAP is a widely studied problem, not just because its difficulty, but also because several real world problems can be modeled as QAP, such as the typewriter keyboard design, the location of the hospital departments, the backboard wiring, among others (Burkard et al., 1998a).

Because the QAP is an NP-hard problem (Sahni e Gonzalez, 1976), the use of exact methods is unfeasible for instances with large sizes. Therefore, practitioners have used meta-heuristics approaches to obtain good solutions in a reasonable computational time. Meta-heuristics are generic methodologies that act as guiding strategies for heuristic search operators (Talbi, 2009). Several meta-heuristics have been proposed, each one with distinct properties, such the ones based on single-solution perturbation, like the Tabu Search (Glover, 1989), the population evolutionary approaches, like the Genetic Algorithm (Holland, 1975), and constructive methods, such the Ant Colony Optimization (Dorigo e Caro, 1999).

However, although the proposed meta-heuristics have achieved satisfying results, they have different biases. Hence, there is not a single algorithm able to outperform the others in all cases, according to the No Free Lunch Theorem (Wolpert e Macready, 1997), meaning that some algorithms are preferable to others on specific problems (Schumacher et al., 2001; Langdon e Poli, 2007; Auger e Teytaud, 2010).

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. But, selecting an appropriate algorithm for a given problem is a difficult task (Tang et al., 2014) that requires expert knowledge on search algorithms (Blum et al., 2011). In the machine learning community, the task of automated algorithm selection has been tackled by meta-learning. Meta-learning aims at understanding the relationship between the problem characteristics and the performances of the solving algorithms (Smith-Miles, 2008), thus allowing 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 (Smith-Miles, 2008) such as regression, time-series forecasting, sorting, constraint satisfaction, optimization and recommender systems (Cunha et al., 2018).

The success of a meta-learning approach relies mostly on the quality of the characteristics used to represent the instances, namely the meta-features. Some of them can be properties of

the instance definition itself, such as matrix and graph properties. Additionally, it has been demonstrated that measures based on Fitness Landscape Analysis (FLA) can be used for this goal (Pitzer et al., 2013). FLA is a technique that aims at gathering more knowledge about the internal structure of a problem. This is done by analysing the shape of the landscape formed by a set of sampled solutions, given their costs and neighborhoods (Pitzer et al., 2011).

This work is compound of several studies that were conducted on automatic algorithm selection with meta-label and FLA features for the Quadratic Assignment Problem. These studies were performed in a progressive way, meaning that each next in order improves some aspect of the previous. Wherefore, this research culminated on the production of three scientific papers, and the experiments and results of each of them are presented here in full in their respective chapter.

The first paper, titled A Meta-Learning Algorithm Selection Approach for the Quadratic Assignment Problem, is presented in Chapter 5. The meta-learning process was performed using all 135 instances from the QAPLIB benchmark, with 14 meta-features, being 7 based on matrices and 7 based on FLA, and considering three state-of-the-art meta-heuristics. A Random Forest was used as the prediction model.

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. A brief discussion about the role of the meta-features on prediction is also presented. Albeit the promising results, the additional work that was required for achieving good classification performance is undesirable.

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. This is done in the next paper named Selecting Algorithms for the Quadratic Assignment Problem with a Multi-label Meta-learning Approach (Chapter 6). Two techniques for dealing with multi-label datasets are evaluated and, in addition to having high prediction accuracies without much effort, the results point out that the meta-learning approach is statistically better than running only the most occurrent meta-heuristic on all problems.

At last, Chapter 7 shows the third paper with the title Low Cost Fitness Landscape Features for Meta-Learning Algorithm Selection: a Study on the Quadratic Assignment Problem. The main aspect of this study was to evaluate an additional set of recently proposed FLA measures, by also highlighting the required computational time for their extraction. Moreover, 17 new and hard instances were added to the dataset, and the worst algorithm from the previous studies was replaced by a novel evolutionary algorithm in the algorithm pool.

The results demonstrated that it is possible to obtain good classification performances by using metrics extracted without the need of an expensive sampling methodology, which is a recurring problem of meta-learning using FLA. Nevertheless, even when removing the problem specific meta-features, these FLA measures are representative enough to discriminate the instances with a satisfactory accuracy.

2 Related Works

One of the first research on Algorithm Selection Problem (ASP) was proposed by Rice (Rice, 1976) which focus on selecting an algorithm from a portfolio that is likely to perform best based on measurable features of the problems instances. The understanding of problem instances and algorithm performance is a task that can be tackle with meta-learning approach (ML). The ML has been used effectively in algorithm portfolio to predict the algorithm that likely performs best for unseen problems (Smith-Miles, 2008).

A pioneer work regarding automated algorithm selection for QAP has been done in Smith-Miles (2008), 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 Beham et al. (2017) 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 Pitzer et al. (2013), 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 Smith-Miles (2008), only 28 instances were analysed, whereas in Beham et al. (2017) the size of the dataset was 47. Moreover, the mentioned works did not evaluate the individual classes performances. In Pitzer et al. (2013), 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, such as our first work on this matter. Instead, the algorithm selection problem can be tackled as a Multi-label classification task, since there are instances in which more than one algorithm achieve the same solution costs. In Kanda et al. (2011), 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.

3 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

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} \tag{3.1}$$

where S_n represents all possible permutations of a set $N = \{1, 2, \ldots n\}$, 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 a flow and a distance matrices whose values are used to compute the solution cost. Figure 3.1 exemplifies an instance of size 5 and a possible solution configuration. Considering that each element on the permutation represents a facility, the positions indicate their 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 (3.1), would be equal to 2140.

Figure 3.1: 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 the experiments, it was used all 135 instances [1] from the standard benchmark library QAPLIB (Burkard et al., 1998b). Additionally, the third work also includes 17 new instances drawn from Drezner et al. (2005), which were designed to be hard for heuristic search.

[1] The instance `esc16f` was not considered due to its flow matrix having only zero values.

The QAP is a widely studied problem because several real world problems can be modeled as QAP, such as the typewriter keyboard design, the location of the hospital departments, backboard wiring, among others (Burkard et al., 1998a).

The QAP is a NP-hard optimization problem and is considered as one of the hardest optimization problems since the largest instances that can be solved today with exact algorithms are limited to instances of size around 30 (Anstreicher et al., 2002) (instances with sizes of 64 and 128 were solved in Fischetti et al. (2012), but benefiting on their extremely symmetric structures). So, several meta-heuristic algorithms have been proposed to solve QAP, which include algorithms like Simulated Annealing (Connolly, 1990), Tabu Search (Battiti e Tecchiolli, 1994), Genetic Algorithm (Drezner, 2003), GRASP (Pardalos e Resende, 1994), Ant Algorithms (Maniezzo e Colorni, 1999), and Scatter Search (Cung et al., 1997). Previous research results show that the characteristics of the QAP instances strongly influence the relative performance of the various algorithms (Drezner et al., 2005) and that motivate the meta-learning approach presented here.

4 Meta-learning

Since the seminal work of Rice (1976) Rice (1976), the field of meta-learning (ML), especially with respect to algorithm selection and configuration, has been an active area of research. According to Brazdil et al. (2009), 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. Although ML has been traditionally applied to classification and data mining problems, its concept has been extended to other application domains (Smith-Miles, 2008) such as regression, time-series forecasting, sorting, constraint satisfaction, optimization and recommender systems (Cunha et al., 2018).

In optimization, ML can be defined as mapping the characteristics of problem instances to the performance of a set of algorithms allowing the selection of the most promising algorithm.

Therefore, a main aspect for ML is the characterization of the problem instances or meta-knowledge of the problem. One important component of the meta-knowledge are the meta-features, which in this case 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.

In ML approach, the algorithm selection problem is addressed like a traditional learning task. A meta-model is induced, which can be described like the meta-knowledge 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. The approach includes a sequence of activities, as presented in Figure 4.1.

- Meta-knowledge extraction: Given a base of past problems with information about the application of the different algorithms, this task extracts knowledge for the meta-learning task: meta-features and meta-label.
- Meta-learning: This task corresponds to the application of an ML technique to the Meta-knowledge base generated in the step before.
- Meta-model: The model obtained in the last step is used to select among the algorithms, the most suitable considering the characteristics (meta-features) of the current problem.

4.0.1 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).

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

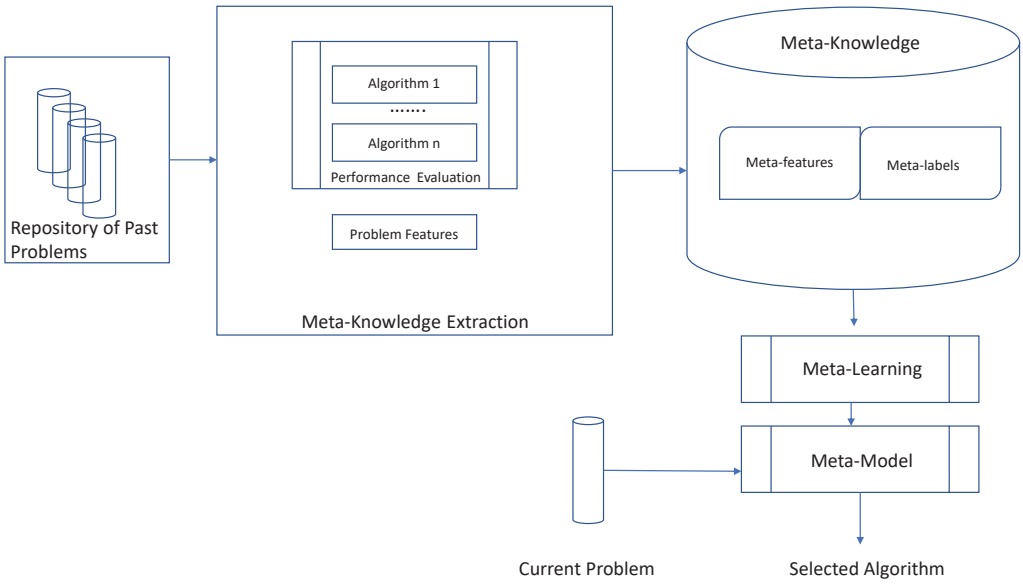


Figure 4.1: Meta-learning sequence of activities

the required computing time. For QAP, there are seven simple characteristics that may represent the instances, which are described next.

- 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 (Stützle, 2006).

- 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 $\left(\frac{n^2-n}{2}\right)$.

Fitness Landscape Analysis

FLA aims at gathering more knowledge about the internal structure of a problem. This is done by analysing the shape of the landscape formed by a set of sampled solutions, given their costs and neighborhoods. Therefore, the 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 4.2, and, for an instance of size n, produces $\frac{n(n-1)}{2}$ neighbors. This is also the operator used during the local search phase in the meta-heuristics described later.

There are two major FLA 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, usually by sampling local optima solutions (Pitzer et al., 2011). The following meta-features are of the global FLA type.

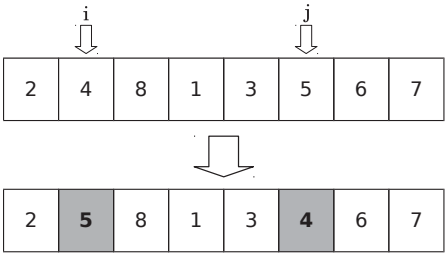


Figure 4.2: The swap operator

- Fitness Distance Correlation (FDC): the FDC measures the distance correlation between the distances from the closest global optimum and the solution cost (Jones e Forrest, 1995). The distance in this context is the amount of different elements in the same positions on both solutions, also called as the Hamming Distance. Let c_i and d_i be respectively the cost and distance of the i_{th} solution from the set of local minima and m be the number of local optima, the FDC is calculated by

$$FDC = \frac{\frac{1}{m}\sum_{i=1}^{m}(c_i - \mu_c)(d_i - \mu_d)}{\sigma_c.\sigma_d} \tag{4.1}$$

where μ_c and μ_d are the average cost and distance and σ_c and σ_d are the cost and distance standard deviation. Since knowing the set of global optimum solutions is unpractical, we refer instead to a set of pseudo-global optima, which are the unique best solutions found during the sampling.

- Number of pseudo global optima: is the size of the set of the best unique 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) (Lu et al., 2011). 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 (Lunacek e Whitley, 2006). 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. It is also divided by the instance size.

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

The first study, described in Chapter 5, also includes 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 (Pitzer et al., 2013). Although this type of FLA is more computational expensive, the authors from Chicano et al. (2012) proposed an expression to calculate 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.

4.0.2 Meta-labels

The meta-labels represent the most suitable algorithms for a problem instance. So, the considered algorithms must be executed on all instances and those with the best performances are chosen as the meta-labels for the dataset entry.

There are different ways to define the performance of an algorithm. For example, the authors from Pitzer et al. (2013) created a history of solution costs taken at determined periods during the runtime of the algorithm. The performance then was determined by the domination of this history, i.e., the algorithms that had a higher amount of better costs considering all snapshots. The paper from Chapter 5 defines the performance as the average solution cost in 30 runs of the algorithms and, in case of ties, the execution time. Meanwhile, the next two works (Chapters 6 and 7) deal with multi-label settings and run all algorithms with the same CPU time, hence the algorithm with the best performance is that with the least average 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. The first two studies employ the algorithms Breakout Local Search (BLS) (Benlic e Hao, 2013), the Max-Min Ant System with Best Improvement Local Search (MMASBI) (Stützle e Hoos, 2000) and the Robust Taboo Search (RO-TS) (Taillard, 1991). In the last work, due to the relatively poor performance of the RO-TS, it is replaced by the novel Breakout Memetic Algorithm (BMA) (Benlic e Hao, 2015).

They all have in common the appliance of a local search as an intensification mechanism. More precisely, they employ a Best Improvement (BI) local search, in which the whole neighborhood (or a truncated one) is first scanned and then it moves to the best solution among them (Burkard et al., 1998a).

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 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 (Taillard, 1995)

$$\begin{aligned}\delta(\phi,i,j) = {} & d_{ii} * (f_{\phi(j)\phi(j)} - f_{\phi(i)\phi(i)}) + d_{ij} * (f_{\phi(j)\phi(i)} - f_{\phi(i)\phi(j)}) + \\ & d_{ji} * (f_{\phi(i)\phi(j)} - f_{\phi(j)\phi(i)}) + d_{jj} * (f_{\phi(i)\phi(i)} - f_{\phi(j)\phi(j)}) + \\ & \sum_{k=1,k\neq i,j}^{n} (d_{ki} * (f_{\phi(k)\phi(j)} - f_{\phi(k)\phi(i)}) + d_{kj} * (f_{\phi(k)\phi(i)} - f_{\phi(k)\phi(j)}) + \\ & d_{ik} * (f_{\phi(j)\phi(k)} - f_{\phi(i)\phi(k)}) + d_{jk} * (f_{\phi(i)\phi(k)} - f_{\phi(j)\phi(k)}))\end{aligned} \tag{4.2}$$

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 (Taillard, 1995)

$$\begin{aligned}\delta(\phi', u, v) = \delta(\phi, i, j) * (d_{ru} - d_{rv} + d_{sv} - d_{su}) * (f_{\phi(j)\phi(u)} - f_{\phi(j)\phi(v)} + \\ f_{\phi(i)\phi(v)} - f_{\phi(i)\phi(u)}) * \\ (d_{ur} - d_{vr} + d_{vs} - d_{us}) * (f_{\phi(u)\phi(j)} - f_{\phi(v)\phi(j)} + \\ f_{\phi(v)\phi(i)} - f_{\phi(u)\phi(i)})\end{aligned} \tag{4.3}$$

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 taboo 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 (Benlic e Hao, 2013).

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) (Dorigo e Caro, 1999). 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 (Stützle e Hoos, 2000). 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.

Robust Taboo Search

The Taboo 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 taboo movement results in a solution better than the best found so far (Glover, 1989). The robust version introduces a randomly variable duration for considering each movement as taboo, 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 (Taillard, 1991).

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 (Benlic e Hao, 2015). 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 (Benlic e Hao, 2015).

5 A Meta-Learning Algorithm Selection Approach for the Quadratic Assignment Problem

This was the first conducted study, and was mainly focused on discovering the maximum potential of classification by performing extensive feature selection steps. Also, because in the ASP all classes have the same importance in terms of misclassification, the classification performance also takes into account the individual classes predictions, instead of only the overall accuracy like some previous works have done (Smith-Miles, 2008; Pitzer et al., 2013).

This approach lead us to propose a cascade classification scheme, which divides the problem and allows us to train more specialized models. For this, we conducted a series of experiments that were useful at gaining insight of what steps to take in order to improve the individual performances, such as data cleansing and dimensionality reduction. At the end, a reliable automatic algorithm selection system for the QAP was achieved.

5.1 Activities Description

Apart from the 7 static features described in Chapter 4.0.1, this work extracts the same five meta-features used in Smith-Miles (2008) and with the sampling method proposed by Stützle e Hoos (2000), in which the authors used them to study the viability of their new algorithm. The extraction process is based on experimental runs of simple search algorithms and further analysis of the gathered local optima solutions. Additionally, two meta-features, namely the autocorrelation coefficient and the correlation length, are also included because their values for all QAPLIB instances are given in Chicano et al. (2012).

First, an Iterated Local Search (ILS) is executed 1000 times, with 500 iterations each, and all achieved unique local optima solutions are stored. The best solutions (with the same cost) within this set are called the pseudo global optima, which are required to compute the features, since it is unfeasible to know the actual global optima set. Next, a Best Improvement Local Search is randomly repeated until 5000 unique local optima is found, or until it exceeds a certain amount of execution time (which was set to 5 minutes for both the ILS and BI runs).

Table 5.1 summarizes the FLA meta-features that were used. The average distance and the FDC are computed for both the samples gathered by the ILS and the BI runs. For more detailed explanations, see Chapter 4.0.1.

Table 5.1: Extracted FLA meta-features

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 (Jones e Forrest, 1995)	$FDC = \frac{\frac{1}{m}\sum_{i=1}^{m}(c_i-\mu_c)(d_i-\mu_d)}{\sigma_c.\sigma_d}$
bi_fdc		
acc	Autocorrelation coefficient (Chicano et al., 2012)	Average fitness variation caused by a single step along the neighborhood
acl	Correlation length (Chicano et al., 2012)	Number of required steps to make the correlation values statistically different

As meta-labels candidates, the following algorithms were considered:

- Breakout Local Search (BLS)
- Max-Min Ant System with Best Improvement Local Search (MMASBI)
- Robust Taboo Search (RO-TS)

The three meta-heuristics were run 30 times for every instance, all with the same set of 30 initial seeds. A great obstacle when comparing algorithms performances is related to the definition of a stopping criterion. The most natural approach is limiting their execution with the same amount of CPU time. But even by doing so, it does not make it necessarily fair, specially when the algorithms were implemented by different people with different paradigms, as is our case.

Also, since our goal was to study the ASP as a single label problem, we decided to use the execution time just as a mean of untying. So, the stopping criterion for each algorithm was set based on numbers of iterations, it was done according to preliminary tests and considering the algorithms behaviors. With that said, the BLS and MMASBI were limited to $100 * n$ iterations, where each iteration contains full local search appliances. Although the MMASBI operates with 5 solutions at each iteration (as it is the default configuration), the quality of the solutions resulted by the construction process implicates that less movements are performed during local search, compared to the random or perturbed initial positions in the BLS. As for the RO-TS, the stopping criterion was set to $2000 * n$ iterations, because in this implementation each iteration is equivalent to one single movement. With these configuration we could achieve a relatively balanced dataset after performing some data cleansing, as is discussed later.

Two other reported good algorithms, the Hybrid Genetic Algorithm (Misevicius, 2004) and the Fast Ant System (Taillard e Gambardella, 1997), were also initially considered, but, with the implementations that we had at hand and with their default configurations, they could not achieve competitive results.

The executions were performed on an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz and with a time limit of 10 minutes for run, that was only reached for the larger instances. 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. As mentioned later, most of ties only occurred on small and easy instances. Table 5.2 shows the resulting classes distribution.

Perhaps, in a real case scenario, it would be more interesting to label the data based on statistical equivalences of the performances. This would result in a multi-label classification problem, which brings different caveats and was not the objetive of this study. Nevertheless, this is mentioned as a possible future work.

Table 5.2: Full dataset

Class	Instances
BLS	33
MMASBI	89
RO-TS	13

As classification model, we used the Random Forest implementation from the library `scikit-learn` [1]. The parameters settings were 100 estimators and a tree depth limit of 10. The Random Forest was chosen because it can inherently work with multi-class problems and does not require data normalization. Also, some other models were used in preliminary tests, but, without any parameter tunning, they showed overall worse predicting performances.

For splitting the dataset, a stratified KFold Cross Validation with 10 groups strategy was adopted, thus, the performance measurements were obtained by micro-averaging the groups results. Both the classifier and the folding methods received the same initial seed 0 as parameter, in order to keep the tests equivalent and reproducible. All other parameters were left the same as the default.

Before arriving at our proposed cascade scheme, a series of experiments were performed which guided us in the task of data preprocessing. Initially, we trained the Random Forest for the complete original dataset and with all features, in which, due to the class imbalance, the minority classes had considerably worse performances. Therefore, some data cleansing was necessary to try to improve the results. For this, we inspected the optimization performances 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 said 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 (Smialowski et al., 2009). 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, also discussing some remarks about the behavior of the resulting features, given at Section 5.3.

Finally, when observing that even with the improvements there was still a large difference in the individual classes performances, we conclude that it would be better to project our multi-class problem into sequential binary problems, resulting in the proposed cascading scheme. The experiment materials, such as the source codes of the meta-heuristics and the classification datasets, can be found online [2].

5.2 Experiments Results

In this section, we show the details of the experiments that lead us to propose the cascade classification scheme, along with its configuration and results.

[1] `http://scikit-learn.org/stable/index.html`
[2] `https://github.com/aldantas/AS_QAP_CEC2018`

5.2.1 Data Cleansing

The results for the classification task using all 14 features are shown in Table 5.3. It can be noted that, although it achieved a considerably high accuracy, it was mainly due to the performance on the majority class (MMASBI).

Table 5.3: Evaluation metrics for the classification on the full dataset using all features

Accuracy	F-score			
	BLS	MMASBI	RO-TS	Average
0.8741	0.8254	0.9180	0.6667	0.8034

In fact, by looking at the normalized confusion matrix illustrated in Figure 5.1, it is possible to observe that all misclassifications on class BLS (0) and most of the misclassifications on class RO-TS (2) were related to the class MMASBI (1).

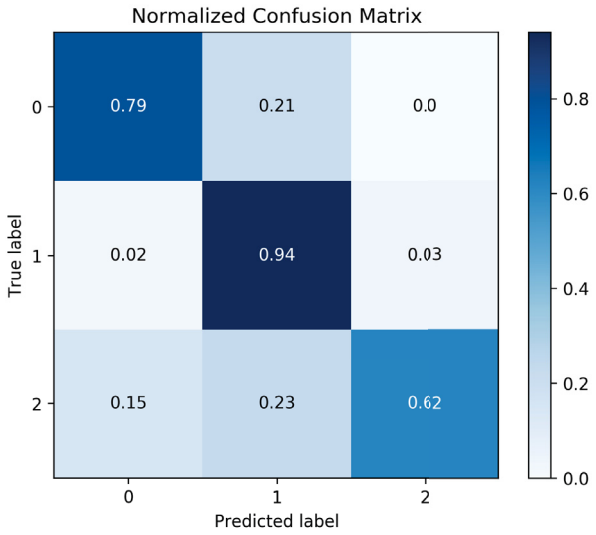


Figure 5.1: Normalized confusion matrix for classification on the full dataset using all features

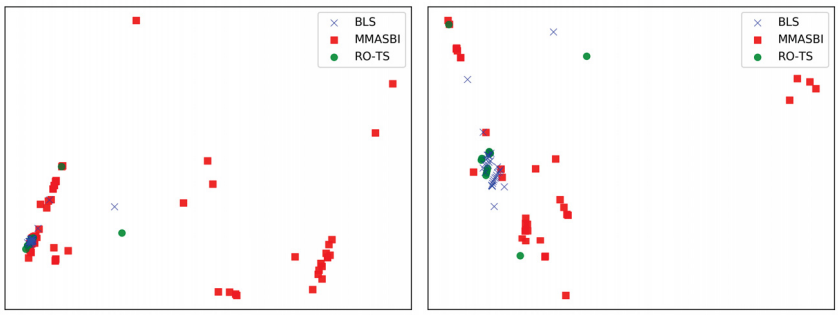
When inspecting the results achieved by the meta-heuristics, we observed that, for several instances (most with sizes smaller than 20), the three algorithms achieved the same average cost over the 30 runs. Thus, it is reasonable to conclude that these easily solvable instances contain features that may not correctly indicate the performances of the algorithms.

Therefore, we created a reduced dataset by eliminating said instances, resulting on the removal of 54 instances, all previously belonging to class MMASBI. This happened because of the different chosen stopping criteria for the algorithms, which gave the MMASBI a small advantage regarding the execution time. Since these time diferences are mostly in matter of seconds, all algorithms could be considered as corrected classified for those instances when applying the model as an algorithm selection system. Table 5.4 summarizes this new dataset.

Table 5.4: Reduced dataset

Class	Instances
BLS	33
MMASBI	35
RO-TS	13

Figure 5.2 shows the bidimensional feature spaces given by applying Principal Component Analysis (PCA) (Wold et al., 1987), both to the full dataset (5.2a) and to the reduced dataset (5.2b). It is clear that, by removing the easy instances, the instances labeled as BLS and RO-TS became more distinguishable from the instances labeled as MMASBI.



(a) Full Dataset (b) Reduced Dataset

Figure 5.2: Bidimensional feature spaces resulted by PCA

However, when training the classifier with this reduced dataset, the prediction performances for each class were actually deteriorated, as shown in Table 5.5. Because this result contradicts what the PCA has indicated, it is likely that a proper feature selection is required to improve the performance on each dataset.

Table 5.5: Evaluation metrics for the classification on the reduced dataset using all features

Accuracy	F-score			
	BLS	MMASBI	RO-TS	Average
0.8148	0.8182	0.8493	0.6087	0.7587

5.2.2 Dimensionality Reduction

Since our feature set has a small size of 14, it is feasible to evaluate all possible combinations, which results in a total of 16383 subsets. The best performing feature set is then determined by picking the one with the highest accuracy, highest average F-score and the least amount of features, in this respective order. Table 5.6 shows the best features found for the both the datasets.

Table 5.6: Best features for each dataset

Dataset	Best Features
Full	n, fd, dd, fsp, fas, ils_fdc, bi_fdc, acc
Reduced	fd, dd, dsp, fas, ils_dst, ils_fdc, bi_dst, bi_fdc

The results for the classification using the aforementioned features are given in Table 5.7. We can see that, on both cases, the use of a proper set of features enhanced the overall performances. Also, by reducing the dataset, we noticed a small decay of accuracy and average F-score comparing to the full dataset, but we could improve the performances on the minority classes.

But even with all those improvements, the performance on class RO-TS is still considerably below the performances on the other classes. By looking at the Figure 5.2b, we can

Table 5.7: Evaluation metrics for the classification on the full and reduced datasets using the selected best features

Dataset	Accuracy	F-score			
		BLS	MMASBI	RO-TS	Average
Full	0.9037	0.8667	0.9355	0.75	0.8507
Reduced	0.8765	0.8923	0.8947	0.7619	0.8496

notice that the instances which the best algorithm is the RO-TS are close to the ones labeled with the BLS algorithm. This proximity is even more highlighted when observing the boxplot distributions of each feature displayed in Figure 5.3. It is clear that both classes have instances with similar features, specially those based on fitness landscapes.

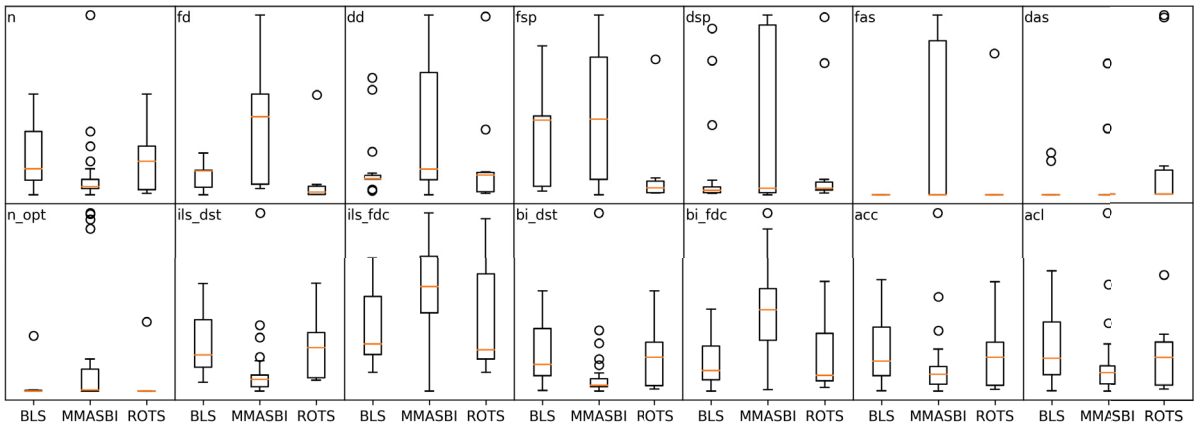


Figure 5.3: Features distributions for each class

These similarities are actually not surprising, since both the RO-TS and BLS algorithms share same common properties such as being single solution based, being iterative improvement strategies and applying the same neighborhood structures. Hence the RO-TS outperformed the other algorithms on fewer instances, it ends up getting more disadvantaged from these similarities. Thus, in an attempt to improve the individual predictions on the minority classes, we propose a cascade classification scheme.

5.2.3 Cascade Classification 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 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.

As before, 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 5.4 illustrates our proposed scheme, along with the best meta-features for each model.

In order to fairly compare it to the previous classification made on the reduced multi-class dataset (Table 5.4), we used the same folding groups as before. The prediction performances of the described cascade classification can be seen in Table 5.8. When comparing them to the multi-class results, displayed in Table 5.7, we can observe that we managed to maintain the exact same accuracy, but considerably improving the performance on class RO-TS, which, in turn, improves the classes average F-score.

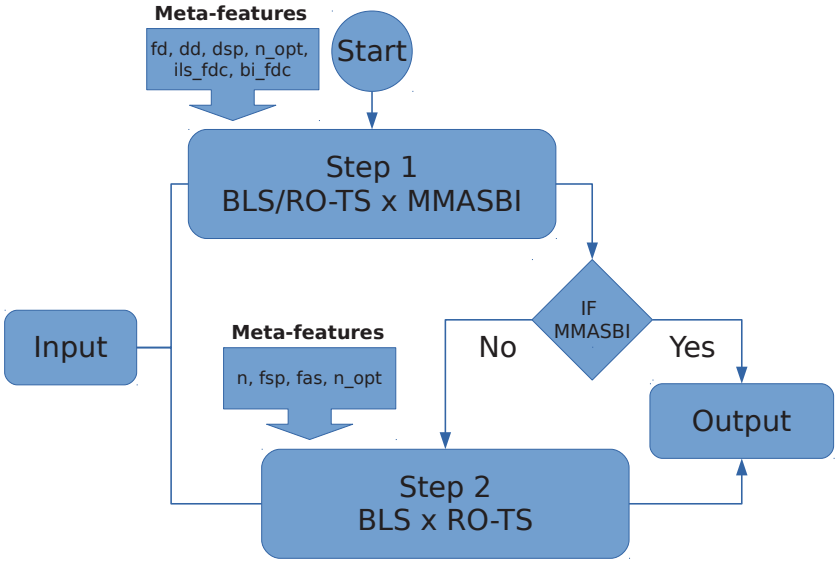


Figure 5.4: Proposed cascade classification scheme

Table 5.8: 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

These results indicate that our scheme has the ability to balance the individual classes prediction power, making the selection system more reliable.

5.3 Meta-features Analysis

In this section we investigate the resulting meta-features subsets that were able to better discriminate the instances in the cascade scheme. At the top level of our system, the optimal subset was: {fd, dd, dsp, n_opt, ils_fdc, bi_fdc}. It is noticeable that the dominance (concentration of association weights) and the FDC (problem difficulty) have an important role in distinguishing the local search based algorithms from the MMASBI, which is a bio-inspired constructive algorithm.

To better understand the influence of the meta-features, in Figure 5.5 we show their importances returned by the `scikit-learn` implementation when training the model with the complete feature set. We can observe that three features from the optimal subset (fd, dd, bi_fdc) presented high individual performances, but others, such as the n_opt, would likely not be chosen if this metric was used to perform feature selection. Another remark is that, the second most important meta-feature, the ils_dst, is not present in the optimal set, probably because, when removing the noisy features, it becomes redundant, which indicates that thresholding based feature selection is not a robust approach, since it does not consider the correlational influences.

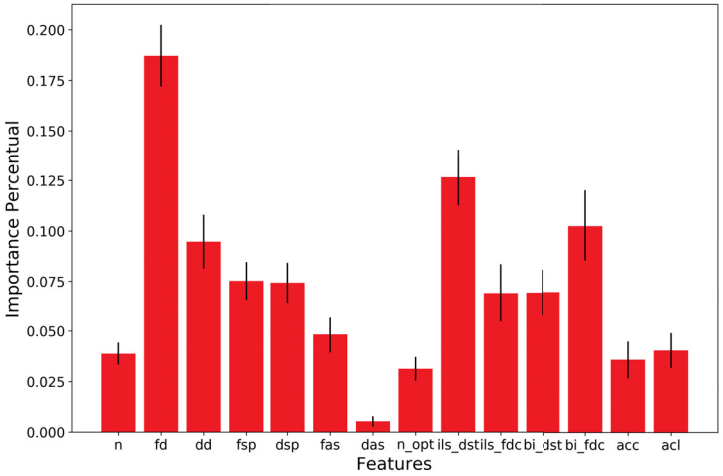


Figure 5.5: Feature importances for the step 1 in the cascade scheme

Going down the cascade scheme, the optimal subset is now: {n, fsp, fas, n_opt}. Indeed, it is a smaller set than the previous one, which was expected since they are the best features for a model trained with fewer data instances. Here, we can see that the majority of the optimal meta-features is static, meaning that the BLS and RO-TS algorithms suffer more influence from the problem instances structures, and not so much from their landscapes. Again, we show the feature importances when training with all of them in Figure 5.6. Although the most valuable meta-feature (fsp) was in fact selected to be in the optimal subset, curiously, the others that remained are actually those with least individual importances.

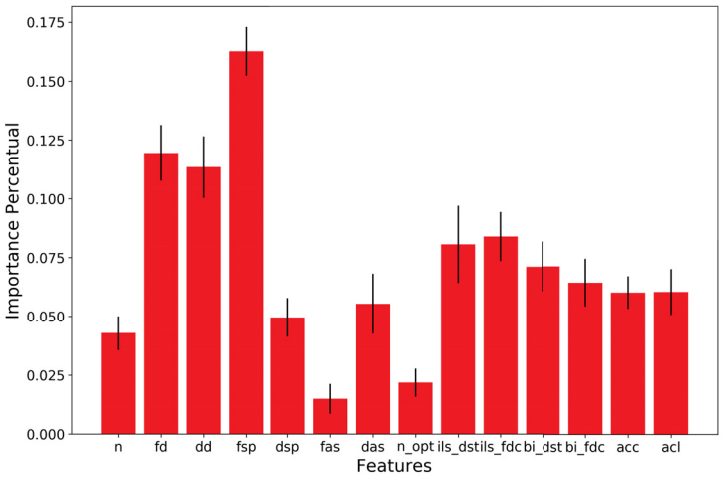


Figure 5.6: Feature importances for the step 2 in the cascade scheme

For visualizing how these meta-features may influence the classification task, we trained a Decision Tree using all instances from the dataset (without splitting for testing) and its graphical representation is shown in Figure 5.7. As we can see, the combination of the meta-features n, n_opt and fas (the least valuable ones) was able to isolate 29 out of 33 samples of class BLS. This highlights the difficulty in performing feature selection, since there may be unknown relationships between them.

Another observation that we can point out is that neither acc and acl, which belong to the local method type of FLA features, appeared in any of the cascade optimal subsets. A possible reason for this is that they were overshadowed by the local search based features, which

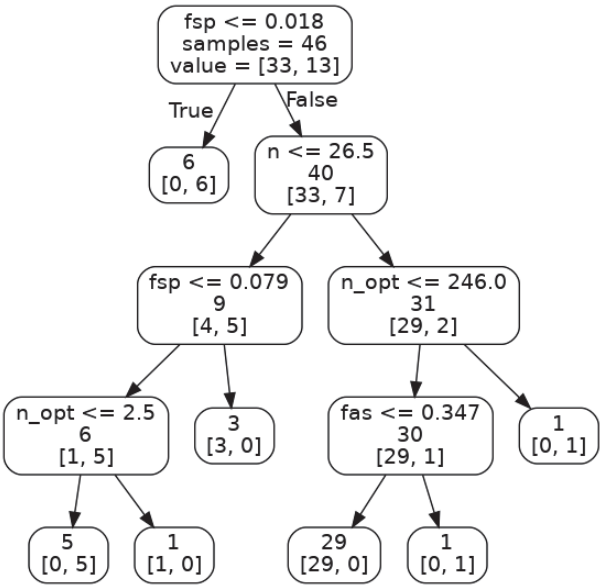


Figure 5.7: Decision tree visualization for the BLS x RO-TS dataset using the optimal meta-features

also employ a form of landscape sampling. In fact, in both dataset, as shown in Figures 5.5 and 5.6, they demonstrated subpar relevancies.

Lastly, the fact that both levels ended up with different meta-features (with the exception of one) indicates the beneficial traits of a cascade classifying system.

A natural concern regarding algorithm selection based on meta-learning is the computational time required for extracting the instances features. It is clear that, for practical appliances, the extraction process should not exceed the time for executing the whole set of algorithms, otherwise, the selection is unjustified.

In this case, the bottleneck of our approach is the sampling for the FLA features. Hence we employed two local search methods, each limited by 300 seconds, the execution of the three meta-heuristics should ideally take more combined time than 600 seconds. In Figure 5.8 is shown the average sum of the three algorithms execution time by their size. We can see that the extraction process begins to be worth for instances of size around 100 and beyond. This could be changed if the sampling had a smaller time limit or if there were more optimization algorithms being selected. Nevertheless, performing a selection for only new large instances, which are the most challenging ones, is a valid case scenario.

5.4 Discussion

This paper presented a study on algorithm selection for the Quadratic Assignment Problem under a meta-learning approach. For this, we have generated classification datasets considering a meaningful number of QAP instances, 3 meta-heuristic optimization algorithms and 14 meta-features. These meta-features were extracted based on available information contained in the problem instance structures, on properties of solution samples collected by simple local search algorithms and others traditional FLA metrics.

After analysing the solution space representations and the meta-features distributions, we have applied some steps in order to improve the classifier performance. For data cleansing, we removed a few redundant instances and then we performed exhaustive searches of optimal features for the sake of dimensionality reduction.

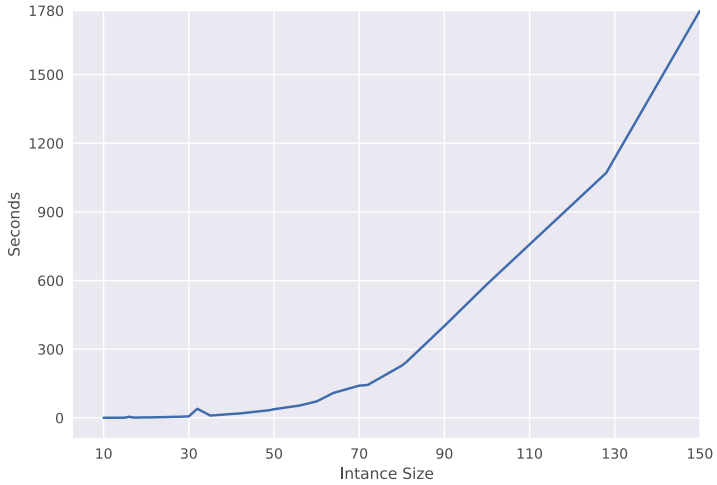


Figure 5.8: Time spent for the execution of the three meta-heuristics by size

Since our goal was to evaluate not only the overall accuracy, but also the individual classes performances, we came to the proposal of a classification performed on a cascade scheme, breaking the multi-class problem into two binary output problems. Thus, at the first level, a model is trained to distinguish instances labeled with the MMASBI algorithm from the ones labels with either the BLS or RO-TS. Next, for the entries that are not classified as the MMASBI class, they are inputted on a second model specialized for selecting between the BLS and RO-TS classes.

With this model, we were able to achieve satisfactory results, improving considerably the performances on the minority classes with a small decay on the majority class, which we considered as a fair trade-off for an algorithm selection system. Also, we have demonstrated the importance and difficulty of selecting a proper set of meta-features, highlighting the advantages of training different specialized models.

Possible future works are the inclusion of more instances, specially larger ones, using some crafted generator and the extraction of additional meta-features, considering that this could imply the necessity of studying a more intelligent feature selection approach. Also, it would be interesting to add different optimization algorithms for labeling, like an evolutionary one, provided that its performance is competitive with the others already in use. Another interesting direction is to tackle the task as a multi-label classification problem, in which the algorithms are ranked based on statistical equivalences.

6 Selecting Algorithms for the Quadratic Assignment Problem with a Multi-label Meta-learning Approach

Although the previous paper reported great classification performances, they were achieved only after several manual steps, which is not desirable for an automated algorithm selection application. Therefore, in this next study, we did not include feature selection and we did not break down the learning task as before.

Moreover, previously the ranking of the meta-heuristics was done by setting pre-determined number of iterations and using the computation time as an untying measure. Now, we investigated the algorithm selection task as a multi-label classification problem, in which the data entries can be assigned to many classes at the same time (Madjarov et al., 2012). Since more than one meta-heuristic can have the same performance for a given instance, this approach is adequate to perform algorithm selection (Kanda et al., 2011).

For dealing with multi-label learning, there are mainly two approaches: the problem transformation, that manipulates the dataset, and algorithm adaptation, in which the model is adapted to deal with the multiple labels (Zhang e Zhou, 2014). We investigated both approaches by using a Random Forest implementation that can inherently work with multi-label classification. Then, they were compared in terms of evaluation metrics of predictions, namely the accuracies and the individual label F-scores.

Lastly, we showed the effectiveness of the meta-learning approach by comparing the solution costs returned by the selected algorithms instead of running only the algorithm with the best performance among them.

6.1 Experiments

For this study, we used the same set of meta-features extracted in the previous work (see Chapter 5), with the exception of the autocorrelation coefficient and the correlation length which, as reported previously, did not contribute for the classification very much. As for the meta-labels, the same three meta-heuristics are considered, namely the Breakout Local Search, The Max-Min Ant System with Best Improvement and the Robust Taboo Search.

Two approaches for dealing with a multi-label classification were compared. 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. The other approach is to let the learning algorithm deal with the multi-label configuration by its own, which in the literature it is called algorithm adaptation (Zhang e Zhou, 2014). To make the comparison fair, we used a model implementation that can handle both cases. First, we present the classifications performances, measured by accuracies and F-scores. Next, we analyse how good are the solutions retrieved by the selected algorithms

in relation to the best known solutions, also comparing the improvement of employing algorithm selection against applying only a single algorithm over the instances.

For generating the multi-label dataset, the three described meta-heuristics were run 30 times for every instance, always with the same set of 30 distinct seeds. These executions were performed on an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz, with each 10 tests being run in parallel with independent processes. Then, each data entry, i.e., the instance meta-features, was labeled to all algorithms that achieved the same average solution cost over the 30 runs.

In a scenario like this, it is desirable that all algorithms should be executed on the same amount of computational time. Therefore, to determine the execution time limit for each instance, we used the experiments from Chapter 5, in which the algorithms were executed for a predetermined number of iterations, along with a time limit of 10 minutes. The MMASBI and BLS were limited to $100 * n$ iterations, where each iteration contains full local search appliances. Although the MMASBI operates with 5 solutions at each iteration (as it is the default configuration), the good quality of the solutions given by the construction process implicates that less movements are performed during local search, when comparing to the random or perturbed initial positions in the BLS. As for the RO-TS, the stopping criterion was set to $2000 * n$ iterations, because in this implementation each iteration is equivalent to one single movement.

Then, using the aforementioned executions, we collected the average time that the algorithms spent to reach their best found solutions. This data was then used to fit a polynomial function that dictates the required computational time given the instance size. The data points and the fitted function $t(n)$ are illustrated in Figure 6.1. Because of the 10 minute limit, the algorithms could only achieve good solutions for the instances with sizes up to 100, thus, the remaining instances data points were not used in the fitting process.

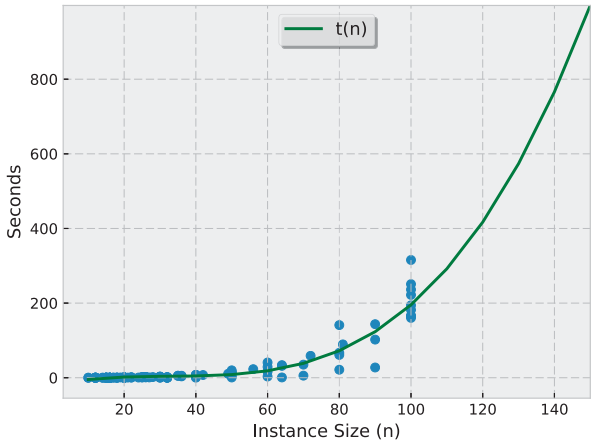


Figure 6.1: Fitted function of required running time by size

Therefore, the meta-heuristics were executed for a particular instance according to the time limit given by (6.1). As it will be pointed out later, with this constraint the algorithms could achieve overall high quality solutions.

$$\begin{cases} \lceil t(n) \rceil & \text{if } t(n) > 1 \\ 1 & \text{otherwise} \end{cases} \tag{6.1}$$

The resulting classes distribution of the generated learning dataset is shown in Figure 6.2. Because of the quality of the chosen algorithms, and because there are several easy instances in QAPLIB, a reasonable amount of data entries received all labels. Nevertheless, it is still noticeable that each algorithm could outperform another, or all others, in many instances.

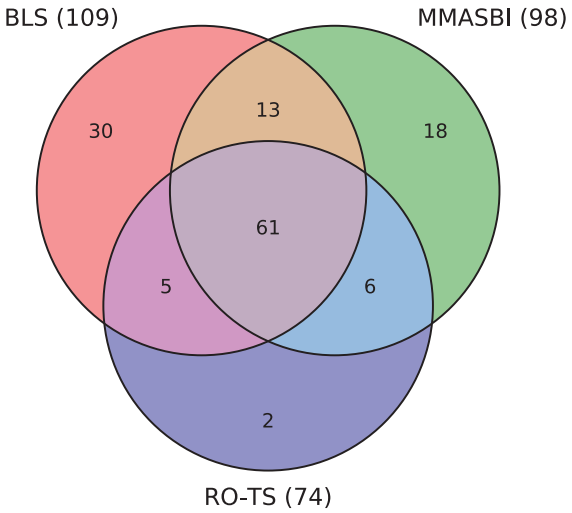


Figure 6.2: Multi-label dataset (BLS/MMASBI/RO-TS)

As classification model, it was used the Random Forest implementation from the `scikit-learn` library, with the parameters set to 100 estimators and a tree depth limit of 10. In order to keep the tests equivalent and reproducible, the seed for the random state parameter was fixed to 0. The Random Forest was chosen because, in this implementation, it can inherently work with multi-class and multi-label problems, also dispensing data normalization. Besides, some other models, such as the Multi Layer Perceptron and the Support Vector Machine on a Grid Search, were used in preliminary tests, but, without any parameter tunning, they showed overall worst predicting performances.

For validating the model, the Leave-one-out Cross Validation strategy was adopted. Thus, the performance measurements were obtained by micro-averaging the results in each train/test folding. The evaluation metrics used to assess the classification performances were the accuracy and the F-scores of each individual class. However, for multi-label classification, there are different types of accuracies that can be evaluated. 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 (Madjarov et al., 2012).

Additionally, a third accuracy was used, which 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. The experiment materials are available at `https://bit.ly/2KtOlEL`.

6.2 Classifications Results

Initially it was validated the classification using the label powerset transformation method, which means that the problem was turned 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 6.1.

Table 6.1: 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

However, to evaluate the predictions and to further compare them to the direct multi-label classification, we converted the powerset labels back to their original multi-label settings. Nevertheless, the traditional single label accuracy is still measured by the subset accuracy.

Next, we performed the same cross validation over the classification using the algorithm adaptation method, i.e., the learning algorithm is prepared to directly deal with a multi-label dataset by itself. The accuracies of both approaches are shown in Table 6.2.

Table 6.2: Evaluated accuracies on the complete dataset

Approach	Instance	Subset	Recommendation
Powerset Labels	0.881	0.793	0.837
Multi-labels	0.893	0.815	0.852

As it can be observed, both approaches achieved good performances, although the algorithm adaptation presented higher accuracies, showing that the model could probably identify some classes relationships, which is not possible with the label powerset method. Moreover, even when analysing the individual classes performances (BLS, MMASBI and RO-TS), as shown in Table 6.3, both approaches presented similarly high F-scores.

Table 6.3: Labels F-scores

Approach	BLS	MMASBI	RO-TS	Average
Powerset Labels	0.920	0.960	0.904	0.928
Multi-labels	0.916	0.980	0.910	0.935

However, we suspected that those high evaluations were influenced by the 61 easy instances that were labeled to all three meta-heuristics. In fact, by looking at the individual subsets F-scores, displayed in Table 6.4, we can notice that said instances (subset BLS/MMASBI/RO-TS) are indeed showing high performances on both approaches. But, interestingly, the subsets only BLS and only MMASBI also achieved predicting measurements above random (0.5), specially the former. Therefore, these two algorithms were able to outperform the others for a reasonable amount of instances and with distinct properties. Meanwhile, the subset only RO-TS suffered from the lack of samples, being unable to be correctly predicted.

Thus, we calculated the accuracy metrics considering only the instances not belonging to subset BLS/MMASBI/RO-TS (without the 61 instances with all labels), which are shown in Table 6.5. As it can be observed, the accuracies were considerably decreased, confirming our belief.

Table 6.4: Subsets F-scores

Subset	Powerset Labels	Multi-labels
only BLS	0.903	0.933
only MMASBI	0.722	0.689
only RO-TS	0.000	0.000
BLS/MMASBI	0.538	0.533
BLS/RO-TS	0.500	0.800
MMASBI/RO-TS	0.000	0.000
BLS/MMASBI/RO-TS	0.891	0.928

Table 6.5: 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

Nevertheless, the recommendation accuracy was still considerably above random. Besides, this decay of performance was mainly due to the prediction of the minority class (RO-TS), as displayed in Table 6.6.

Table 6.6: 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

Therefore, our meta-learning based algorithm selection approach still demonstrated to be able to provide good prediction results. This is likely to be even further improved if the RO-TS would be assigned more often when including more QAP instances outside from the benchmark.

6.3 Output Performances

This section discusses the comparison of the solution costs returned by the selected algorithms with the output of each individual algorithm for all 135 instances. The obtained solutions were evaluated in relation to the best known solutions of the respective instance.

Table 6.7 shows the amount of instances that the algorithms themselves, and both the selection approaches, could achieve the best solution at least once among the 30 executions. We can observe that the BLS presented the best results among the three meta-heuristics, which was expected since it was the most present class in the dataset. For the classifier outputs, in cases that more than one class was predicted, the algorithm was selected at random among them. By doing so, the algorithms selected by the multi-label approach were able to find the best-known solutions more often than if using only the BLS.

Although it was a low improvement, this was the case because the QAPLIB contains only a small set of instances, and most of them may not be representative enough for current research tendencies. Perhaps the advantages of algorithm selection would be even more highlighted if larger and harder instances (Drezner et al., 2005) were included.

Since the generation of the learning dataset was based on average costs, we also evaluated the frequencies that each algorithm and classification strategy was able to reach the best solutions

Table 6.7: Best known solutions achieved at least once

BLS	MMASBI	RO-TS	Powerset Labels	Multi-labels
122	116	107	121	124

in all 30 runs. Table 6.8 displays those frequencies, showing that the MMASBI presented more stable results in this case. Because of the randomness for choosing the algorithm when more than one is predicted, the outputs of the classifiers may vary between worst and best cases.

Table 6.8: Best known solutions achieved in all runs

BLS	MMASBI	RO-TS
79	91	76

Powerset Labels		Multi-labels	
Worst Case	Best Case	Worst Case	Best Case
80	91	82	95

The problem adaption approach, i.e., using the multi-label dataset, achieved best solutions in average more often than the MMASBI at the best cases. Meanwhile, the selections made by the problem transformation approach could only achieve the same performance as MMASBI and only at the best case.

Considering the worst cases of both approaches, we evaluated the solution costs of the remaining instances, that is, the instances that the selected algorithms did not achieve the best known cost in average. Table 6.9 displays how many times the classifiers were statistically better, worse and equivalent, in comparison with the solutions given by MMASBI for the same instances. This is done by applying the Kruskal-Wallis H test on each pair of 30 results sets, considering a threshold of 0.05 for the *p-value*.

Table 6.9: Statistical comparison of the selected algorithms results against the MMASBI standalone results

Approach	Better	Worse	Equivalent
Powerset Labels	18	12	25
Multi-labels	28	7	18

Therefore, even in the worst case, the classifiers presented overall better performances. The advantage of using algorithm selection was even more noticeable with the classifications made on the original multi-label dataset, showing that this approach is better for the given task. Nevertheless, although the recommendation accuracy was around 85%, the misclassifications are less costly in a multi-label setting because it still has a fair possibility to fallback to the next best algorithm, thus making the system more reliable.

6.4 Discussion

This paper investigated a meta-learning approach for algorithm selection applied to the Quadratic Assignment Problem. A multi-label classification dataset was created considering: all instances from the QAPLIB, 3 meta-heuristic algorithms and 12 instance meta-features. Two approaches for dealing with multi-label learning problems were evaluated. One is the label powerset method, in which the original dataset is turned into a single label multi-class dataset by considering each

subset of labels as a different class. In the other approach, the classification is directly made over the multi-labeled instances. A implementation of Random Forest that is able to deal with both cases was used as classification model.

Regarding the predictions, both methods achieved similarly good performances, which were measured by three different types of accuracies and by the individual label F-scores. However, besides presenting slightly higher evaluation metrics, the direct multi-label strategy allowed to select the algorithms that resulted in overall better solutions than the execution of the individual algorithms, even in the worst case regarding misclassifications. Meanwhile, the label powerset method could not outperform the results of the best algorithm. Therefore, the experiments point out that the proposed multi-label algorithm selection is a reliable approach.

Future works should use larger and harder QAP instances. In addition, it would be interesting to raise the dimensionality of the classification problem by extracting more meta-features and considering other competitive meta-heuristics.

7 Low Cost Fitness Landscape Features for Meta-Learning Algorithm Selection: a Study on the Quadratic Assignment Problem

In the past experiments, a few weaknesses in the approach were detected, which are the target of this study First, one of the meta-heuristics, the Robust Tabu Search (RO-TS) (Taillard, 1991) had a considerably worse performance compared to the other algorithms, making the meta-learning dataset unbalanced. So, here we replaced the RO-TS by the recently proposed Breakout Memetic Algorithm (Benlic e Hao, 2015). By doing so, we also improved the diversity of the nature of our algorithm pool by having one perturbative, one constructive and one evolutionary.

Moreover, we also included additional instances drawn from Drezner et al. (2005), which were designed to be hard for heuristic search. Specifically, they are 12 instances of the type `drexx` with sizes ranging from 15 to 132, and 5 instances of the type `taixxeyy` whose size is 343, which is very large for the QAP.

Finally, and mainly, this work performs the meta-learning with a new set of FLA based meta-features that were found in the literature. Besides that, the main drawback of FLA features for algorithm selection is the high time consumption for the extraction (Pitzer et al., 2013). Therefore, these features are extracted with a cheaper sampling methodology, namely the Metropolis-Hasting algorithm, as described in Vanneschi et al. (2004).

7.1 QAP Meta-learning Activities

This Section gives details of the Meta-learning activities that we conduct in this study. Initially, the meta-heuristic are executed on all instances in order to rank their performances and generate the meta-labels. Then, the meta-features must be extracted to compose the meta-knowledge base. Here, we create three different meta-knowledge base composed by three different sets of meta-features. One of them is the same set generated in Chapter 6. The other two include the additional FLA meta-features that are extracted with the Metropolis-Hastings sampling, but considering different sample sizes. At last, the evaluation results of the classification are described.

7.1.1 Meta-heuristics ranking

For generating the labels, the three described meta-heuristics were run 30 times for every instance, always with the same set of 30 seeds. These executions were performed on an Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz, where 10 trials were run in parallel as independent processes. Then,

each data entry, i.e., the instance meta-features, was labeled as all algorithms that achieved the same average solution cost over the 30 runs.

Again, the algorithms were executed under the same computational time, such as in Chapter 6, that uses a fitted polynomial function to dicate the allowed time for an instance based on its size (Figure 6.1). Therefore, the meta-heuristics were executed for a particular instance according to the time limit in seconds given by (6.1). For the 5 instances of size greater than 300, this time was fixed to 5 hours. This time limit is not an actual run time prediction, since it was set with the only intention of giving the same computational resources for the algorithms in a fairly reasonable way.

Also, as previously observed, there are several small and easily solvable instances in which all algorithms achieve the same performance. This means that their meta-features may not be representative enough for performance prediction. Additionally, their presence in the learning dataset would deceptively increase the accuracy. Therefore, the dataset was built without these instances, resulting in the class distribution shown in Figure 7.1.

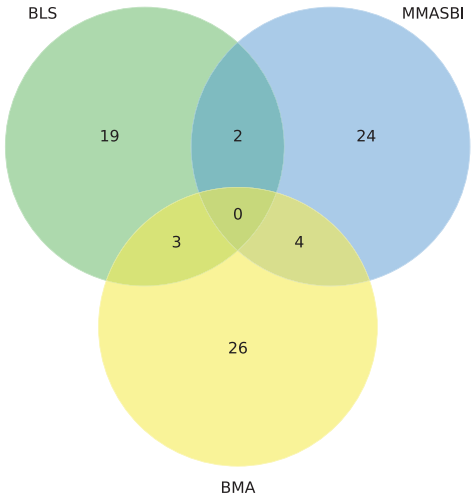


Figure 7.1: Multi-label dataset (BLS/MMASBI/BMA)

7.1.2 Sampling

The first set of meta-features were extracted by running the sampling strategy proposed by Stützle (2006), this set is the same used in Chapter 7. First, an Iterated Local Search (ILS) is shortly executed 1000 times, with 500 iterations each, and all achieved unique local optima solutions are stored. Next, a Best Improvement Local Search (BI) is randomly repeated until 5000 unique local optima is found.

Previously, we had set a time limit of 5 minutes for both stages. Since now we are also including larger instances, it became necessary to run this sampling for a longer period. Therefore, for running the ILS 1000 times, we set the same time limit as given by (6.1). For the BI, the execution stops prematurely if no new unique local optima is found after 1000 tries.

We refer to these sets of solutions as ILS/BI sampling. With them, the following meta-features were extracted:

- Number of pseudo global optima (n_opt): the best unique solutions found during all short runs of the ILS

- Fitness Distance Correlation (ils_fdc) and Average Distance to Optima (ils_dst) for the solutions set given by the ILS sampling
- Fitness Distance Correlation (bi_fdc) and Average Distance to Optima (bi_dst) for the solutions set given by the BI sampling

The new set of FLA meta-features, which are the main target of this study, are extracted by applying the Metropolis-Hastings (MH) algorithm. The MH 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. Also, this form of exploration is more relatable to the exploration performed by a searching algorithm (Vanneschi et al., 2004).

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

$$\alpha(x, y) = min\left\{1, \frac{y}{x}\right\} \tag{7.1}$$

Algorithm 1: Metropolis-Hastings

Result: The sample γ with m solutions
$\gamma_1 \leftarrow$ random solution
for $k = 2$ **to** , **do**
  repeat
    $\phi \leftarrow$ random solution
    $u \leftarrow$ random number from uniform (0,1) distribution
  until $u <= \alpha(f(\gamma_{k-1}), f(\phi))$
  $\gamma_k \leftarrow \phi$
  $k \leftarrow k + 1$
end

We run the MH until 5000 unique points are stored, which are then referred to as the base solutions. Moreover, we evaluate the prediction performance of the meta-features calculated using all 5000 solutions (the complete MH sampling), but also when using only the first 1000 solutions, which is called as the short MH sampling. With these base solutions, the following meta-features are computed for both the complete and short samples:

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

Then, for each base solution, a BI Local Search is applied and the unique local optima found are stored. The following meta-features are calculated using this set of local optima (also for the both sample sizes):

- Optima Fitness Coefficient (opt_fit_coef)
- Average Descent (avg_descent)
- Fitness Distance Correlation (fdc) and Average Distance to Optima (avg_dst): the best solutions among the local optima set are referred as the pseudo-global optima

- Optima Dispersion Metric (opt_dm)

All described sampling methodologies are executed on the same CPU that the meta-heuristics were run (Section 7.1.1), with at maximum 10 processes being run simultaneously. Figure 7.2 compares the execution times of the three samplings with the time spent when running all three meta-heuristics. It is noticeable that any sampling method requires less time than running all search algorithms for large instances. Moreover, the computational advantage of short MH sampling is visibly evident.

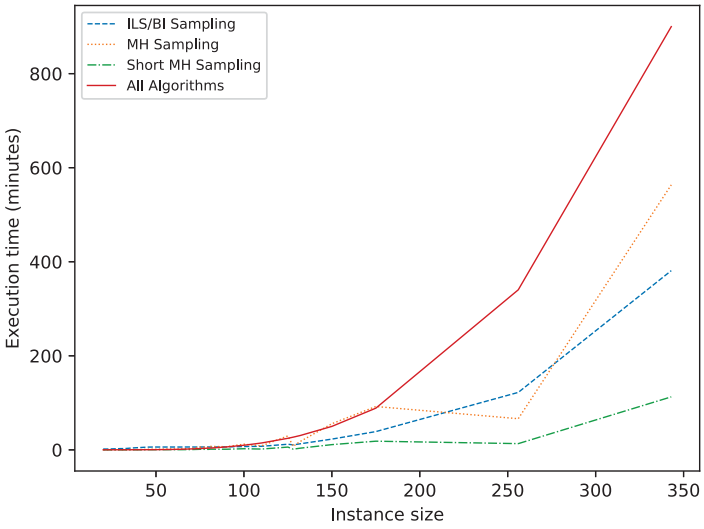


Figure 7.2: Comparison of sampling execution times

7.1.3 Classification

As meta-model, we used the Random Forest implementation from the *scikit-learn* library, with the parameters set to 100 estimators and a tree depth limit of 10. In order to keep the tests equivalent and reproducible, the seed for the random state parameter was fixed to 0. This Random Forest implementation was arbitrarily chosen because it works with multi-class and multi-label problems, also dispensing data normalization.

For validating the model, the Leave-one-out Cross Validation strategy was adopted. Thus, the performance measurements were obtained by micro-averaging the results in each train/test folding. The evaluation metrics used to assess the classification performances were the accuracy and the F-scores of each individual class. However, for multi-label classification, there are different types of accuracies that can be evaluated. In this work, we used the traditional instance accuracy, given by the Jaccard similarity coefficient between the true and predicted labels (Madjarov et al., 2012).

Additionally, a second type of accuracy was reported, which 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.

7.2 Results

As first experiment, we trained and evaluated the Random Forest model on the three datasets, with each having different meta-features, as discussed in Section 7.1.2. Specifically, the dataset obtained using the ILS/BI sampling contains the same 12 meta-features (7 static and 5 based on FLA) from our previous work. Meanwhile, the two datasets built with the MH sampling are made of the same 7 static meta-features plus the 7 new landscape metrics previously described.

Table 7.1 shows the accuracies on the three datasets with different sets of meta-features. Both the Metropolis-Hastings samplings resulted in better predictions, even with a slight advantage for the short version. Considering the small computation cost required, this recommendation accuracy of 75% is very satisfactory.

Table 7.1: Sampling meta-features accuracies

Sampling	Instance Accuracy	Recommendation Accuracy
ILS/BI	0.66	0.67
MH	0.71	0.74
Short MH	0.72	0.75

If we look at the individual labels F-scores, as displayed in Table 7.2, we can observe that, besides giving good accuracies, the short sampling meta-features also resulted in better and more balanced prediction across the classes.

Table 7.2: Labels F-scores

Sampling	BLS	MMASBI	BMA	Average
ILS/BI	0.61	0.77	0.66	0.68
MH	0.56	0.83	0.74	0.71
Short MH	0.65	0.81	0.73	0.73

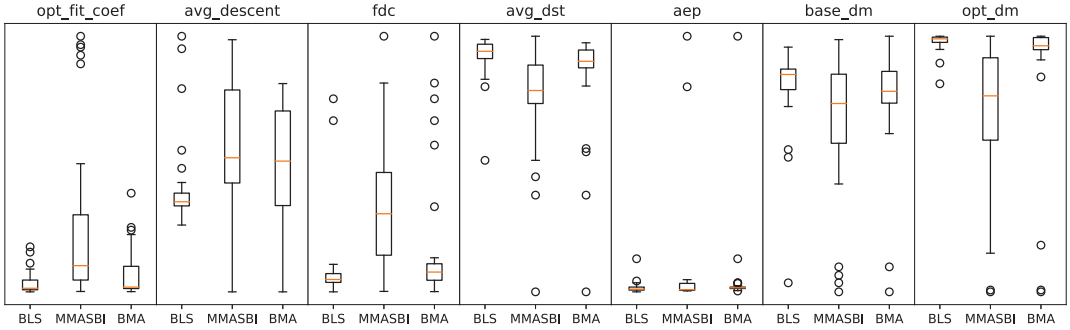
Next, because the meta-features extracted with the MH sampling yielded overall better results, they are further compared. Here, we evaluated the prediction performances when using only the FLA meta-features (without the static ones) of both complete and short sampling. The accuracies shown in Table 7.3 indicates that, in both cases, we could still achieve prediction performances above random. But now, without the aid of the static features, having a more complete sample results in a small improvement on accuracy.

Table 7.3: Accuracies with only MH samplings meta-features

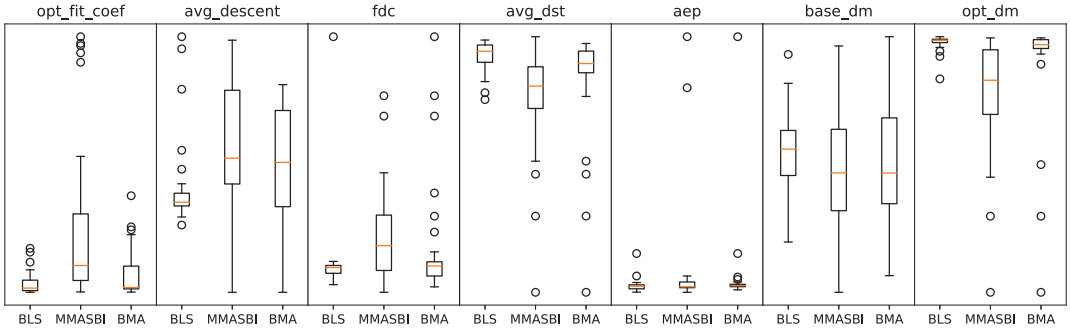
Sampling	Instance Accuracy	Recommendation Accuracy
MH	0.75	0.77
Short MH	0.68	0.70

Nevertheless, the huge reduction of computational cost for the short sample still makes it desirable, considering that the time consumption is so far a significant drawback when using FLA metrics for automatic algorithm selection. In fact, if we look at the features distributions of both samples in Figure 7.3, there are almost no visible differences.

Additionally, the good performances when using only the FLA meta-features are particularly interesting because, differently from the matrix measures, they can be extracted for any combinatorial problem.



(a) MH sampling



(b) Short MH sampling

Figure 7.3: Features distribution

At last, focusing on the short sample and on the FLA meta-features, we trained the model to perform binary classifications for each algorithm, like a one-vs-all strategy. The intention here is to better visualize the role of the meta-features to distinct the performances of the algorithms. The accuracies and F-scores of the positive class, shown in Table 7.4, remained as expected.

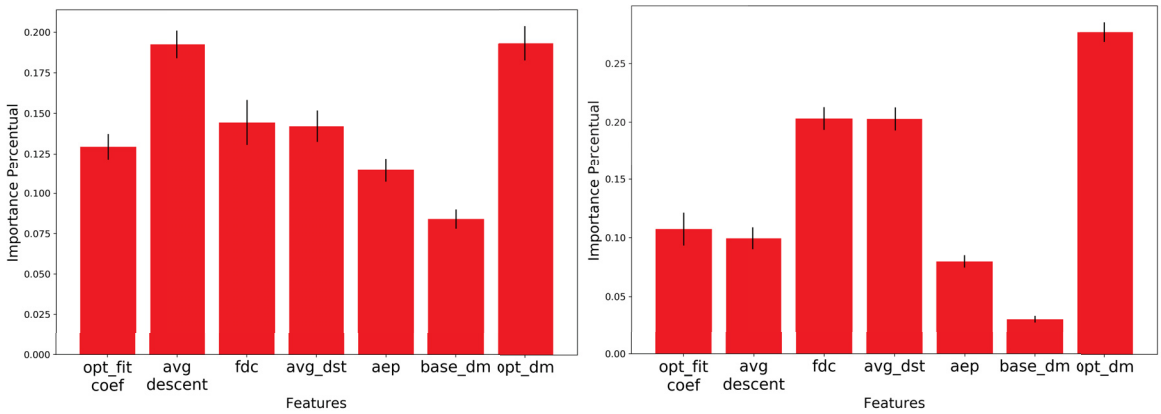
Table 7.4: Performance of binary classifications

Algorithm	Accuracy	F-Score Positive Class
BLS	0.79	0.65
MMASBI	0.87	0.83
BMA	0.76	0.71

The importances percentual of the features (reported by the scikit-learn implementation) for each algorithm are displayed in Figure 7.4. A first remark is that the dispersion of the local optima solutions remained as the most important meta-feature in the three cases, whereas the base dispersions did not have a significant role.

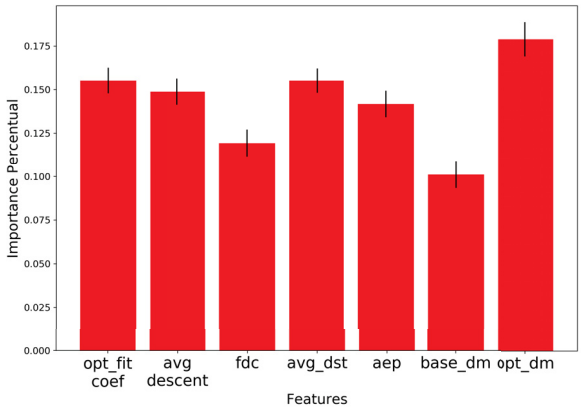
For the MMASBI classification, which had the best prediction accuracy, fewer meta-features had greater importances. Specifically, the dispersion and the FDC related measures were fundamental for the classification. If we look at the distribution of these features in Figure 7.3, we can notice that the instances labeled as MMASBI have a lower optima dispersion with a higher fitness correlation. This means that those instances are better solved by algorithms with a more exploitative behavior, which is the case for the MMASBI and most of the constructive meta-heuristics.

Meanwhile, the BLS and BMA classifications required the use of the all features more equally. This was not unexpected since the BMA also embeds the BLS, meaning that they



(a) BLS

(b) MMASBI



(c) BMA

Figure 7.4: Feature importances

have the same intensification properties. Nevertheless, we can see that the Accumulated Escape Probability, which is an evolvability metric, had a greater impact on classifying the evolutionary algorithm.

7.3 Discussion

This paper presented a study on meta-learning for algorithm selection applied to the Quadratic Assignment Problem. The main objetive was to investigate the use of some recently proposed meta-features based on Fitness Landscape Analysis. The results demonstrated that it is possible to obtain good classification performances by using metrics extracted without the need of a sampling methodology with a high time consumption, which is a recurring problem of FLA.

Nevertheless, we could also achieve satisfactory prediction results even if no problem specific meta-feature is included. Further, the good predictive power of the meta-features extracted with low cost of sampling methods, highlights the possibility to incorporate these features in other techniques that aims at automatising the solving of optimization problems, such as hyper-heuristics.

8 Conclusion

This research presented a set of incremental studies about algorithm selection with meta-learning concepts and measures of 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. Nevertheless, the second paper shows that even without the manual effort made in the previous study, the selection approach yield reliable results, at least better than using always only one algorithm to solve the instances.

Finally, the last work demonstrates the possibility of using measures of Fitness Landscape Analysis as meta-features without requiring a huge amount of computational effort, which is normally the case when dealing with FLA.

Moreover, since the chosen case study was the Quadratic Assignment Problem, which is known for being a very hard problem and also being a generalization of other combinatorial problems, it is likely that similar good performances may be achieved on different domains. This remains as an open question for future researches.

References

Anstreicher, K., Brixius, N., Goux, J.-P. e Linderoth, J. (2002). Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91(3):563–588.

Auger, A. e Teytaud, O. (2010). Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146.

Battiti, R. e Tecchiolli, G. (1994). The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140.

Beham, A., Affenzeller, M. e Wagner, S. (2017). Instance-based algorithm selection on quadratic assignment problem landscapes. Em *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, páginas 1471–1478, New York, NY, USA. ACM.

Benlic, U. e Hao, J.-K. (2013). Breakout local search for the quadratic assignment problem. *Applied Mathematics and Computation*, 219(9):4800–4815.

Benlic, U. e Hao, J.-K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1):584 – 595.

Blum, C., Puchinger, J., Raidl, G. R. e Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151.

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

Brazdil, P., Giraud-Carrier, C. G., Soares, C. e Vilalta, R. (2009). *Metalearning - Applications to Data Mining*. Cognitive Technologies. Springer.

Burkard, R. E., Cela, E., Pardalos, P. M. e Pitsoulis, L. S. (1998a). The quadratic assignment problem. Em *Handbook of Combinatorial Optimization*, páginas 1713–1809. Springer.

Burkard, R. E., Karisch, S. E. e Rendl, F. (1998b). Qaplib – a quadratic assignment problem library. *Journal of Global Optimization*, 10(4):391–403.

Chicano, F., Luque, G. e Alba, E. (2012). Autocorrelation measures for the quadratic assignment problem. *Applied Mathematics Letters*, 25(4):698–705.

Connolly, D. T. (1990). An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1):93 – 100.

Cung, V.-D., Mautor, T., Michelon, P. e Tavares, A. (1997). A scatter search based approach for the quadratic assignment problem. Em *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, páginas 165–169.

Cunha, T., Soares, C. e de Carvalho, A. C. (2018). Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering. *Information Sciences*, 423:128 – 144.

Dorigo, M. e Caro, G. D. (1999). Ant colony optimization: a new meta-heuristic. Em *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 2, página 1477.

Drezner, Z. (2003). A new genetic algorithm for the quadratic assignment problem. *INFORMS Journal on Computing*, 15(3):320–330.

Drezner, Z., Hahn, P. M. e Taillard, É. D. (2005). Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. *Annals of Operations Research*, 139(1):65–94.

Fischetti, M., Monaci, M. e Salvagnin, D. (2012). Three ideas for the quadratic assignment problem. *Operations Research*, 60(4):954–964.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206.

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

Jones, T. e Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. Em *Proceedings of the 6th International Conference on Genetic Algorithms*, páginas 184–192, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kanda, J., Carvalho, A., Hruschka, E. e Soares, C. (2011). Selection of algorithms to solve traveling salesman problems using meta-learning 1. *International Journal of Hybrid Intelligent Systems*, 8(3):117–128.

Koopmans, T. C. e Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: Journal of the Econometric Society*, páginas 53–76.

Langdon, W. B. e Poli, R. (2007). Evolving problems to learn about particle swarm optimizers and other search algorithms. *IEEE Transactions on Evolutionary Computation*, 11(5):561–578.

Lu, G., Li, J. e Yao, X. (2011). Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. Em Merz, P. e Hao, J.-K., editores, *Evolutionary Computation in Combinatorial Optimization*, páginas 108–117, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lunacek, M. e Whitley, D. (2006). The dispersion metric and the cma evolution strategy. Em *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, páginas 477–484, New York, NY, USA. ACM.

Madjarov, G., Kocev, D., Gjorgjevikj, D. e Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern recognition*, 45(9):3084–3104.

Maniezzo, V. e Colorni, A. (1999). The ant system applied to the quadratic assignment problem. *IEEE Transactions on knowledge and data engineering*, 11(5):769–778.

Misevicius, A. (2004). An improved hybrid genetic algorithm: new results for the quadratic assignment problem. *Knowledge-Based Systems*, 17(2):65–73.

Pardalos, L. e Resende, M. (1994). A greedy randomized adaptive search procedure for the quadratic assignment problem. *Quadratic Assignment and Related Problems, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 16:237–261.

Pitzer, E., Affenzeller, M., Beham, A. e Wagner, S. (2011). Comprehensive and automatic fitness landscape analysis using heuristiclab. Em *International Conference on Computer Aided Systems Theory*, páginas 424–431. Springer.

Pitzer, E., Beham, A. e Affenzeller, M. (2013). Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis. Em *Evolutionary Computation in Combinatorial Optimization*, Lecture Notes in Computer Science, páginas 109–120. Springer, Berlin, Heidelberg.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

Sahni, S. e Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565.

Schumacher, C., Vose, M. D. e Whitley, L. D. (2001). The no free lunch and problem description length. Em *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, páginas 565–570, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Smialowski, P., Frishman, D. e Kramer, S. (2009). Pitfalls of supervised feature selection. *Bioinformatics*, 26(3):440–443.

Smith-Miles, K. A. (2008). Towards insightful algorithm selection for optimisation using meta-learning concepts. Em *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, páginas 4118–4124.

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539.

Stützle, T. e Hoos, H. H. (2000). Max–min ant system. *Future Generation Computer Systems*, 16(8):889–914.

Taillard, É. (1991). Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5):443–455.

Taillard, E. D. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3(2):87–105.

Taillard, É. D. e Gambardella, L. M. (1997). Adaptive memories for the quadratic assignment problem. Relatório técnico, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

Tang, K., Peng, F., Chen, G. e Yao, X. (2014). Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94 – 104.

Vanneschi, L., Clergue, M., Collard, P., Tomassini, M. e Vérel, S. (2004). Fitness clouds and problem hardness in genetic programming. Em Deb, K., editor, *Genetic and Evolutionary Computation – GECCO 2004*, páginas 690–701, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wold, S., Esbensen, K. e Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37–52.

Wolpert, D. H. e Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Zhang, M.-L. e Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8).