UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS MARCONDES PAVELSKI

PER-INSTANCE ALGORITHM CONFIGURATION: FROM META-LEARNING TO MULTI-OBJECTIVE DECOMPOSITION

CURITIBA

2021

LUCAS MARCONDES PAVELSKI

PER-INSTANCE ALGORITHM CONFIGURATION: FROM META-LEARNING TO MULTI-OBJECTIVE DECOMPOSITION

Configuração de Algoritmos baseada em Instância: do Meta-aprendizado à Decomposição Multiobjetivo

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI) da Universidade Tecnológica Federal do Paraná (UTFPR) como requisito à obtenção do título de "Doutor em Ciências". Área de Concentração: Engenharia De Computação.

Orientadora: Profa. Dra. Myriam Regattieri De Biase da Silva Delgado
Coorientadora: Profa. Dra. Marie-Éléonore Kessaci

CURITIBA

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

LUCAS MARCONDES PAVELSKI

PER-INSTANCE ALGORITHM CONFIGURATION: FROM META-LEARNING TO MULTI-OBJECTIVE DECOMPOSITION

Trabalho de pesquisa de doutorado apresentado como requisito para obtenção do título de Doutor Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Computação.

Data de aprovação: 13 de Dezembro de 2021

Prof.a Myriam Regattieri De Biase Da Silva Delgado, Doutorado - Universidade Tecnológica Federal do Paraná

Prof.a Aurora Trinidad Ramirez Pozo, Doutorado - Universidade Federal do Paraná (Ufpr)

Prof.a Carolina Paula De Almeida, Doutorado - Universidade Estadual do Centro Oeste (Unicentro)

Prof.a Clarisse Dhaenens, Doutorado - University Of Lille

Prof Ricardo Luders, Doutorado - Universidade Tecnológica Federal do Paraná

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 21/12/2021.

Dedico este trabalho a minha família e aos meus amigos, pelos momentos de ausência.

ACKNOWLEDGEMENTS

Agradeço imensamente a todos que colaboraram direta ou indiretamente para a realização desse trabalho. Foi um grade desafio para mim e nunca esquecerei das pessoas que participaram dessa importante fase da minha vida.

A minha família, minha mãe Maria, meu pai Floriano, e minha irmã Alinne, pelas orações, pelo carinho, incentivo e total apoio em todos os momentos.

A queridíssima e amada Derli, que foi a melhor companheira que eu poderia ter nesses momentos. Pelas conversas longas, pela paciência e pelo carinho.

A minhas orientadoras, prof. Myriam e prof. Marie. Pelos valiosos conselhos, por servirem de exemplo, e pelas inúmeras horas de dedicação a esse trabalho. Certamente não conseguiria sem vocês.

A todos meus amigos. Pelo apoio nesse período e, mesmo eu por vezes estando ausente, sempre estiveram firmes como um porto seguro.

Enfim, a todos os que de alguma forma contribuíram para a realização deste trabalho, estarão sempre no meu coração.

A espantosa realidade das coisas
É a minha descoberta de todos os dias.
Cada coisa é o que é,
E é difícil explicar a alguém quanto isso me alegra,
E quanto isso me basta.
(CAEIRO, Alberto, 1889).

RESUMO

PAVELSKI, Lucas. **Configuração de Algoritmos baseada em Instância: do Meta-aprendizado à Decomposição Multiobjetivo**. 2021. 131 f. Tese (Doutorado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

A busca pelo melhor algoritmo e sua configuração é uma tarefa difícil na maioria dos cenários de otimização, especialmente em problemas NP-difíceis, uma vez que existem diferentes meta-heurísticas propostas, e testar muitos parâmetros demanda um alto custo computacional. Além disso, o entendimento de tais parâmetros e sua relação com as instâncias do problema é de suma importância para a área de configuração de algoritmos. A literatura sobre Configuração Automática de Algoritmos (AAC do inglês *Automatic Algorithm Configuration*) propõe várias estratégias para encontrar a melhor configuração, embora o foco geralmente seja menos na explicabilidade e mais no desempenho dessas diferentes configurações. Com base em experiências anteriores obtidas a partir de dados, a área de Configuração Automática Baseada em Instância (PIAC, do inglês *Per Instance Algorithm Configuration*) foca no mapeamento construído para recomendar as melhores configurações. Este trabalho tem como objetivo propor e analisar duas abordagens PIAC. A primeira, chamada de MetaL PIAC, é uma extensão do problema de seleção de algoritmo e usa meta-aprendizado para recomendar meta-heurísticas e seus parâmetros de configuração. A segunda abordagem, chamada de MOAAC/D, é baseada em uma nova formulação multiobjetivo do problema AAC, que decompõe o espaço do problema e usa uma plataforma baseada em decomposição para fornecer configurações generalistas e especialistas ao mesmo tempo. Para cada objetivo, existe um conjunto de problemas relacionados a ele, e um algoritmo multiobjetivo baseado em decomposição é proposto para encontrar configurações com bons balanceamentos. Como estudo de caso principal, o trabalho considera problemas Flowshop. Experimentos extensivos realizados em mais de 6000 instâncias, consideram o MetaL PIAC para ajustar os parâmetros de diferentes meta-heurísticas, e o MOAAC/D para ajustar configurações da busca local iterativa e busca gulosa iterativa. Os resultados mostram que ambas as estratégias superam a solução generalista fornecida pelo irace – uma das abordagens de AAC mais conhecidas na área – com uma ligeira vantagem do MOAAC/D sobre o MetaL PIAC.

Palavras-chave: configuração automática de algoritmos. meta-aprendizado. otimização multiobjetivo. decomposição do espaço do problema. problemas flowshop.

ABSTRACT

PAVELSKI, Lucas. **Per-Instance Algorithm Configuration: from Meta-learning to Multi-objective Decomposition**. 2021. 131 p. Thesis (Doctorate in Electrical and Computer Engineering) – Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

The search for the best algorithm and its configuration is a difficult task on most optimization scenarios, especially on NP-hard problems, since different proposed metaheuristics exist, and testing many parameters demands high computational costs. Moreover, the understanding of such parameters and their relation to problem instances is of great importance in the field of algorithm configuration. The literature on Automatic Algorithm Configuration (AAC) proposes several strategies to find out the best configuration, although the focus is usually less on explainability and more on the performance of the different configurations. Based on past experience obtained from data, Per-Instance Algorithm Configuration (PIAC) focuses on the mapping built to recommend the best configurations. This work aims at proposing and analyzing two PIAC approaches. The first, namely MetaL PIAC, is an extension of the algorithm selection problem and uses meta-learning to recommend metaheuristics and their configuration parameters. The other, namely MOAAC/D is based on a brand new multi-objective formulation of the AAC problem, that decomposes the problem space and uses a decomposition-based framework to provide generalist and specialist configurations at the same time. For each objective, there is a set of problems related to it, and a decomposition based multi-objective algorithm is proposed to find good trade-off configurations. As the main study case, the work addresses flowshop problems. Extensive experiments performed on more than 6000 instances, consider MetaL PIAC to tune the parameters of different metaheuristics, and MOAAC/D to tune iterated local search and iterated greedy configurations. The results show that both strategies outperform the generalist solution provided by irace – one of the best well-known AAC – with a slight advantage of MOAAC/D over MetaL PIAC.

Keywords: automatic algorithm configuration. meta-learning. multi-objective optimization. problem space decomposition. flowshop problems.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

INITIALISM	
AAC	Automatic Algorithm Configuration
ACO	Ant Colony Optimisation
ARPD	Average Relative Performance Difference
AUC	Area Under the receiver operating characteristic Curve
BR	Binary Relevance
BRPLUS	Binary Relevance Plus
CART	Classification and Regression Tree
CC	Classifier Chain
CLR	Calibrated Label Ranking
CTRL	ConTRolled Label correlation exploitation
DBR	Dependent Binary Relevance
E-IRACE	Original irace algorithm optimizing a particular Extreme of the Pareto front
EBR	Ensemble of Binary Relevance
ECC	Ensemble of Classifier Chains
ESL	Ensemble of Single Label
FDC	Fitnes Distance Correlation
FLA	Fitness Landscape Analysis
FS	Feature selection
FSP	FlowShop Problem
G-IRACE	the original irace algorithm used to provide a generalist configuration
HC	Hill Climbing
HOMER	Hierarchy Of Multi-label classifiER
IG	Iterated Greedy
ILS	Iterated Local Search
iMOEA/D	MOEA/D with (optional) irace operator for automatic algorithm configuration
ISA	Instance Space Analysis
LON	Local Optima Network
LP	Label Powerset
LSPS	Local Search on Partial Solution
MAE	Mean Absolute Error
MBR	Meta Binary Relevance
MetaL PIAC	Meta-Learning based Per-Instance Algorithm Configuration
MH	Metaheuristic

MMAS	Min-Max Ant System
MOAAC/D	Multi-objective Automatic Algorithm Configuration based on problem space Decomposition
MOEA	Multi-objective evolutionary algorithm
MOEA/D	Multiobjective Evolutionary Algorithm Based on Decomposition
NEH	Nawaz-Enscore-Ham
NILS	Neutral Iterated Local Search
NS	Nested Stacking
PCA	Principal Component Analysis
PIAC	Per-Instance Algorithm Configuration
PP	Performance-Priority tie-breaking between recommended MHs
PRUDENT	Pruned and Confident Stacking Approach
RAKEL	RAndom k-labELsets
randPIAC	Recommends a random configuration
RDBR	Recursive Dependent Binary Relevance
RF	Random Forest
RPC	Ranking by Pairwise Comparison
SA	Simulated Annealing
TS	Tabu Search
XGB	eXtreme Gradient Boosting

LIST OF SYMBOLS

LATIN LETTERS

α	An algorithm in the space of algorithms
A	The space of algorithms
$budget$	Flowshop problem maximum budget
$\boldsymbol{\theta}$	Algorithm configuration vector
Θ_α	Configuration space of algorithm α
$corr$	Processing times correlation
$dist(\mathbf{x}, \mathbf{x}')$	Distance between solutions $\mathbf{x}$ and $\mathbf{x}'$
$distribution$	Processing times distribution
$down(\mathbf{x})$	Number of down edges of a solution $\mathbf{x}$
ϵ	Random walk ensemble sensitivity parameter
$f(p)$	Feature extraction function
F	Feature space
F	Input or feature space
$\boldsymbol{\phi}$	A feature vector
$\hat{\rho}(s)$	Estimated random-walk autocorrelation with step s
$\mathcal{PF}$	Pareto front
$\mathcal{PS}$	Pareto set
E_η	Edges of a fitness landscape graph connecting neighbor solutions
$\eta : \mathscr{S} \to \wp(\mathscr{S})$	A neighborhood relation in a solution space $\mathscr{S}$
N_{obj}	Number of objectives
N_{opt}	Number of global optima solutions
$g : \mathbf{x} \to \mathbb{R}$	Objective or performance function
$\mathbb{O}$	Output or label space
π	Algorithm parameter
U	The performance space
$\wp(A)$	Powerset of a set A
p	A base problem sample
P	The space of problems
$\mathbb{S}$	Ranking space
$h(\boldsymbol{\phi})$	Recommendation function
$\rho(s)$	Random-walk autocorrelation with step s
$\rho_{FDC,dist}$,$\hat{\rho}_{FDC,dist}$	FDC and estimated FDC with distance metric $dist$
T	Random walk length
$side(\mathbf{x})$	Number of side edges of a solution $\mathbf{x}$
$\sigma[\cdot]$	Standard deviation statistic
$\mathscr{S}$	Solution space
$stopC$	Flowshop problem stopping criterion

$up(\mathbf{x})$	Number of up edges of a solution $\mathbf{x}$
$u(\boldsymbol{\theta})$	Utility function value for configuration $\boldsymbol{\theta}$
$\boldsymbol{f}$	Vector of objective function values
$\mathbf{x}_L$	Decision variables lower bounds
$\mathbf{x}_U$	Decision variables upper bounds
$\mathbf{w}_i$	Weight of objective i
B_i	Neighborhood associated with the i-th solution $\mathbf{x}_i$
$Cov[\cdot,\cdot]$	Covariance statistic
$d_{abs}(\mathbf{x},\mathbf{x}')$	Absolute distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$d_{adj}(\mathbf{x},\mathbf{x}')$	Adjacency distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$d_{dev}(\mathbf{x},\mathbf{x}')$	Deviation distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$d_{pre}(\mathbf{x},\mathbf{x}')$	Precedence distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$d_{shift}(\mathbf{x},\mathbf{x}')$	Shift distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$d_{swap}(\mathbf{x},\mathbf{x}')$	Approximated swap distance between permutations $\mathbf{x}$ and $\mathbf{x}'$
$E[\cdot]$	Mean statistic
g^{tche}	Tchebycheff aggregation function
$H(\epsilon)$	Random-walk entropy with sensitivity ϵ
$h(\epsilon)*$	Random-walk density basin with sensitivity ϵ
L_ρ	Random-walk autocorrelation length
N_{gen}	Number of generations
N_{neigh}	Number of neighbors
N_{pop}	Population size
$S(\epsilon)$	Ensemble of a random walk correlation, indicating up, down and side movements
$C_{x_j,m}$	Completion time of job x_j on machine m
C_{max}	Makespan, the maximum completion time of a given job sequence
due_j	Due time of job j
E_j	Early time or earliness of job j
F	Total flowtime of a given job sequence
F_j	Total processing time or flowtime of job j
J	Number of jobs
M	Number of machines
N_{IPLAT}	Plateau solutions
N_{LEDGE}	Ledge solutions
N_{LMAX}	Local maxima solutions
N_{LMIN}	Local minima solutions
N_{SLMAX}	Strict local maxima solutions
N_{SLMIN}	Strict local minima solutions
N_{SLOPE}	Slope solutions
$p_{j,M}$	Processing time required of job j on machine m
rel_j	Release time of job j
S	Number of samples to compute in utility evaluation

T_j	Delay time or tardiness of job j
T_{max}	Maximum tardiness of a given jobs sequence
U	Number of tardy jobs
$U_{we,wt}$	Weighted earliness (wt) and tardiness (we)

INDICES AND SETS

$j \in \{1, \ldots, J\}$	Job index
$m \in \{1, \ldots, M\}$	Machine index in the order of operations
$\mathbf{x} = (x_0, \ldots, x_J)$	Flowshop solution or job permutation

CONTENTS

1 INTRODUCTION

The algorithm selection problem (RICE, 1976) consists in recommending or choosing, among all the available algorithms, the best one to solve a given problem (namely base problem), as an attempt to mitigate the considerable effort of trying different possibilities for each previously unseen problem. The goal of algorithm selection is to automatically find algorithm configurations without the need of expert knowledge or rules of thumb (BIRATTARI, 2009). Moreover, the recommendation model can be used to gain insights into the base problem, combine different algorithms, or even develop new heuristics aiming to achieve better performance (XU *et al.*, 2008).

The problem of algorithm recommendation is related to many other problems, like instance-based algorithm selection, static and dynamic algorithm schedules, parallel algorithm portfolios, automatic algorithm configuration, and others. According to (HUTTER *et al.*, 2014), in Automatic Algorithm Configuration (AAC), specialized methods optimize parameter values to find the best configuration of an algorithm for a given base problem, which can be anyone solved by an algorithm with more than one possible configuration, e.g., optimization of machine learning problems.

One of the most powerful generalizations of those recommendation tasks is Per-Instance Algorithm Configuration (PIAC). For that, based on past experiences, a proper configuration is recommended considering the instance's meta-features. Ideally, Per-Instance Algorithm Configuration (PIAC) strategies produce close to ideal performance with low cost and without the burden of testing different configurations. In this sense, PIAC is a generalization of the Automatic Algorithm Configuration (AAC) problem. Despite that, there are not many proposed strategies in the literature and PIAC remains as an open problem due to the complexity of the recommendation space (KERSCHKE *et al.*, 2019).

The main field dedicated to study learning of algorithms design based on past experiences is called meta-learning, and it has been proved to be very helpful to deal with the algorithm selection problem (SMITH-MILES, 2009). At the base problem level, given a set of instances and algorithms, a performance dataset is built through the experience of solving the base problem by each of the available algorithms. In the meta-level, useful problem features like instance's size and statistics and information about the problem space are extracted, forming the meta-features dataset. Finally, using this meta-data, the system can build a meta-model to help with base

problem decision-making like building algorithm recommendation, algorithm ensembles, and transfer of problem-solving knowledge to other domains.

Recently there is a growing interest in the application of meta-learning and AAC to optimization problems (PAPPA *et al.*, 2014; HUTTER *et al.*, 2014; BISCHL *et al.*, 2016). Optimization is an essential tool for decision making and analysis of physical systems in many branches of science. A quantitative objective function measures the system's quality or cost, like product's performance or energy consumption. The system's characteristics can be modeled as variables (namely, decision variables). The general goal is to find decision variables' values that optimize the objective function. This goal can be approached from a variety of methods, depending on the objective function formulation, domain of the decision variables and optimization hardness (GOLDBERG, 1989; NOCEDAL; WRIGHT, 2006; FLETCHER, 2013). When the problem has multiple conflicting objective functions to be optimized simultaneously, multi-objective optimization algorithms can be used to search for, not only one, but a set encompassing specialists and compromise solutions called Pareto set (DEB, 2011).

The present work proposes two general PIAC approaches for optimization problems:

- The first proposal is based on meta-learning and can be considered as an extension to the algorithm selection problem, where irace is used to generate the performance dataset and multi-label machine learning models are trained to recommend Metaheuristics (MHs) and configurations for unseen test problems;

- The second proposal is based on a new multi-objective formulation of AAC. This formulation considers that each objective is associated with different sets of problem instances, where specialist configurations would optimize single objectives on the extremes of the Pareto-front approximation and generalist configurations would be part of the trade-off. Additionally, we propose a hybridization of Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) with irace as an efficient method for exploring compromises between configurations.

The first proposal can be considered mono-objective in the sense that it performs an instance-wise search for the best configuration. The second is considered multi-objective since it uses a multiple instances during the configurations search. The proposals combine efforts of fitness landscape analysis and existing AAC strategies in a novel way. We also focus on interpretable recommendations, as we use tree-based machine learning models in the first

proposal and gain insights into the base problem by analyzing the Pareto set approximation in the second one.

Although we have obtained preliminary results for two additional domain problems (binary, and continuous optimization), the main case study relies on the flowshop problem, that is an example of a combinatorial optimization problem (BAKER; TRIETSCH, 2009). It models a sequential production line in which every machine has to perform a set of operations aiming to complete a set of jobs. Scheduling the jobs in order to minimize objectives like total production time and job delays is the goal of flowshop. Since its conception by Johnson (1954), there are several approaches to solve flowshop problems and their variants. The general flowshop problem is known to be NP-hard (GAREY *et al.*, 1976). Therefore there is no known efficient method to solve arbitrarily large instances optimally. In the present work, the experiments include an extensive analysis of the proposals, using more than $6{,}000$ permutation flowshop problems, with different features such as size, objective, processing time distributions and budgets.

Many heuristics and MHs are proposed to approximate flowshop optimal solutions (HOOS; STÜTZLE, 2004; RUIZ; MAROTO, 2005). Heuristics like the Nawaz-Enscore-Ham (NEH) (NAWAZ *et al.*, 1983) are low-cost algorithms (compared to exact-solvers) that try to construct and improve a solution using problem information. MHs like the Iterated Greedy (IG) (RUIZ; MAROTO, 2005) are problem-independent general heuristics that usually try to improve a solution iteratively. In the experiments we considered metaheuristics based on hill-climbing, simulated annealing, tabu-search, ant-colony optimization, iterated local search and IG. These metaheuristics include several parameters like local search type, neighborhood size, cooling schedules, and perturbation type.

1.1 OBJECTIVES

This work aims to propose and analyze on the context of flowshop problems, two approaches for the per-instance algorithm configuration problem: one based on meta-learning and other based on multi-objective problem space decomposition.

The specific objectives include:

1. To formulate the PIAC problem under the Rice's framework perspective. This formulation is general and can be applied to any base problem;

2. To propose a meta-learning-based approach, namely MetaL PIAC, to solve the formulated

PIAC problem;

3. To propose a new formulation for AAC, namely Multi-objective Automatic Algorithm Configuration based on problem space Decomposition (MOAAC/D), as a multi-objective framework for decomposing the base problem space and to further extend it to the general PIAC problem;

4. To study flowshop instances, including different objective functions and processing time distributions;

5. To provide a representative set of MHs used to solve flowshop instances, ranging from classic to state-of-the-art variants including their highly configurable parameter space;

6. To test different parameter tuning techniques, aiming to find the best configuration for a set of similar instances and build the performance dataset;

7. To implement different fitness landscape analysis metrics, analyze the generated problems dataset and build the meta-features dataset;

8. To experiment MetaL PIAC with different machine learning models and strategies that use meta-data for MH and parameter recommendation, focusing on interpretable models;

9. To propose a local search for MOEA/D, to solve MOAAC/D formulation, efficiently finding specialists and compromise configurations on the decomposed base problem space;

10. To compare the recommendations provided by MetaL PIAC and MOAAC/D with other recommendation strategies.

Overall, considering the objectives and the two proposals, the research hypothesis can be summarized as: *per-instance strategies can generate configurations with good quality, even for unseen problem instances, using either mono-objective based on meta-learning or multi-objective based on problem space decomposition.*

1.2 SCOPE AND MOTIVATIONS

As the case for several optimization problems, practitioners have a huge variety of possibilities to solve them. This usually implies in a deep dive into the state-of-the-art by comparing performance of different algorithms. Hopefully, the performance can be drawn on

problems similar to the one at hand. However, more often than not, knowing relevant problem features and comparing instances is altogether another deep dive.

The difficulties are not only a problem for the practitioner, but also reflect on the paths taken by optimization research. In (HOOKER, 1994; HOOKER, 1995), the author criticizes the benchmark-driven algorithm evaluation methodology, pointing that the results report merely *which* algorithm is better instead of *why* it is better for the testbed. Also, the MH field is criticized for propagating proposals with different names for algorithms with equal or very similar inner workings to existing ones or over-tuning on specific benchmarks only to claim novelty and better results. Theses aspects can obfuscate important analysis on truly innovative approaches that contribute to the state-of-the-art (SÖRENSEN, 2015).

Another important consideration is the No Free Lunch theorems presented by Wolpert and Macready (1997). The theorems state that no single algorithm is capable of performing well on all problems. This implies that in a broad problem space, algorithms should be diverse: different algorithms specialized for different problems. Another implication is that averaging performance values over a set of different instances is not too informative and is often not suitable for comparing algorithms (SMITH-MILES *et al.*, 2014).

Currently, there are important developments in areas dedicated to analyze and propose strategies that overcome the issues of metaheuristic developments, such as:

- Algorithm selection: defined by Rice (1976), the algorithm selection problem consists in finding the map between problem features and algorithm performance. There are several proposals of algorithm recommendation for different optimization problems, even improving results of hand-crafted strategies (KERSCHKE *et al.*, 2019);

- Automatic Algorithm Configuration: the idea of AAC is to turn the algorithm parameter setting as a high level optimization problem. One of the first instances of AAC was proposed by Grefenstette (1986), where a meta-genetic algorithm is used to tune itself on a base problem. AAC and algorithm selection are closely related. They overcome the burden of hand picking different strategies and can produce better and more reproducible results (MONTERO *et al.*, 2014; KERSCHKE *et al.*, 2019);

- Hyper-heuristics: aim at finding a self-adaptive parameter setting. Online hyper-heuristics are proposed to generate or select the best strategy (or parameter value) during the search. These strategies often use learning mechanisms that receive feedback from their choices

during the search process. These methods aim to solve a wide variety of problems without calibration (BURKE *et al.*, 2019).

- Fitness Landscape Analysis (FLA): takes inspiration from theoretical biology where the goal is to analyze and visualize the relationships between genotypes, phenotypes and fitness (PITZER; AFFENZELLER, 2012; RICHTER; ENGELBRECHT, 2014; OCHOA; MALAN, 2019). Many studies in this area propose metrics for measuring problem hardness by investigating the searched *landscape*: neighbor solutions (connected by perturbation or local search operators) having its fitness as the *height*. FLA main uses are understanding the effects of problem features (REEVES, 1999; CZOGALLA; FINK, 2012; OCHOA; MALAN, 2019), analyzing local search and perturbation operators behavior (OCHOA *et al.*, 2008), performance prediction (WATSON *et al.*, 2005), and algorithm recommendation (BISCHL *et al.*, 2012).

- Instance Space Analysis (ISA): studies the instance space to identify strengths and weaknesses of an algorithm. It does so by modeling structural properties of an instance and the performance of a set of algorithms. The main motivations are automatically defining *partitions* of the problem space where a given algorithm performs well, as well as determining whether the (benchmark) instance set covers the problem space (SMITH-MILES; LOPES, 2012; SMITH-MILES *et al.*, 2014).

- White-box metaheuristic research: promotes component-based metaheuristic development, making it better for sharing operators and results for reproducibility, integrating problem-specific knowledge, and also automatic search on metaheuristic design spaces (HUMEAU *et al.*, 2013; SWAN *et al.*, 2021). Several applications of white-box frameworks show good results on several problem domains (MARMION *et al.*, 2013; Alfaro-Fernández *et al.*, 2020);

- Multi-objective approaches for automatic algorithm configuration: this is an under-explored area. The few existing works consider (i) the addition of other objectives than the traditional base problem fitness (ZHANG *et al.*, 2015; BLOT *et al.*, 2016), and (ii) optimizing more than one budget at once (DRÉO, 2009; DYMOND *et al.*, 2017).

There are also proposals of unifying these fields to share common knowledge, even related disciplines from machine learning (SMITH-MILES, 2009). As such, the scope and

motivations followed by the present work span from different approaches with a common objective: use data-driven learning techniques and multi-objective optimization to explore, predict and solve optimization problems. The approaches we use and propose are problem agnostic but we decide to apply them mainly to flowshop problems. Moreover, the multi-objective approach being proposed can be considered a generalization of the few existing works, since the base problem fitness or budget is one among all the possible features being optimized.

1.3 CONTRIBUTIONS

This work applies concepts from several works in the literature of flowshop problems (PINEDO, 2016), MHs (HOOS; STÜTZLE, 2004; RUIZ; STÜTZLE, 2007), fitness landscape analysis (PITZER; AFFENZELLER, 2012) and automatic algorithm configuration (López-Ibáñez *et al.*, 2016). It also has several related works from meta-learning literature, mainly on combinatorial problems (KANDA *et al.*, 2016; DANTAS; POZO, 2018), scheduling problems (SMITH-MILES *et al.*, 2009) and PIAC meta-learning for optimization proposals (KADIOGLU *et al.*, 2010). Nonetheless, we can summarize our novel contributions as:

- The proposal and analysis of two PIAC strategies using parameter tuner and fitness landscape analysis meta-data;
- A brand new formulation of AAC based on problem space decomposition;
- The comparison between several multi-label models for the task of algorithm recommendation.
- A broad study and analysis on solving over $6{,}000$ flowshop problems with different MHs, from classic to state-of-the-art algorithms;
- An in-depth fitness landscape analysis on different flowshop variants;

The present work is also based on seven (one in an international journal) works published by the authors:

1. PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore. Meta-Learning for Optimization: A Case Study on the Flowshop Problem Using Decision Trees. *In*: **2018 IEEE Congress on Evolutionary Computation (CEC)**. *[S.l.: s.n.]*, 2018. p. 1–8.

In this work, decision trees are used for MH and parameters recommendation for three flowshop variants, using the data from irace parameter tuner on single instances and simple flowshop instance features;

2. PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Recommending Meta-Heuristics and Configurations for the Flowshop Problem via Meta-Learning: Analysis and Design. *In*: **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**. *[S.l.: s.n.]*, 2018. p. 163–168.

   In this paper, we perform a more in-depth meta-data analysis and extreme boosting machines are used for MH and parameter recommendation, using fitness landscape analysis as meta-features;

3. PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore. Meta-learning on Flowshop Using Fitness Landscape Analysis. *In*: **Proceedings of the Genetic and Evolutionary Computation Conference**. New York, NY, USA: ACM, 2019. (GECCO '19), p. 925–933. ISBN 978-1-4503-6111-8.

   In this work, neural network models perform MH recommendation, pairwise ranking, parameter recommendation, and parameter distribution estimation.

4. PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore; FREITAS, Alex A. Stochastic local search and parameters recommendation: A case study on flowshop problems. **International Transactions in Operational Research**, p. itor.12922, Dec. 2020. ISSN 0969-6016, 1475-3995.

   This paper describes the main contributions of the multi-label models for the task of algorithm recommendation and tree-based learning models for PIAC task.

5. PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Local Optima Network Sampling for Permutation Flowshop. *In*: **2021 IEEE Congress on Evolutionary Computation (CEC)**. Kraków, Poland: IEEE, 2021. p. 1131–1138. ISBN 978-1-72818-393-0.

   In this work, we present a fitness landscape analysis of medium to large-scale flowshop problems, using local optima networks and state-of-the-art algorithms like IG.

6. PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Flowshop NEH-Based Heuristic Recommendation. *In*: ZARGES, Christine; VEREL, Sébastien

(Ed.). **Evolutionary Computation in Combinatorial Optimization**. Cham: Springer International Publishing, 2021. v. 12692, p. 136–151. ISBN 978-3-030-72903-5 978-3-030-72904-2.

This paper results show that the PIAC meta-learning approach can be used to automatically tune the parameters of a flowshop NEH-based heuristic and provide interpretable models.

7. PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Dynamic Learning in Hyper-Heuristics to Solve Flowshop Problems (to appear). *In*: **2021 7th Brazilian Conference on Intelligent Systems (BRACIS)**. *[S.l.: s.n.]*, 2021. p. 1–15.

   As an extension to the proposals of the present work, this paper presents a novel hyper-heuristic based IG algorithm with adaptive online configuration.

Detailed results achieved by the multi-objective algorithm configuration proposal and its generalization to perform PIAC task are in preparation to be submitted to a conference and high-quality journal paper.

1.4 STRUCTURE

The work is organized as follows. After this introduction, Chapter 2 provides a review and background on essential topics like PIAC, meta-learning, fitness landscape analysis, multi-objective optimization, flowshop problem and also works related with the main subjects addressed in the present work. Chapter 3 describes the proposed meta-learning based model as well as the multi-objective formulation for AAC and PIAC problems. Chapter 4 is dedicated to contextualize the proposals on the application domain - flowshop problems - and it also describes the experiments performed to evaluate the proposals on the flowshop context. Chapter 5 presents some results achieved by both proposals, as well as a comparison that highlights their pros and cons of each one. Finally, Chapter 6 concludes this work and presents future works.

2 BACKGROUND

This chapter presents topics necessary to understand the proposals like recommendation of algorithms/parameters and meta-learning, fitness landscape analysis, mono and multi-objective optimization, and flowshop problem. The chapter also discusses works whose content is directly related to the thesis subject.

2.1 ALGORITHM/PARAMETER RECOMMENDATION

The algorithm recommendation (selection) problem introduced by Rice (1976) is a classic problem in computer science. It consists in finding the mapping between problem features and the algorithm with the maximum performance on the problem. Usually this mapping is built using the past information on solving similar problems and algorithms. As we can see, the problem tacked by the algorithm selection/recommendation operates a level above the base problem (the one it tries to infer the best algorithm for) (LEMKE *et al.*, 2015). Therefore it is also called meta-learning, since it uses past information (learning) from the performance data from the base problem (meta).

According to Brazdil *et al.* (2008), in the machine learning context, it is essential to distinguish between base-learning and meta-learning. Learning at the base level focuses on accumulating experience on a specific learning task, whereas learning at the meta-level concerns with accumulating experience on the performance of multiple applications of a learning system. Therefore, applying a base-learner (e.g., decision tree, neural network, or support vector machine) on some data produces a predictive function that depends on the fixed assumptions embedded in the learner. Working at the base level exhibits two significant limitations. First, data patterns are usually not placed aside for interpretation and analysis, but instead embedded in the predictive function itself. Second, the experience or knowledge gained when applying a learning algorithm using one dataset is generally not readily available as we move to another dataset.

In optimization problems, the primary use of meta-learning is algorithm recommendation, where the goal is to predict the best algorithm for a new instance (SMITH-MILES, 2009). As previously discussed, even though in an optimization context the prefix “meta-” might not be appropriate since the learning and knowledge in the base level may be absent, the literature on meta-learning for optimization has maintained the nomenclature. Therefore, in this work, build-

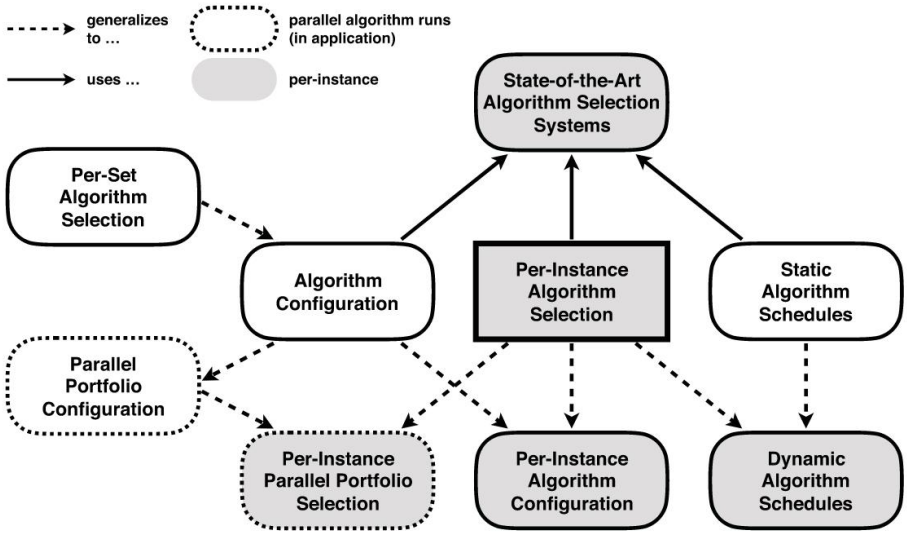
ing heuristic recommendation models based on the knowledge extracted from problem instances and algorithm performance will be referred to as meta-learning based on metaknowledge.

Lemke *et al.* (2015) consider that a meta-learning system includes a learning subsystem, which adapts with experience. The experience might come from previous knowledge of different runs or different domains or problems. Therefore, meta-learning is a vast area of study encompassing algorithm recommendation, dynamic bias selection, transfer learning, and exploratory analysis of metaknowledge. Even though other applications might not consider base-learner algorithms (like MHs in optimization tasks), the term meta-learning is used to describe systems developed to recommend algorithms in a broad set of applications like ensembles, machine learning, optimization, time-series prediction, operational research, and algorithm design.

A key to solving these problems is gathering knowledge about the learning process itself. Such knowledge (called *metaknowledge*) can be used to improve the learning mechanism after each training episode. Metaknowledge is any knowledge derived in the course of employing a given learning system and may take on different forms and applications (BRAZDIL *et al.*, 2008). The term *meta-features* is usually used to define the different features that are used to characterize the problem solved on the meta-level. For example, on flowshop problem, some simple meta-features include the problem size while complex meta-features can derive from FLA metrics.

Figure 1 shows how different problems on algorithm selection or recommendation and algorithm configuration are related (KERSCHKE *et al.*, 2019).

Figure 1 – Algorithm selection and algorithm configuration problems.



Source: Adapted from Kerschke *et al.* (2019)

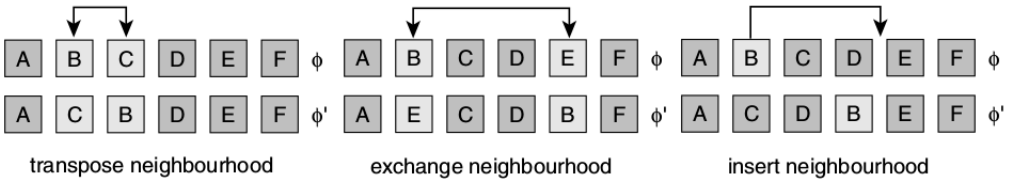
We can see that AAC, per-instance algorithm selection and algorithm schedules generalize the algorithm selection problem. Also, Per-Instance Algorithm Configuration (PIAC) is one of the most general approaches, since it can solve AAC and per-instance algorithm selection. Different from the algorithm selection, PIAC uses information from individual instances and produces a configuration tailored to the problem. It mainly differs from AAC in that it produces a meta-learning model instead of a fixed number of configurations for a whole set of problems.

2.1.1 Metaheuristics (MHs)

MHs are iterative improvement methods proposed to solve optimization problems. They are mostly considered iterative procedures and make extensive use of neighborhood search, perturbations, and simple heuristics over the current iteration solution (the incumbent) (HOOS; STÜTZLE, 2004). Besides, adding the possibility to accept worse solutions, they could escape from regions of local optima. Another advantage of MHs is that they usually offer a trade-off between solution quality and computational cost (i.e., better solutions are found increasing the number of iterations) (GRAHAM *et al.*, 1979; BAKER; TRIETSCH, 2009).

Some MHs can be part of a broader algorithm category called local-search methods. Local-search algorithms mainly (1) start from an initial solution, (2) iteratively move from current position to neighboring position and (3) use the objective function for guidance (HOOS; STÜTZLE, 2004). They can be expressed in four design decisions (PINEDO, 2016): (i) solution representation; (ii) neighborhood design; (iii) search process within the neighborhood; (iv) acceptance-rejection criterion.

Figure 2 – Types of neighborhood of permutation problems: transpose, exchange and insert.



Source: Adapted from Hoos and Stützle (2004).

In permutation problems, *Shift* operations form the insertion neighborhood of a permutation. A shift on (j,j') removes the job from position j and inserts it at position j'. In this way, a permutation has a total of $(J-1)^2$ neighbors. Other common neighborhoods are the transpose and exchange neighborhood, formed by swapping two jobs, adjacent or not. Figure 2 shows an example of transpose, exchange, and insertion neighborhoods. Other types embed structural

properties of flowshop exist in the literature (JIN *et al.*, 2009). Initialization, search process, and acceptance-rejection criterion are particular to each MH.

The Simulated Annealing (SA) metaheuristic was proposed by Kirkpatrick *et al.* (1983). Based on the annealing process in solids, Simulated Annealing (SA) starts the search with a flexible acceptance criterion that accepts most solutions (high temperature) and, as times goes on, the probability of accepting worse solutions is decreased (low temperature).

Tabu Search (TS) (GLOVER, 1989) is a very successful local-search MH for the flowshop problem. Tabu Search (TS) can avoid cycling through the same solutions in a local optima region by using a "tabu list" of old solutions that should not have their neighborhood explored. TS can restart the search until it reaches a limit number of iterations.

The Iterated Local Search (ILS) framework encompasses most local-search based MHs. The steps of ILS are (HOOS; STÜTZLE, 2004):

1. Define an initial solution and perform a local search
2. While a termination criterion is not satisfied:
    a) Perform a perturbation on the current solution
    b) Perform local search
    c) Use an acceptance criterion to decide if the new one replaces the current solution

It is clear that using the appropriate perturb and acceptance functions, and the search history, ILS can simulate SA and TS.

The basic ILS framework has inspired many other MHs. An MH similar to ILS is the Greedy Randomized Adaptive Search, that constructs a new solution with a greedy strategy and later applies a local search procedure (FEO; RESENDE, 1995; PRABHAHARAN *et al.*, 2006). Marmion *et al.* (2011a) propose Neutral Iterated Local Search (NILS) based on theoretical studies about neutral regions in the fitness landscape.

All MHs mentioned so far use a single candidate solution throughout the iterations. Population-based MHs simultaneously maintain several solutions or a population of them. Some advantages of these population-based MHs is enhanced exploration and diversification of the search space, as well as the capacity of combining several individual candidate solutions into another (HOOS; STÜTZLE, 2004; ENGELBRECHT, 2007). There are not many recent works using population-based MHs on flowshop problems and most are adapted to single-solution

versions to reduce the cost of maintaining a population (STÜTZLE, 1998a; RUIZ; MAROTO, 2006).

Another population-based MH widely used in combinatorial problems is the Ant Colony Optimisation (ACO) (DORIGO *et al.*, 1996). The inspiration for ACO is the trail-following behavior of real ants, where each ant constructs a path (solution), finding for food (local search) and updates the pheromones trails to reflect the collective search experience. Stützle (1998a) proposes the Min-Max Ant System (MMAS) for flowshop problem, using a single incumbent solution and local search.

2.2 FITNESS LANDSCAPE ANALYSIS

The FLA[1] consists in using metrics to define or estimate properties of the objective function and/or search operators, usually over the search space $\mathscr{S}$ of NP-hard problems. A solution $\mathbf{x} \in \mathscr{S}$ of size J is more commonly represented as a real vector on the real space $\mathscr{S} = \mathbb{R}^J$, a bit vector from the binary space $\mathscr{S} = \{0,1\}^J$ or a permutation from the permutation space $\mathscr{S} = (x_1, \ldots, x_J : x_a, x_b \in \{1, \ldots, J\} \wedge x_a \neq x_b, \forall a,b \in \{1, \ldots, J\})$. An objective function $g : \mathscr{S} \to \mathbb{R}$ maps each solution to a quantitative value to be minimized or maximized (STADLER, 2002; PITZER; AFFENZELLER, 2012).

Besides solution space and objective function, solution neighborhood relation is necessary to define a fitness landscape. A neighborhood relation is usually defined as $\eta : \mathscr{S} \to \wp(\mathscr{S})$, but in continuous optimization problems it could also encompasses solution distance notions ($dist : \mathscr{S} \times \mathscr{S} \to \mathbb{R}$) (PITZER; AFFENZELLER, 2012). In permutation problems like *flowshop*, the neighborhood can be defined using operators like *transpose*, *exchange* and *insert*, as shown in Figure 2.

By definition, local searches trace one or more paths on the search landscape, until meeting a stopping criterion. The search can be done from a single solution, following a single path, performing perturbations with kicks or restarts, using information from previous steps, or also exploring the space with many parallel solutions.

The fitness landscape can, therefore, be defined as a triplet $(\mathscr{S}, g, \eta)$. Many times the landscape is seen as a graph $G = (\mathscr{S}, E_\eta)$ where solutions are nodes and edges are neighborhood relationships: $(\mathbf{x}, \mathbf{x}') \in E_\eta$ if $\mathbf{x}' \in \eta(\mathbf{x})$. Usually, neighborhoods are defined such that G is a regular graph (every solution has the same number of neighbors), symmetric (if $\mathbf{x}$ is neighbor of

[1] Also known in the literature as: Search Space Analysis or Exploratory Landscape Analysis.

$\mathbf{x}'$, then $\mathbf{x}'$ is neighbor of $\mathbf{x}$) and connected (every solution is reachable from any other solution).

2.2.1 Simple fitness landscape metrics

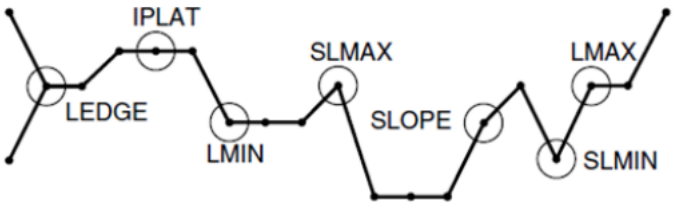
Some common fitness landscapes metrics use simple statistics over the neighborhood graph. For example, the distribution of node degrees and graph diameter. For the neighborhoods in Figure 2, the graph is regular, with a constant degree of $J-1$ for transpose and $(J-1)^2$ for exchange and insertion. The diameter, even on regular graphs, is not trivially calculated but can be approximated considering the search space size (HOOS; STÜTZLE, 2004). On the mentioned neighborhoods, the graph diameter is $J-1$ for exchange and insertion, because every solution can be turned into another solution with $J-1$ or less exchanges or insertions. The diameter for transpose neighborhood is not trivial, being $3, 6, 10, 15, 21$ for permutations of sizes $3, 4, 5, 6, 7$, respectively.

An interesting measure regarding search-based strategies is optimal solutions density. Considering a minimization problem $N_{opt} = |\{\mathbf{x} : f(\mathbf{x}) \geq f(\mathbf{x}'), \mathbf{x}, \mathbf{x}' \in \mathscr{S}\}|$ as the number of optimal solutions, and assuming that solutions are uniformly spaced, the mean number of steps a local search would take is $|\mathscr{S}|/N_{opt}$. Therefore, $N_{opt}/|\mathscr{S}|$ is the probability of finding the global optimum randomly. This measure is usually very small for large problems.

Other fitness landscape metrics come from counting the types of nodes and edges from the graph. Given a fitness landscape $(\mathscr{S},g,\eta)$ we can classify edges of a given node $\mathbf{x}$ as:

- $up(\mathbf{x})$: $|\{(\mathbf{x},\mathbf{x}') \in E_\eta : g(\mathbf{x}) < g(\mathbf{x}')\}|$
- $down(\mathbf{x})$: $|\{(\mathbf{x},\mathbf{x}') \in E_\eta : g(\mathbf{x}) > g(\mathbf{x}')\}|$
- $side(\mathbf{x})$: $|\{(\mathbf{x},\mathbf{x}') \in E_\eta : g(\mathbf{x}) = g(\mathbf{x}')\}|$.

Figure 3 – Types of solution nodes.



Source: Adapted from (HOOS; STÜTZLE, 2004)

The nodes or solutions can also be classified as (see Figure 3):

- Strict local minima (N_{SLMIN}): $down(\mathbf{x}) = side(\mathbf{x}) = 0$
- Local minima (N_{LMIN}): $down(\mathbf{x}) = 0 \wedge side(\mathbf{x}) > 0 \wedge up(\mathbf{x}) > 0$
- Plateau (N_{IPLAT}): $down(\mathbf{x}) = up(\mathbf{x}) = 0$
- Ledge (N_{LEDGE}): $down(\mathbf{x}) > 0 \wedge side(\mathbf{x}) > 0 \wedge up(\mathbf{x}) > 0$
- Slope (N_{SLOPE}): $down(\mathbf{x}) > 0 \wedge side(\mathbf{x}) = 0 \wedge up(\mathbf{x}) > 0$
- Local maxima (N_{LMAX}): $down(\mathbf{x}) > 0 \wedge side(\mathbf{x}) > 0 \wedge up(\mathbf{x}) = 0$
- Strict local maxima (N_{SLMAX}): $side(\mathbf{x}) = 0 \wedge down(\mathbf{x}) = 0$

These node types are a partition of $\mathscr{S}$. The percentage of each node and edge type is useful since it represents a measure independent of the landscape size. These percentages can be sampled for large instances, and the tendency with large neighborhoods (like insertion) is to find less strict node types like ledges and rarely nodes of type strict local minima/maxima.

2.2.2 Random-walk based fitness landscape measures

Random-walks simulate a non-informed search, where neighbors (or close solutions) are chosen randomly at each step. Typical fitness landscape measures analyze the fitness value along a random walk of length T ($g_1, \ldots, g_T$) using similar metrics used in time series analysis.

The autocorrelation of fitness values of close solutions is used to inform the space ruggedness. When the fitness values vary rapidly from close solutions, the space resembles a difficult problem with random surface and no information to guide the search. Given the distance between solutions $dist(\mathbf{x}, \mathbf{x}')$ as the number of edges between them, the autocorrelation with delay s is defined as (WEINBERGER, 1990):

$$\rho(s) = \frac{Cov[g(\mathbf{x}), g(\mathbf{x}')]_{\forall \mathbf{x},\mathbf{x}':d(\mathbf{x},\mathbf{x}')=s}}{\sigma[\boldsymbol{f}]^2} = \frac{E[g(\mathbf{x})g(\mathbf{x}')]_{\forall \mathbf{x},\mathbf{x}':dist(\mathbf{x},\mathbf{x}')=s} - E[\boldsymbol{f}]^2}{\sigma[\boldsymbol{f}]^2}.$$

where $Cov[\cdot,\cdot]$ is the covariance, $E[\boldsymbol{f}]$ and $\sigma[\boldsymbol{f}]$ are the mean and the standard deviation of all fitness values, respectively. For large instances this measure can be estimated using the fitness values $\boldsymbol{f}_r w = (g(\mathbf{x}_1), \ldots, g(\mathbf{x}_T))$ of a random walk over the graph as:

$$\hat{\rho}(s) = \frac{(1/(T-1)) \sum_{k=1}^{T-s} (f_k - E[\boldsymbol{f}_r w])(f_{k+s} - E[\boldsymbol{f}_r w])}{(1/T) \sum_{k=1}^{T} (f_k - E[\boldsymbol{f}_r w])^2}$$

The correlation between neighbor solutions $\rho(1)$ is the most common measure. In many problems, the autocorrelation is an exponential decaying function $\rho(1) = exp(-1/L_\rho)$ where L_ρ is the mean distance for which two solutions become uncorrelated. The measure $L_\rho = 1/ln(|\rho(1)|)$, called autocorrelation length, is also used to inform search space ruggedness.

In some problems, the objective function can be decomposed into a sum of elementary functions. This decomposition can then be used to find correlation formulas and more statistics algebraically (STADLER, 1996; CHICANO *et al.*, 2011).

Other types of measures over the fitness value of random walks are information-theoretic measures (VASSILEV *et al.*, 2000). These measures aim to provide different aspects of the landscape, such as the degree of regularity and diversity of local optima. The first step for information analysis is to transform the fitness sequence $(g_1, \ldots, g_T)$ into an ensemble of objects string $S(\epsilon) = s_2 \ldots s_m$ where

$$s_t = \begin{cases} -1 & \text{if } g_t - g_{t-1} < -\epsilon \\ 0 & \text{if } |g_t - g_{t-1}| \leq \epsilon \\ 1 & \text{if } g_t - g_{t-1} > \epsilon \end{cases}$$

where $\epsilon \geq 0$ is the parameter controlling the sensitivity of the metrics, such that high values of ϵ lead to very flat landscapes. The entropy metric can then be defined as:

$$H(\epsilon) = -\sum_{a \neq b} P_{[ab]} log_6 P_{[ab]}$$

where $P_{[ab]}$ is the probability of blocks $ab \in \{-1, 0, 1\} \times \{-1, 0, 1\}$ in the sequence $S(\epsilon) = s_1 \ldots s_m$, i.e., $P_{[ab]} = n_{[ab]}/T$ given $n_{[ab]}$ as the number of occurrences of ab in $S(\epsilon)$. Related to the entropy, the density basin of the sequence S measures the information over smooth regions:

$$h(\epsilon) = -\sum_{a \in \{-1,0,1\}} P_{[aa]} log_6 P_{[aa]}$$

Another shortened sequence of landscape elements can be build from $S(\epsilon)$ as $S'(\epsilon) = (s'_a : s'_a \neq 0 \wedge s'_i \neq s'_{i-1})$. The length $|S'(\epsilon)|$ of this new sequence gives the modality of the landscape walk and, when scaled by the original sequence length, gives the partial information content metric:

$$M(\epsilon) = \frac{|S'(\epsilon)|}{T}$$

Finally, the difference between the lowest and highest fitness value between neighbors is another metric called information stability $\epsilon*$. It is also the smallest ϵ such that the ensemble $S(\epsilon)$ is entirely flat.

2.2.3 Adaptive-walk based fitness landscape measures

Local search performance most likely depends on the location of local optima. On easy problems, the local optima are usually close together and lead to global optima. Fitness landscape metrics that consider the interval and distribution of distances between solutions to the local optimum can show correlation with the problem difficulty.

A high correlation between a solution fitness and the distance to the closest global optimum, the Fitnes Distance Correlation (FDC), indicates that the local search can be easily guided to good regions. Given $\boldsymbol{f}$ as the fitness values of each solution in $\mathscr{S}$ and $\boldsymbol{dist}$ the distances to the closest global optimum (according to some distance function) the FDC can be calculated as (JONES *et al.*, 1995):

$$\rho_{FDC,dist} = \frac{Cov[\boldsymbol{f}, \boldsymbol{dist}]}{\sigma[\boldsymbol{f}]\sigma[\boldsymbol{dist}]} = \frac{E[\boldsymbol{f} * \boldsymbol{dist}] - E[\boldsymbol{f}*]E[\boldsymbol{d}]}{\sigma[\boldsymbol{f}]\sigma[\boldsymbol{d}]}$$

When ρ_{FDC} is close to 1, the fitness is a good indicator of good solutions. If ρ_{FDC} is close to 0, most local search procedures become random searches. Otherwise if ρ_{FDC} is negative the problem is called “deceptive”.

Table 1 – Permutation distances.

Distance	Definition	Max
Absolute	$d_{abs}(\mathbf{x}, \mathbf{x}') = \sum_1^n k_i$, where $k_i = 1$ if $x_i = x'_i$, otherwise $k_i = 0$.	n
Adjacency	$d_{adj}(\mathbf{x}, \mathbf{x}') = \sum_1^n k_i$, where $k_i = 1$ if $x_i = x'_j$ and $x_{i+1} = x_{j+1}$, otherwise $k_i = 0$.	$n-1$
Precedence	$d_{pre}(\mathbf{x}, \mathbf{x}') = n(n-1)/2 - n_{pre}$, where n_{pre} is the number of times x_i is preceded by x_j in $\mathbf{x}$ and $\mathbf{x}'$.	$\frac{n(n-1)}{2}$
Deviation	$d_{dev}(\mathbf{x}, \mathbf{x}') = \sum \lvert\sigma_j - \sigma'_j\rvert$, where σ_j is the position of x_j in reversed $\mathbf{x}$.	$\lfloor n^2/2 \rfloor$
Approximated swap	$d_{swap}(\mathbf{x}, \mathbf{x}') = n_{swap}$, where n_{swap} is the number of swaps to transform $\mathbf{x}$ into $\mathbf{x}'$, approximated by iteratively swapping out of order jobs from left to right.	$n-1$
Shift	$d_{shift}(\mathbf{x}, \mathbf{x}') = n - LCS(\mathbf{x}, \mathbf{x}')$, where LCS is the length of the longest common sub-sequence of $\mathbf{x}$ and $\mathbf{x}'$.	$n-1$

Source: Adapted from Reeves (1999), Czogalla and Fink (2012)

For large problems, a set of fitness and distances can be sampled using a local search method. Usually, a Hill Climbing (HC) performing an adaptive walk can lead from a randomly

chosen solution to a close local optimum. The approximated FDC based on N_{lopt} local optima samples can then be computed as (JONES *et al.*, 1995):

$$\hat{\rho}_{FDC,dist} = \frac{Cov[\boldsymbol{f}, \boldsymbol{dist}]}{\sigma[\boldsymbol{f}]\sigma[\boldsymbol{dist}]} = \frac{1}{(N_{lopt}-1)\sigma[\boldsymbol{f}]\sigma[\boldsymbol{dist}]} \sum_{i=1}^{N_{lopt}} (g_i - E[\boldsymbol{f}])(dist_i - E[\boldsymbol{dist}]).$$

where $\boldsymbol{f}_a w$ is the fitness of N_{lopt} solution samples and $\boldsymbol{dist}$ their respective distances to the closest local optima. These local optima are found using a given local search procedure.

The distance metric used in FDC depends on the problem space. The flowshop problem studied has a permutation space, and there are several ways to measure the distance between two permutations (REEVES, 1999). The FDC experiments on Section 4.2 use six different distances defined in Table 1.

Other features of the landscape are also defined in the literature. Neutral networks are sub-graphs of the neighborhood where each solution has the same fitness as its neighbor. Features of this graph include the average and autocorrelation of neutral degrees (VEREL *et al.*, 2006). Basins of attraction are partitions of the search space where every solution in the basin arrives at the same local optima with an adaptive walk (PITZER; AFFENZELLER, 2012).

Moreover, a common hypothesis in FLA is the presence of a big valley structure, which states that local optima are relatively close to each other and the global optima. Several problems like the Traveling Salesperson Problem, quadratic assignment problem, capacitated vehicle routing problem and permutation flowshop are known to have a big valley structure.

2.3 MULTI-LABEL LEARNING

This work also deals with some machine learning concepts like multi-label classification and feature selection. In multi-label learning, each input from feature space F is associated with multiple output labels from a label space $\mathbb{O}$. Given an example set $\{(\boldsymbol{\phi}, y_i) : \boldsymbol{\phi} \in F, y_i \in \mathbb{O}\}$, the goal of multi-label classification is to find a function $h : F \to \wp(\mathbb{O})$ that maximizes a performance measure. Likewise, in multi-label ranking, a model learns $h_{ML} : F \times \mathbb{O} \to \mathbb{S}$ where $\mathbb{S}$ is the ranking space of each pair of input and label.

There are several strategies for multi-label classification and ranking (TSOUMAKAS *et al.*, 2009; MADJAROV *et al.*, 2012; MONTAÑES *et al.*, 2014; GIBAJA; VENTURA, 2015; RIVOLLI; CARVALHO, 2018). The methods are usually classified as algorithm transformation and problem transformation. In algorithm transformation, a special algorithm is proposed, or a

single-label one is adapted to deal with multi-label data. In this work, we investigate problem transformation, where the multi-label problem is transformed, allowing it to be solved with any traditional single-label model. According to the descriptions provided in Rivolli and Carvalho (2018), 17 strategies are investigated and those can be classified into four main categories:

- **Binary relevance methods**. Binary Relevance (BR) transforms the problem into one binary problem for each label. Therefore, BR assumes that the labels are independent. The Ensemble of Binary Relevance (EBR) uses multiple models and a simple voting scheme for prediction. Ensemble of Single Label (ESL) focuses models on the least frequent labels.

- **Label powerset methods**. Label Powerset (LP) creates one class for each combination of output labels, transforming the problem into a single-label multi-class problem. RAndom k-labELsets (RAKEL) builds and ensamble of LP classifiers and avoids problems with unseen class combinations. Hierarchy Of Multi-label classifiER (HOMER) decomposes the problem into an efficient hierarchy of multi-label problems (recursively solved by inner multi-label algorithms) reducing the complexity of LP for problems with many labels.

- **Pairwise methods**. Ranking by Pairwise Comparison (RPC) transforms the k-label problem into $(k-1)k/2$ binary problems, for each pair of labels. During the testing, the labels are chosen by counting the votes for each comparison. Calibrated Label Ranking (CLR) extends RPC by using an artificial label to indicate the split between the output labels.

- **Transformations for Identifying Label Dependencies.** Classifier Chain (CC) method for a k-label problem uses k chained classifiers, where each classifier includes the previous labels as part of its feature set. Ensemble of Classifier Chains (ECC) uses an ensemble of multiple random orders of labels where the output is decided by a voting scheme.

  Nested Stacking (NS), is similar to CC, but the classifiers in the chain use the predicted label instead of the labels known during the training step. Another method that overcomes the BR assumption of label independence is Meta Binary Relevance (MBR). In MBR, the BR procedure is done twice, first as the regular BR and second as a meta-learning model taking the outputs of the previous BR models as inputs like in a stacking ensemble.

  Dependent Binary Relevance (DBR), and Binary Relevance Plus (BRPLUS) are methods that include all other labels as inputs in order to provide conditional dependency informa-

tion. ConTRolled Label correlation exploitation (CTRL) applies feature selection methods to identify label dependencies. Pruned and Confident Stacking Approach (PRUDENT) iterates the stack of classifiers, eliminating features with low information gain.

There are several multi-label performance metrics, mostly are an adaptation of known metrics used in machine learning like accuracy, precision, recall, F_1, and others. Specifically, multi-label performance can be evaluated in the micro-, macro- and example-level, where the prediction quality is averaged over the individual outputs, labels and examples, respectively. Besides those, other metrics based on ranking are used to evaluate multi-label models like Hamming and ranking loss.

In this work, we use macro-AUC and average-precision, as those are known to show complementary behavior (WU; ZHOU, 2017). The macro-AUC is the average Area Under the receiver operating characteristic Curve (AUC) for each label (in our case, each recommended MH). The average precision is the mean value of the fraction of labels ranked above each particular label for all test samples (as recommended MHs are above the statistically worse non-recommended MHs). In both cases, higher values indicate better performance.

2.4 MONO AND MULTI-OBJECTIVE OPTIMIZATION

An unrestricted optimization problem with a single objective function can be formulated as minimization problem:

$$\begin{aligned} &\min g(\mathbf{x}) \\ &\text{subject to } \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{aligned} \tag{1}$$

where: $\mathbf{x} = (x_1, \ldots, x_n)$ is a solution vector of n decision variables between upper and lower bounds, $\mathbf{x}_L$ and $\mathbf{x}_U$ respectively, and g is the objective function to be optimized.

An optimization problem is multi-objective when there are two or more objective functions. Several engineering problems are modeled as multi-objective ones. An unrestricted multi-objective problem can be formulated as (DEB, 2011):

$$\begin{aligned} &\min \mathbf{g}(\mathbf{x}) = \left\{g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_{N_{obj}}(\mathbf{x})\right\} \\ &\text{subject to } \mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U \end{aligned} \tag{2}$$

where: $\mathbf{x} = (x_1,...,x_n)$ is a solution vector, and $\mathbf{g}$ is a vector of N_{obj} objective functions to be optimized.

On multi-objective problems, a solution is compared with another based on a dominance relation. A solution $\mathbf{x}_a$ *dominates* $\mathbf{x}_b$ if $\mathbf{x}_a$ is, for at least one objective, strictly better than $\mathbf{x}_b$. More precisely (DEB, 2011):

$$\begin{aligned} \mathbf{x}_a \text{ dominates } \mathbf{x}_b \rightarrow & \forall i \in \{1,\ldots,N_{obj}\}, f_i(\mathbf{x}_a) \leq f_i(\mathbf{x}_b) \text{ and} \\ & \exists k \in \{1,\ldots,N_{obj}\} : f_k(\mathbf{x}_a) < f_k(\mathbf{x}_b). \end{aligned} \tag{3}$$

The set of all non-dominated solution in the decision space is called a Pareto set and the corresponding image on the objective space is called a Pareto optimal front, Pareto front for short.

2.4.1 MOEA/D: Multi-objective evolutionary algorithm based on decomposition

As multi-objective problems are often unfeasible for solving using exact approaches, the Pareto front is approximated by Multi-objective evolutionary algorithms (MOEAs). Decomposition is a successful technique used by most state-of-the-art MOEAs. It uses weights to decompose the problem into several mono-objective problems and aggregation functions to select the most fitted solutions.

The algorithm in Figure 4 describes the original MOEA/D, where first, the population of N_{pop} solutions is initialized randomly.

Figure 4 – MOEA/D algorithm.

```
 1: procedure MOEAD(g, N_pop, N_gen, N_neigh)
 2:     P ← RandomPopulation(N_pop)
 3:     W ← UniformWeights(N_pop)
 4:     B ← AssignNeighbors(P, W)
 5:     while stop condition is not achieved do
 6:         for x_i ∈ P do
 7:             x'_i ← GeneticOperators(x_i, B_i, w_i)
 8:             z* ← min{z*, g(x'_i)}
 9:             for x_neigh ∈ B_i do
10:                 if g(x'|w_i,z*) ≤ g(x_neigh|w_i,z*) then
11:                     x_neigh ← x'
12:                 end if
13:             end for
14:         end for
15:     end whilereturn All non-dominated solutions from P
16: end procedure
```

Source: Adapted from Zhang and Li (2007)

Each individual is associated with a weight $\mathbf{w}_i$ from a weight set W, where $\mathbf{w}_i$ consists on a vector of N_{obj} numbers that sum to 1 and are distributed uniformly in the objective space.

The corresponding solutions for the closest N_{neigh} weights form a neighborhood set (B_i) for each solution. For N_{gen} generations, each solution in the population is modified by a variation operator and the child solution is evaluated. After that, an aggregation function $g(\cdot)$ value is used to compare and update the solutions from the neighborhood B_i. At the end, the algorithm returns the non-dominated solutions from the population.

The original MOEA/D uses simulated binary crossover and polynomial mutation as variation and Tchebycheff aggregation function, given by:

$$g^{tche}(\mathbf{y}|W_i,\mathbf{z}) = \max_{1\leq i\leq N_{obj}} \{w_i|y_i - z_i^*|\} \tag{4}$$

where $\mathbf{z}^*$ is the current reference point, composed of the best value of each objective function.

2.5 FLOWSHOP PROBLEMS

Flowshop is a combinatorial optimization problem of sequencing or scheduling. The problem involves deciding how J jobs will be processed on M machines in series, given their processing times. A permutation $\mathbf{x} = (x_0, \ldots, x_J)$ informs the order jobs will be executed on each machine. Besides, a *schedule* shows the system state at any time, allowing job preemption, dynamic release times, and out of order execution. Given that, the most common flowshop formulations look for a sequence of jobs that maximizes a performance measure and obeys given constraints (PINEDO, 2016).

This section presents a review of the literature of methods to solve the permutation flowshop problem. It covers a broad set of methods ranging from classic exact methods like Johnson's algorithm (JOHNSON, 1954) to recent state-of-the-art MHs like Iterated Greedy (RUIZ; STÜTZLE, 2007).

Scheduling problems are a broad topic with several works in the literature. Graham *et al.* (1979), Baker and Trietsch (2009), Pinedo (2016) provide theoretical results and formulations. The review made by Potts and Strusevich (2009) focuses on historical milestones in the field. More recently, Ruiz and Maroto (2005), Fernandez-Viagas *et al.* (2017) review and compare several flowshop heuristics and MHs.

According to Baker and Trietsch (2009), the usual permutation flowshop formulation has the following conditions:

1. a set of J unrelated, multiple-operation jobs is available for processing at time zero;

2. each job requires M operations, and each operation requires a different machine;
3. setup times for the operations are sequence-independent and included in processing times;
4. job times on each machine are known in advance;
5. all machines are continuously available;
6. once the operation begins, it proceeds without interruption.

Besides, a flowshop problem instance for a set of J jobs and M machines is given by three pieces of information:

Processing times ($p_{j,m}$): the amount of processing required by job j in machine m;

Due time (due_j): the time that each job j is due to be completed (required only when the objective involve tardiness or earliness measures);

Release time (rel_j): the time job j is available for processing (usually $rel_j = 0$ according to condition 1).

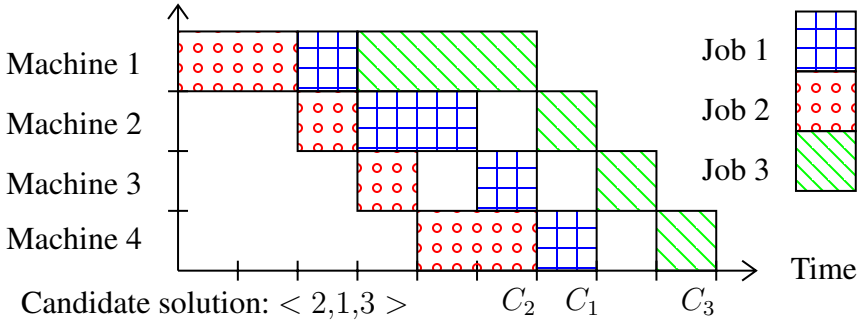
A sequence is a permutation of jobs $\mathbf{x} = (x_1, x_2, \ldots, x_J)$. Given these conditions, each machine can process an available job immediately when it is available. In other words, machine 1 starts processing the first job in the sequence at time zero and m-th machine can start processing job x_k as soon as job x_{k-1} finishes and job x_k is done on machine $m-1$. Formally, the completion time of job x_k on machine m is given by:

$$\begin{aligned}
&C_{x_1,m} = \textstyle\sum_{l=1}^{m} p_{x_1,l} && m = 1,\ldots,M \\
&C_{x_k,1} = \textstyle\sum_{l=1}^{k} p_{x_k,1} && k = 1,\ldots,J \\
&C_{x_k,m} = \max(C_{x_k,m}, C_{x_{k-1},m}) + p_{x_k,m} && m = 2,\ldots,M; k = 2,\ldots,J
\end{aligned}$$

Figure 5 shows an example with $J = 3$ jobs in an $M = 4$ machine shop and a job schedule $\mathbf{x} = (2,1,3)$. The processing times are $p_{1,1} = p_{1,4} = 1$ and $p_{1,2} = p_{1,3} = 4$ for the first job; $p_{2,1} = p_{2,4} = 4$ and $p_{2,2} = p_{2,3} = 1$ for the second job; $p_{3,1} = 3$; and $p_{3,2} = p3{,}3 = p3{,}4 = 1$ for the third job. Due to flowshop restrictions, there are delays for example between jobs 1 and 3 on machine 2.

Given a flowshop instance, usually, the objective is to find the best permutation of jobs with regard to an objective function. Having defined some measurements:

Figure 5 – Permutation flowshop example.



Source: The author.

Flowtime ($F_j = C_{j,M}$): total processing time of job j;

Tardiness ($T_j = \max\{0, C_{j,M} - due_j\}$): delay time on job j;

Earliness ($E_j = \max\{0, due_i - C_{j,M}\}$): time job j is early with respect to its due date.

The most common objective functions are:

Maximum completion time or Makespan: $C_{max} = \max\{C_{j,M}, j = 1, \ldots J\}$;

Total flowtime: $F = \sum_{j=1}^{J} F_j$;

Maximum tardiness: $T_{max} = \max\{T_j, j = 1, \ldots J\}$;

Number of tardy jobs: $U = \sum_{j=1}^{J}\{1 \text{ if } T_j > 0 \text{ else } 0\})$;

Weighted earliness and tardiness: $U_{we,wt} = \sum_{j=1}^{J}\{we_j E_j + wt_j T_j\}$ (for some given earliness and tardiness costs, we_j and wt_j respectively);

It is worth noting that, with some exceptions, non-permutation schedules (different job orders between machines) can be better than permutations (BAKER; TRIETSCH, 2009). Nonetheless, permutations provide a simple solution structure and a search space of $J!$ instead of $J!^M$ for non-permutation flowshops. In this work we only consider permutation flowshops.

Some flowshop formulations add new constraints to better model real-world scenarios. The no-wait flowshop variant includes the requirement that the products can not be stored between operations (BAKER; TRIETSCH, 2009). Storages or buffers are considered in a variant called limited buffers flowshop (EMMONS, 2013). In no-idle flowshops, machines should not have idle time between job exchanges (MARIUSZ, 2015) as is common in steel production.

Storing, transportation, break-down times, and weights of jobs can also be possible constraints (BONNEY; GUNDRY, 1976).

Graham *et al.* (1979) provide a survey and convenient notation for the flowshop and related scheduling problems. In the single-machine model, where $M = 1$, specific job sorting methods and dispatch rules can be efficiently used. Parallel machine formulations cover the cases where $M = 1$, but more than one machine is available to execute a job operation at the same time. On the other hand, hybrid flowshops allow multi-stage parallel machines (LINN; ZHANG, 1999), also, in job shop scheduling problems the machine order is specified for each job (BAKER; TRIETSCH, 2009; MASTROLILLI; SVENSSON, 2011).

Other flowshop variants relax the mentioned conditions. For example, the flowshop with setup times relax condition 3, allowing sequence-dependent setup times (RUIZ; MAROTO, 2006; ALLAHVERDI *et al.*, 2008; BAKER; TRIETSCH, 2009). A relaxation on condition 4, with processing times drawn from a random distribution, is known as the stochastic flowshop (PINEDO; SCHRAGE, 1982; BAKER; TRIETSCH, 2009). Also, in some formulations, the processing times can depend on time (PINEDO; SCHRAGE, 1982), and model the learning effects (BISKUP, 2008). Job preemption and unknown release times relax conditions 1 and 6 and can affect the makespan (BAKER; TRIETSCH, 2009).

Any combination of constraints and objectives can have different methods proposed to solve it. For example, in the $M = 2$ machine flowshop, the optimal makespan sequence is easily found. However, allowing stochastic processing and sequence-dependent setup times, the problem can be hard to solve (BAKER; TRIETSCH, 2009). Alternatively, multi-objective flowshop formulations adopt two or more trade-off objectives to be simultaneously optimized (YENISEY; YAGMAHAN, 2014). However, most of the literature focuses on the permutation flowshop and makespan objective.

There are many algorithms proposed to solve flowshop problems. Exact methods include the classic Johnson's algorithm (JOHNSON, 1954) for two machines. The general case can be modeled as linear programming problems (WAGNER, 1959; MANNE, 1960; TSENG *et al.*, 2004) or branch-and-bound methods (LAGEWEG *et al.*, 1978; MELAB *et al.*, 2012). In spite of being proposed many time ago, these classic methods are still in use. Recent works are proposed, for example, mixed integer programming and constraint programming models for different flowshop variants (ÖZGÜVEN *et al.*, 2010; MALECK *et al.*, 2018).

Although exact methods are optimal, they may not be computationally efficient in

instances with many jobs. The general flowshop problem with makespan minimization for $M = 3$ or more machines is proved to be NP-Hard (GAREY *et al.*, 1976). It means that there is no efficient way to solve it (if P $\neq$ NP). Therefore, it is common in practice to use heuristics yielding solutions that are close enough to optimal.

Well known flowshop heuristics include: Palmer (1965) proposes a constructive heuristic, Campbell-Dudek-Smith heuristic (CAMPBELL *et al.*, 1970), Rapid Access (DANNENBRING, 1977), and Heuristic for Flowshop Scheduling (KOULAMAS, 1998).

The NEH heuristic, proposed by Nawaz *et al.* (1983), is perhaps the most popular flowshop heuristic. It is a constructive heuristic that works by inserting jobs in partial sequences. The following steps can describe the NEH algorithm:

1. **Initialization:** calculate the sum of processing times of each job, and sort them by decreasing order of that sum;
2. For $i = 2, \ldots, J$ do:
    a) **Insertion:** Insert i-th job in every position in the partial sequence $1, \ldots, i-1$
    b) Select the best partial sequence and continue

This procedure has a complexity of $O(J^3M)$. Taillard (1990) later improved NEH's complexity to $O(J^2M)$ because it is easy to recalculate the makespan from each insertion, given the previous partial sequences completion times. With extensive tests, NEH is considered the best heuristic for flowshop (RUIZ; MAROTO, 2005). Thus, many recent heuristics and MHs proposals internally use NEH or some variant of NEH. These variants improve aspects like the initial order (DONG *et al.*, 2008; RIBAS *et al.*, 2010; NAGANO; MOCCELLIN, 2002; KALCZYNSKI; KAMBUROWSKI, 2008) and introduce tie-breaking mechanisms (KALCZYNSKI; KAMBUROWSKI, 2007; KALCZYNSKI; KAMBUROWSKI, 2008; KALCZYNSKI; KAMBUROWSKI, 2009; DONG *et al.*, 2008; Fernandez-Viagas; FRAMINAN, 2014).

2.5.1 Meta-heuristics to solve Flowshop problems

Osman and Potts (1989) propose a SA cooling schedule for flowshop problems with ordered and random neighborhoods (order of neighbors change withing iterations). Later, Ishibuchi *et al.* (1995) propose a SA with a dynamic cooling schedule, that adapts the frequency of temperature updates. Zegordi *et al.* (1995) incorporate flowshop specific knowledge to SA, using a table

with the “move desirability for jobs” during the annealing process. Also, recent work presented by Chen and Sandnes (2015) propose SA with an internal local search based in job insertion.

Widmer and Hertz (1989) propose a TS variant for the flowshop problem named SPIRIT. The proposal uses a constructive heuristic based on a travelling salesperson problem formulation and job insertion, followed by TS with exchange neighborhood. Reeves (1993) further improves SPIRIT with NEH initialization. Nowicki and Smutnicki (1996) improve the basic TS with blocks of jobs, a fast TS implementation that does not explore all solutions in potentially big neighborhoods. Ben-Daya and Al-Fawzan (1998) propose another TS variant considering the intensification versus diversification dilemma. Moccellin and Santos (2000) hybridize TS and SA.

Stützle (1998b) applies the standard ILS with NEH heuristic initialization, the common types of neighborhoods and Metropolis-Hastings acceptance criterion (i.e. the same used by SA metaheuristic) to the permutation flowshop problem.

Ruiz and Stützle (2007) proposes a very successful ILS variant called Iterated Greedy (IG). IG main feature is deconstruction/construction perturbation function based the NEH heuristic. The perturbation locks some jobs positions (destruction) and allows inserting the others in the best possible position (construction). Some variants of IG are proposed, like Fernandez-Viagas and Framinan (2014), that integrate NEH tie-breaking mechanisms to IG, and Dubois-Lacoste *et al.* (2017) that proposes using local-search between the construction and destruction perturbations, namely IG with Local Search on Partial Solution (LSPS).

Rajendran and Ziegler (2004) enhance MMAS construction and local search phases. Also, Tzeng *et al.* (2012) make use of estimation of distribution algorithms to improve ACO performance on the flowshop problem.

Several other population-based MHs are proposed to solve the flowshop problem: genetic algorithms (REEVES, 1995; REEVES; YAMADA, 1998; WANG; ZHENG, 2003; RUIZ; MAROTO, 2006), differential evolution (PAN *et al.*, 2008) and particle swarm optimization (LIU *et al.*, 2011). In this work we mostly focus on stochastic local search metaheuristics based on a single incumbent solution, as those methods show good performance in comparison to population based metaheuristics (RUIZ; MAROTO, 2005; RUIZ; STÜTZLE, 2007).

2.5.2 Flowshop fitness landscape analysis

There are several FLA results on flowshop problems in the literature. One of the first works was presented by Reeves (1999). Reeves investigates local minima distribution, FDCs with different distances and concludes with evidence that flowshop problems do contain big basins of attraction.

Marmion *et al.* (2011a) investigate neutrality on flowshop problems. Neutral regions are characterized by studying the neutral degree during a random walk on flowshop landscapes, concluding that the neutral degree is not random and they are mostly present on instances with processing time correlation. Marmion *et al.* (2011b) apply the previous analysis for the design of a new ILS variant (Neutral ILS).

Czogalla and Fink (2012) analyze the no-wait flowshop variant using node type frequencies and FDC with seven different permutation distances. None of the distances proved to be superior. Therefore it is interesting to use different operators and hybrid MHs on the no-wait flowshop problem.

Hernando *et al.* (2017) perform a fitness landscape analysis with Local Optima Networks (LONs) on permutation flowshop with makespan and flowtime objectives. They measure several features like funnel size, frequency of sink nodes, and page rank. They also propose a compressed LON where nodes are grouped when they have the same fitness and called “neutral nodes”. Given that, they were able to justify why the flowtime objective is usually harder to solve than the makespan objective.

2.6 RELATED WORKS

Analyses of flowshop features effects on MHs performance are present in FLA works (REEVES, 1999; CZOGALLA; FINK, 2012; MARMION *et al.*, 2011a; MARMION; Regnier-Coudert, 2015; HERNANDO *et al.*, 2017). Many concepts of these works contribute to our proposal, but they focus on studying the main effects of different instance features, solution representation, and neighborhood operator on flowshop search landscapes.

Also related to our proposal, there are many meta-learning frameworks for combinatorial optimization problems. One could cite proposals for quadratic assignment problem (SMITH-MILES, 2008; DANTAS; POZO, 2018), graph coloring (SMITH-MILES *et al.*, 2013), travelling salesperson problem (KANDA *et al.*, 2011; KANDA *et al.*, 2012; KANDA *et al.*, 2016), travelling

thief problem (WAGNER *et al.*, 2018), vehicle routing problem (Gutierrez-Rodríguez *et al.*, 2019), statisfiability problem (XU *et al.*, 2008) and more. Compared to our study, the main differences are the focus on PIAC strategy and flowshop scheduling problems which are the focus of our proposal.

Table 2 – Related works.

Base problem	Meta-learning Strategy	Reference
Job-sorting heuristic recommendation for the Early/Tardy scheduling problem	Neural networks, decision trees, and self-organizing maps	(SMITH-MILES *et al.*, 2009)
Predicting the cost and local search dynamics of tabu search for the job shop problem	Linear regression and Markov models	(WATSON, 2010)
Parameter recommendation for tabu search in set covering, Cplex in mixed integer-programming and stochastic local search for satisfability problems	Adaptive K-Nearest Neighbors algorithm	(KADIOGLU *et al.*, 2010)
Performance prediction and algorithm recommendation on six types of combinatorial optimization problem	Generic meta-learning formulations using Rice's framework	(SMITH-MILES; LOPES, 2012)
Parameter recommendation support vector machines and random forest in classification problems	K-nearest neighbors in the feature space are used to warm initialize a bayesian parameter tuner	(FEURER *et al.*, 2015)

Source: The author.

Some closely related works about meta-learning for scheduling problems and parameter recommendation are shown in Table 2. As we can see, different from our proposal, none of the papers directly addresses flowshop problems, and most PIAC strategies use simple models like k-nearest neighbors on the feature space.

There are several works in the literature that address the AAC on different base problems. For example, on flowshop problems using racing and grammar-based evolutionary algorithms (MARMION *et al.*, 2013; MASCIA *et al.*, 2014a; MASCIA *et al.*, 2014b; PAGNOZZI; STÜTZLE, 2017). Multi-objective approaches to the AAC problem are much more scarce.

One way to model AAC as a multi-objective problem is to consider *performance fronts* (DRÉO, 2009). This approach considers two conflicting criteria: the performance on the base problem and the computational cost. The formulation allows solving continuous and combinatorial problems with multiobjective algorithms like Non-dominated Sorting Genetic Algorithm (NSGA-II).

Other proposals (DYMOND *et al.*, 2013; DYMOND *et al.*, 2015; DYMOND *et al.*, 2017) expand this concept by using the search history to evaluate several budgets at once. For the authors, the performance at the beginning, middle or end of the evolutionary process is different due to the budget associated with the number of generations and must be compared in a

multi-criteria way. For example, a solution can be chosen because it provides a good performance earlier than another one with better performance but achieved at the end of evolution. Such approaches can also handle multiple utilities simultaneously and use resampling mechanisms to deal with noisy objective function of stochastic algorithms.

Another example of MOAAC/D is the adaptation of ParamILS, a known single-objective AAC, for handling multi-objective tuning (BLOT *et al.*, 2016). For that, ParamILS is augmented with an archive of non-dominated solutions. The archive is used in the perturbation mechanism, by choosing a random non-dominated configuration, and also on local search uses the dominance relation to explore the neighborhood of the current configuration. Multi-objective ParamILS shows good results for tuning mixed-integer optimization algorithm with bi-objective formulations like solution quality and cutoff time.

The multi-objective model racing SPRINT (ZHANG *et al.*, 2013; ZHANG *et al.*, 2015) extends the dominance concept with statistical inference and shows good results on an ant colony algorithm bi-objective tuning of travelling salesperson tour length and computation time. Also, combining different metrics with multi-objective indicators like hypervolume is useful for selecting metaheuristics for problems like travelling salesperson's penalized average runtime and penalized quantile runtime (BOSSEK *et al.*, 2020).

Despite the similar end goal of AAC, in our proposals, the main focus is the instance-based recommendation of MHs and configurations, or PIAC, based on knowledge of algorithm configurations performance. Compared to other multi-objective AAC approaches, our approach is different in the sense it considers different instances of the problem for each objective and proposes a well-suited strategy to improve compromise solutions in this space with MOEA/D with (optional) irace operator for automatic algorithm configuration (iMOEA/D).

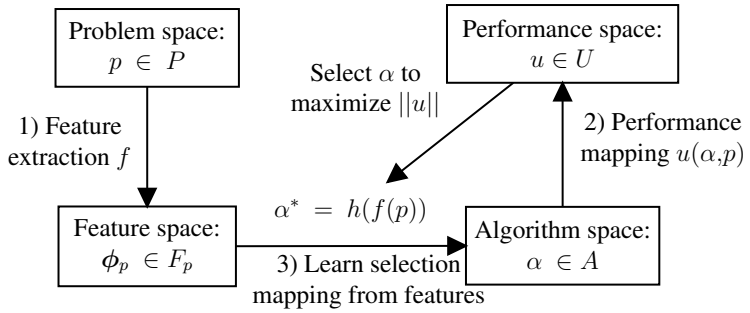
3 PER-INSTANCE ALGORITHM CONFIGURATION

In this chapter, as one of the contributions of our work, we present in Section 3.1 the PIAC problem under the perspective of Rice's framework (RICE, 1976). Then we describe the two approaches (Sections 3.2 and 3.3) proposed to solve the formulated problem.

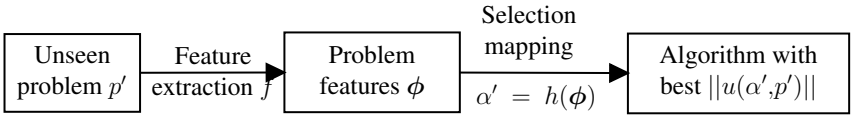
3.1 THE PIAC PROBLEM AND RICE'S FRAMEWORK

Rice (1976) provides the basic notation and a framework for the algorithm recommendation problem. Rice's framework, as seen in Figure 6, models several meta-learning systems in the literature. Nevertheless, in this work, it serves as the base for both of our proposed approaches.

Figure 6 – Building and using algorithm recommendation models with Rice's framework.



(a) Building algorithm recommendation model (training phase).



(b) Using the recommendation model (prediction phase).

Source: Adapted from Rice (1976).

Given a space of problems $p \in P$, a space of features $f(p) \in F$, each one extracted from the problems, a space of algorithms $\alpha \in A$, a performance metric or utility $||u(\alpha,p)|| \in U$, the main task involves learning the mapping h (selection mapping or recommendation function) from problem features $f(p)$ to the algorithm that maximizes $||u(\alpha,p)||$. After building the model that provides h, one can use it to predict the best algorithm, based on the instance features of a new problem (Figure 6).

When designing recommendation algorithms, three main phases are feature extraction, performance mapping, and recommendation model building (Figure 6). The feature selection procedure extracts the meta-features from the problem instances. In the performance mapping,

the main concern is to build knowledge about how good each algorithm in the portfolio is for each problem instance (SMITH-MILES; LOPES, 2012; LEMKE *et al.*, 2015). Finally, with problem features and performance at hand, the recommendation model can be built.

There could be many algorithms ($\alpha \in A$) that yield the (statistically) same performance. Besides, each algorithm could contain a set of parameters that affect its performance. Then, recommending a proper configuration considering instance's meta-features and past experience, like performed in the classic PIAC, is mainly an open problem (KERSCHKE *et al.*, 2019). In the PIAC formulation proposed here, a typical algorithm α contains a set of parameters Π_α. A parameter $\pi \in \Pi_\alpha$ controls some aspect(s) of algorithm α and it can assume a value θ. The set of all parameters' values is called configuration $\boldsymbol{\theta}$ vector, drawn from a configuration space Θ_α. Therefore, in the present work the whole space $A + \Theta$ encompasses not only the algorithms but also their configurations. A function given by $u : \Theta_\alpha \rightarrow \mathbb{R}$ defines a utility value to be optimized for a configuration. The utility represents the quality or cost associated with the configuration, to be maximized or minimized.

To provide the PIAC formalization, first we consider u regarding the configuration cost, and the AAC formulation as the following minimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} u(\boldsymbol{\theta}|\Theta_\alpha^f, P, C_{\boldsymbol{\theta}}, t) \tag{5}$$

where u depends on $\Theta_\alpha^f \subseteq \Theta_\alpha$, i.e., the space of feasible configurations for algorithm α; the problem space P, usually composed by a set of instances $\{inst\}$; the distribution of costs $C_{\boldsymbol{\theta}}$ for a particular configuration $\boldsymbol{\theta}$; and the function $t : P \rightarrow \mathbb{R}$, associating the computation time allocated for each problem.

The estimation of the utility of a configuration $\boldsymbol{\theta} \in \Theta_\alpha^f$, for a particular algorithm α, on a specific problem p, is based on the expected value of the cost distribution as:

$$u(\boldsymbol{\theta}) = \mathrm{E}[C_{\boldsymbol{\theta}}]. \tag{6}$$

$C_{\boldsymbol{\theta}}$ can be sampled by running α configured by $\boldsymbol{\theta}$ with different seeds s, $s = 1,..,S$, with each sample given by $c_{s,\boldsymbol{\theta}} = g(\mathbf{x}_{s,\boldsymbol{\theta},p})$, i.e. the sampled cost is obtained from the objective function (fitness) of the best solution $\mathbf{x}_s$ achieved at the end of evolution, using the seed s to generate the initial population of solutions for the base problem $p \in P$.

In most cases, the configuration $\boldsymbol{\theta}$ guides the algorithm α running on a sample of instances $\{inst\}_{sample} \subset P$. The performance/cost is measured on time limited to $t(inst)$,

and the mean cost of the sample runs, each one with a different seed s in the random number generator, accesses the configuration utility u (BIRATTARI, 2009).

Assuming a given algorithm α and F as the problem features space, the PIAC problem generalizes AAC by finding the mapping $h : F \rightarrow \Theta_\alpha$ that optimizes u:

$$\begin{aligned} h^* &= \arg\min_{h:F\rightarrow\Theta_\alpha} ||\mathbf{u}_h|| \\ \mathbf{u}_h &= (u(h(f(p))), \text{ for } p \in P) \end{aligned} \tag{7}$$

where $f : P \rightarrow F$ is a function that extracts relevant features from the problem space, and $||\mathbf{u}_h||$ is the norm of the mapping h performance over all the problems in the problem space P.

By solving the PIAC problem, one can use the mapping h, usually the one provided by a machine learning model, to find best fit configurations for unseen instances only by computing their feature vectors. The challenges here are (i) designing a good feature extractor, (ii) building a performance dataset consisting of pairs of feature and utility for each configuration $\boldsymbol{\theta}$, i.e., $(f(p)$,$u(\boldsymbol{\theta}|P = \{inst\}))$, and (iii) training the model to recommend configurations.

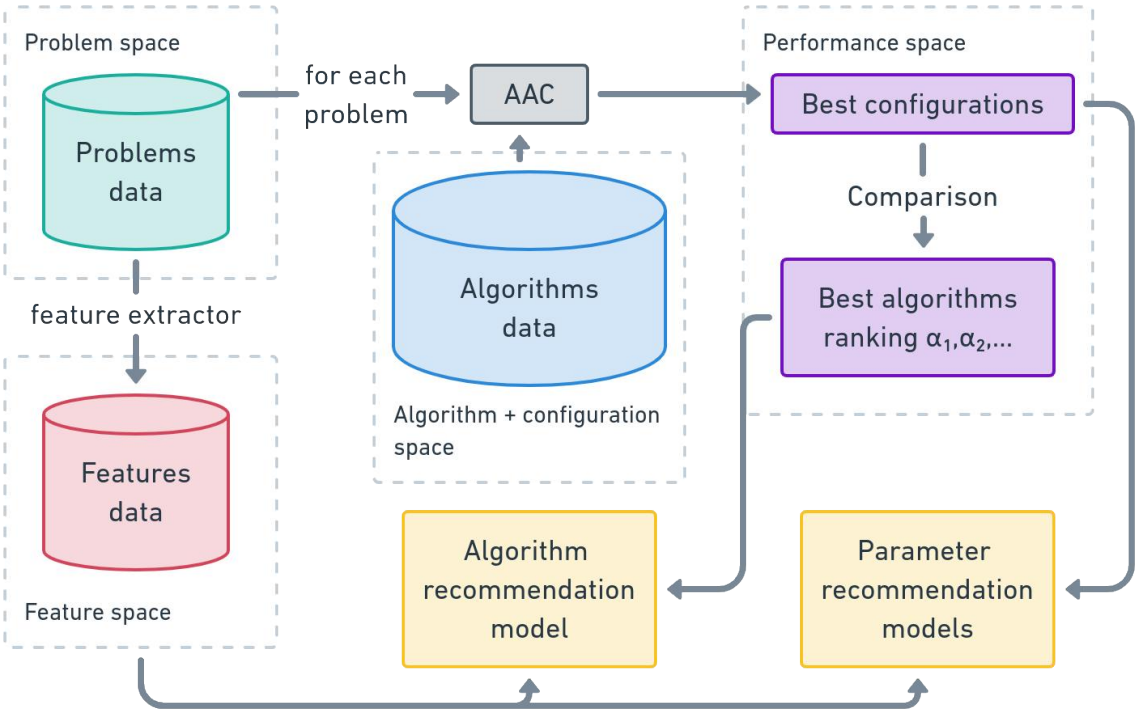
In the next sections, we detail the two proposed approaches: (i) one, based on meta-learning, recommends not only the algorithm (MH) in the optimization task but also its configuration parameters; (ii) the other, considered a completely new approach, addresses a multi-objective AAC based on problem space decomposition. In both cases, the approaches are presented as generalizations of AAC for solving the PIAC problem.

3.2 METAL PIAC: A PROPOSAL BASED ON META-LEARNING

As several examples in the literature, the basic Rice's framework can be expanded to accommodate different systems (SMITH-MILES, 2009). This work proposes and investigates a framework, Meta-Learning based Per-Instance Algorithm Configuration (MetaL PIAC), considering parameters as part of the algorithm space, as well as different performance mappings like multi-label and ranking algorithms recommendations. An overview of Meta-Learning based Per-Instance Algorithm Configuration (MetaL PIAC) can be seen in Figure 7.

As in the case of Rice's framework, MetaL PIAC works on algorithm, performance, problem, and feature spaces. The main differences are: (i) it uses an internal AAC tuner for building the meta-data and (ii) it considers extra information on best configurations for the parameter recommendation phase. In the performance space, a tuner evaluates each configuration

Figure 7 – MetaL PIAC: the proposed framework for algorithm and parameter recommendation.



Source: The author.

based on its utility and defines the best configuration for each algorithm (MH) when solving each instance. The tuned algorithm competes with other tuned algorithms and the "recommended" ones are those with the best mean performance value (best mean fitness) or are statistically equivalent to the one with best mean fitness. The meta-data for MH recommendation is composed of instance features (problem-based and FLA based features) and the recommended MH for each problem. An MH is considered recommended if it is statistically superior to others or equivalent to the MH with the best mean final fitness value. Therefore, it is possible that multiple MHs are recommended at once, making it a multi-label classification problem. The evaluation is made using the multi-label metric macro-AUC and average precision. Finally, the proposal considers training one model for each parameter, using the configurations found by the internal AAC. The output of this procedure is the mapping:

$$\begin{aligned} \alpha' &= h_{MH}(\boldsymbol{\phi}) = \quad \textsc{PredictMultiLabel}(\boldsymbol{\phi}) \\ \boldsymbol{\theta}' &= h_{\alpha',\pi}(\boldsymbol{\phi}) = \quad (\textsc{Predict}(\boldsymbol{\phi}),\pi), \text{ for } \pi \in \Pi_\alpha) \end{aligned} \tag{8}$$

where $h_{MH}(\cdot)$ is the output of the algorithm recommendation model, and $h_{\alpha',\pi}(\cdot)$ is the output of the parameter π model for algorithm α'.

3.2.1 MetaL PIAC Algorithm

Figure 8 details all the steps performed to provide PIAC recommendations.

Figure 8 – MetaL PIAC framework.

1: **procedure** BUILDMETALEARNINGFRAMEWORK(algorithm space A, problem space P)
2: $\boldsymbol{\phi}_p \leftarrow f(p)$ for all problem $p \in P$
3: $\boldsymbol{\theta}^*, \boldsymbol{\alpha}^* \leftarrow$ BUILDPERFORMANCEDATA(A, P)
4: $h_\alpha \leftarrow$ TRAINMODEL($\{(\boldsymbol{\phi}_p, \boldsymbol{\alpha}^*_p) :$ for all problem $p \in P\}$)
5: **for** each algorithm $\alpha \in A$ **do**
6: **for** each parameter π of Π_α **do**
7: $h_{\alpha,\pi} \leftarrow$ TRAINMODEL($\{(\boldsymbol{\phi}_p, \boldsymbol{\theta}^*_{p,\alpha,\pi}) :$ for all $p \in P\}$)
8: **end for**
9: **end for**
10: **end procedure**

11: **procedure** BUILDPERFORMANCEDATA(algorithm space A, problem space P)
12: **for** each instance $p \in P$ **do**
13: **for** each algorithm $\alpha \in A$ **do**
14: $\boldsymbol{\theta} \leftarrow$ AUTOMATICALGORITHMCONFIGURATOR(α, p)
15: $\boldsymbol{\theta}^*_{p,\alpha,\pi} \leftarrow \boldsymbol{\theta}^*_\pi$ for all parameters $\pi \in \Pi_\alpha$
16: **end for**
17: $\alpha^*_p \leftarrow$COMPARE($\{\boldsymbol{\theta}^*_{p,\alpha,\Pi_\alpha} :$ for all algorithm $\alpha\}$, p)
18: **end for**
19: **return** best configurations table $\boldsymbol{\theta}^*$, best algorithms $\boldsymbol{\alpha}^*$
20: **end procedure**

21: **procedure** RECOMMEND(Unseen instance p')
22: $\boldsymbol{\phi}' \leftarrow f(p')$
23: $\alpha' \leftarrow$ PREDICTMULTILABEL(h_α, $\boldsymbol{\phi}$)
24: **for** each parameter π of Π'_α **do**
25: $\boldsymbol{\theta}'_\pi \leftarrow$ PREDICT($h_{\alpha',\pi}$, $\boldsymbol{\phi}$)
26: **end for**
27: **return** $\alpha^*, \boldsymbol{\theta}$
28: **end procedure**

Source: The author.

The first step is to extract the features $feature$ from the problem instances using a given set of metrics, for example, problem-specific metrics and others from FLA (Line 2 in Figure 8). For instance-based recommendations the next step for building the meta-data is to run the tuner for each instance and each algorithm (Lines 13 to 16 of Figure 8). The best configurations are then compared against each other to define the best single algorithm for that instance, using a given statistical test (Line 17 in Figure 8).

Considering the features as inputs and the performance data as outputs, different machine learning models are trained for algorithm recommendation and each individual parameter recommendation (Lines 4 to 9 in Figure 8). At last, these models can be used to predict the best algorithm and, subsequently, its configuration from the parameter recommendation models

(Lines 22 to 26 in Figure 8).

In summary, in MetaL PIAC, the prediction of the best MH and its configuration for unseen instances occurs in three phases:

1. Predicting the best MH strategy. For this, the BR multi-label models for all three inner algorithms (Classification and Regression Tree (CART), Random Forest (RF), and eXtreme Gradient Boosting (XGB)) are used to predict the best MH for all problems from the test set. Since the inner models' output is a probability, the highest probability is selected as the best MH;

2. Predicting the best configurations. All parameter models are used for the chosen MH. If there are parameter dependencies, the dependent models are evaluated later;

3. According to Equation (7), the recommendation results are obtained from the best mapping h^* for unseen problems p' taken from randomly generated test folds.

Unlike previous strategies in the literature, the practitioner only needs to choose the available algorithms to compose the portfolio, since there is no need to tune them with different parameters. Although the procedure to tune each parameter of each problem could be more expensive than solving the base problem directly, it can be easily parallelized. It is also the case for training and prediction of parameter models.

It is possible to extend the meta-data with more algorithms, features, or problems. Adding new instances of the same problem can be done with a small cost, especially with incremental models (VEN; TOLIAS, 2019), like Random Forest or Neural Networks, where the user might just continue to train with new data. Adding new algorithms implies re-training the MH recommendation models and new parameter models. Another possibility is to extend the models for new problem formulations by transfer learning. This implies that the feature spaces of both problem types are related.

Another aspect of our model is the possibility of recommending many algorithms simultaneously. The comparison of configurations (Line 17 in Figure 8) addresses algorithms with similar performance and the configurations therefore could yield a draw. Moreover, the comparison could rank the algorithms accordingly. As a consequence, by having multiple classes or ranking the recommendation models must be chosen to handle multi-label or ranking data.

It is important to note that, during the prediction phase, the recommendation quality diretly depends on how the new problem relates to the problems used during training. For

instance, in a previous work (PAVELSKI *et al.*, 2021b) using MetaL PIAC for NEH algorithms recommendation, the performance of the configuration decreases when the testing problems have more than double the number of jobs of training problems.

3.3 PIAC PROPOSAL BASED ON PROBLEM SPACE DECOMPOSITION

In the present work, we consider that different problems' instances can map to the same configuration, i.e., the same algorithm configuration is the best for a set of instances. Then, the instance space can be decomposed (partitioned) into m sets of related problems, $\{P_1, \ldots, P_m\}$, such that $P = P_1 \cup \cdots \cup P_m$ and $P_i \cap P_j = \emptyset$, for all $i \neq j \in \{1,...,m\}$. In this second proposal, we assume that a function $mshp(\boldsymbol{\phi}) = \boldsymbol{\psi}$ maps the problems' features into a vector $\boldsymbol{\psi} \in [0,1]^m$, such that each element ψ_ρ indicates the membership of problem p to the problem partition subset P_ρ, and $\sum_\rho \psi_\rho = 1$.

Since each problem partition subset can lead to a different configuration, and therefore to a different algorithm's performance, the search for the best configurations can be seen as a multi-objective AAC problem where $m = N_{obj}$. By solving the problem under a multi-objective perspective, we have a Pareto-set approximation $\mathcal{PS}$ of the configurations that optimize the different compromises between the different problem partition subsets P_ρ, $\rho = 1, \ldots, N_{obj}$.

When faced with a new problem p' the following *decision maker* strategy can be used to recommend an adequate configuration:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \mathcal{PS}} d(\boldsymbol{\psi}, \hat{\mathbf{u}}(\boldsymbol{\theta})). \tag{9}$$

where d is an arbitrary distance function, and $\hat{\mathbf{u}}(\boldsymbol{\theta})$ is the vector corresponding to the utility vector $\mathbf{u}(\boldsymbol{\theta})$ scaled to the unit space $[0,1]^{N_{obj}}$. In other words, this strategy simply chooses the configuration whose performance best fits the membership vector $\boldsymbol{\psi}$.

As an example, the function $mshp$ could be used to classify between $P_1 =$ *large* and $P_2 =$ *small* size problems. This way for a new problem p' with an intermediate to large size, the decision maker would choose as the best fit a compromise configuration closer to the *large* objective, i.e. closer to P_1.

The PIAC mapping (7) can be given here equivalently by:

$$h(f(p')|\mathcal{PS}) = \arg\min_{\boldsymbol{\theta} \in \mathcal{PS}} d(mshp(f(p')), \hat{\mathbf{u}}(\boldsymbol{\theta})). \tag{10}$$

Therefore, the PIAC definition in (7) can be generalized to finding the best Pareto-Set ($\mathcal{PS}$) such that:

$$
\begin{aligned}
h^* &= h(f(p')|\mathcal{PS}) \\
\mathcal{PS} &= \arg\min_{\mathcal{PS}\subseteq\Theta} ||\mathbf{q}_{\mathcal{PS}}|| \\
\mathbf{q}_{\mathcal{PS}} &= (\min_{\boldsymbol{\theta}\in\mathcal{PS}} d(mshp(f(p)), \hat{\mathbf{u}}(\boldsymbol{\theta}), \min_{\boldsymbol{\theta}\in\mathcal{PS}} ||\hat{\mathbf{u}}(\boldsymbol{\theta})||), \text{for } p \in P)
\end{aligned}
\tag{11}
$$

where $||\mathbf{q}_{\mathcal{PS}}||$ represents the quality of the Pareto-Set regarding the distance from each membership vector $mshp(f(p))$ to the closest point in the Pareto-Front. As we might expect, finding a good Pareto-Front is equivalent to finding smaller distances between $mshp(f(p))$ and $\hat{\mathbf{u}}(\boldsymbol{\theta})$ (spread) and the norm $||\hat{\mathbf{u}}(\boldsymbol{\theta})||$ (convergence). Therefore, the best Pareto-Fronts yields a good mapping between problems' features and the best compromise configuration.

The decomposition procedure can be performed manually or automatically. In the second case, the feature space F can be used in association of Principal Component Analysis (PCA) to automatically decompose the problem space P into a reduced space of relevant information. Since the feature space F might have many dimensions (one for each feature), we can apply a dimensional reduction method to reduce the feature space into an $\Re^D$ space. Notice that setting not very high D dimensions allows a good number of partition subsets without degenerating the performance of multi-objective optimization algorithms. In case of high D dimensions, many-objective optimization algorithms are better options. In particular, we can build a PCA model to reduce the feature space to D dimensions and select the problems that best represent each of the first D axis. Any metric can be used with PCA to identify such inputs. Therefore the membership function $mshp$ is defined as:

$$
mshp(p) = \boldsymbol{\psi}_p = \text{metric}(PCApred(f(p))) \tag{12}
$$

where $PCApred$ is the model's prediction function, yielding the projected coordinates of the new problem, and metric refers to the quality of this projection on each dimension of the reduced feature space. In this proposal, we use PCA mainly due to its simplicity and because it provides explainable models. Moreover, one could further investigate which are the most important features considered for each objective. Further works might include other clustering methods for this purpose.

In the next sections we detail the proposed approach, which considers the decomposition-based multi-objective context previously discussed. First we detail the MOAAC/D proposal,

particularly its multi-objective formulation. And then we discuss the hybridization of MOEA/D and irace, named iMOEA/D, as a possible solution of the multi-objective formulation.

3.3.1 MOAAC/D: A Multi-objective AAC based on Decomposition

Assuming that the problem space is a set of problem instances $P = \{inst\}$ and it can be decomposed into sets of related problems, $\{P_1, \ldots, P_{N_{obj}}\}$ such that $P = P_1 \cup \cdots \cup P_{N_{obj}}$, the present work proposes an innovative formulation where each problem space partition subset P_ρ is associated with a utility u_ρ, $\rho = 1, \ldots, N_{obj}$, on the objective space and the decision space Θ_α is the set of all possible configurations of algorithm α. The multi-objective configuration optimization problem can then be formulated as (see also Figure 12):

$$\begin{aligned} &\text{Minimize } \mathbf{u}(\boldsymbol{\theta}) = \big(u_1(\boldsymbol{\theta}), \ldots, u_{N_{obj}}(\boldsymbol{\theta})\big) \\ &\text{subject to } \boldsymbol{\theta} \in \Theta_\alpha^f \end{aligned} \tag{13}$$

where N_{obj} is the number of objectives or the total number of problem space partition subsets in our case, $u_p(\boldsymbol{\theta}) = u(\boldsymbol{\theta} \mid \Theta_\alpha^f, P_\rho, C_{\boldsymbol{\theta}}, t), \rho = 1, \ldots, N_{obj}$, and $\Theta_\alpha^f \subseteq \Theta_\alpha$ is the space of feasible configurations of algorithm α. That is, each objective $u_p(\boldsymbol{\theta})$ represents a cost (drawn from $C_{\boldsymbol{\theta}}$) associated with problem partition subset P_ρ, to be minimized within time t, considering as decision variables the algorithm α's configuration $\boldsymbol{\theta} \in \Theta_\alpha^f$.

In contrast to the usual AAC, the proposed formulation considers the fact that there might be several configurations to solve different problem instances. Our proposal can fall in between two different formulation scenarios:

- In one extreme, the problem space can be partitioned into $N_{obj} = |P|$ sets, one for each problem instance, i.e., $|P_\rho| = 1$, for all ρ. In this case, assuming that the best configurations have been found, the Pareto set solutions associated with the extremes of $\mathcal{PF}$ would enumerate each best configuration for each instance. The usual AAC is not adequate for this assumption, since it would at most provide an average-performance generalist configuration.

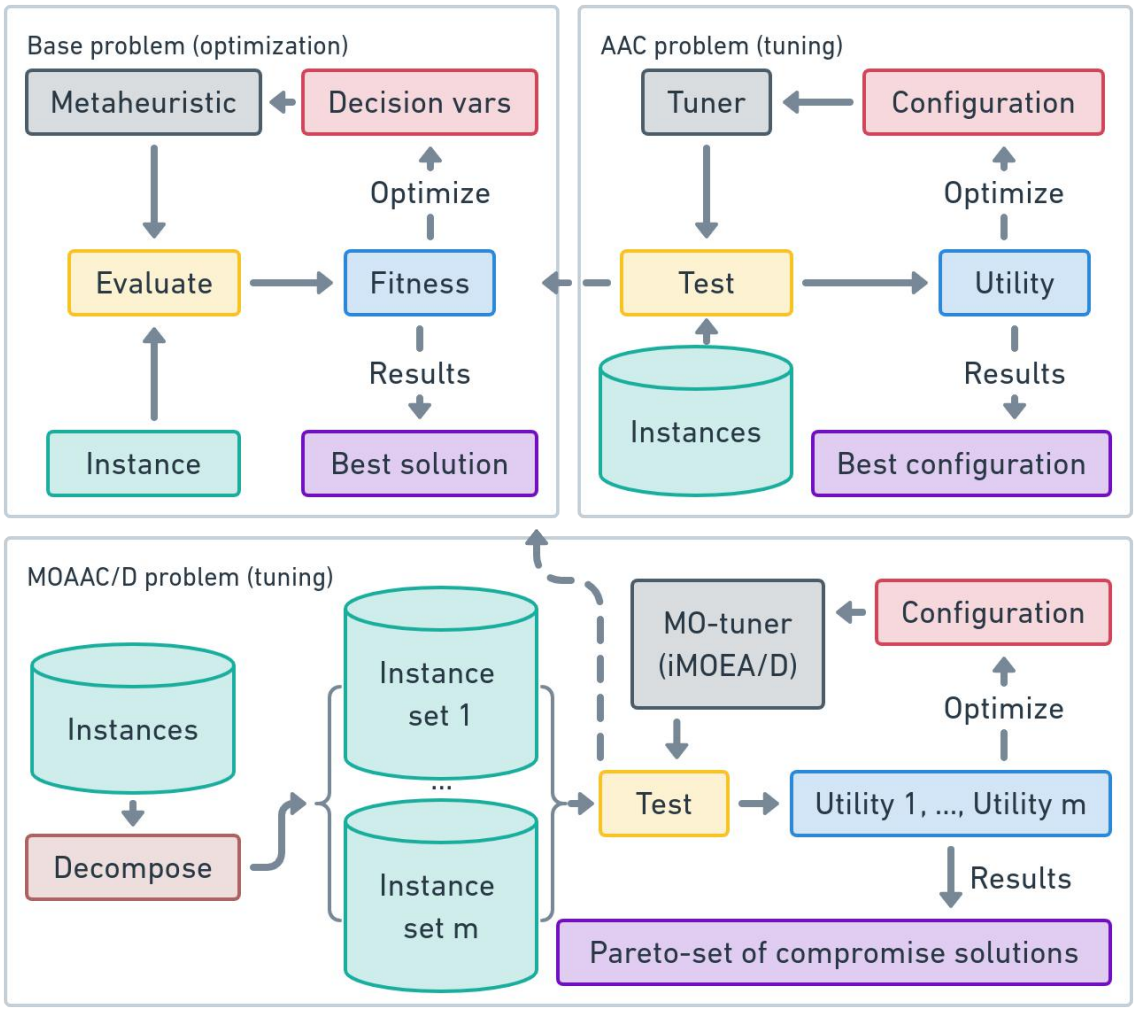
- Alternatively, based on the hypothesis that a unique configuration would be capable of maximizing the performance for all instances, we should not perform any decomposition, assuming a single instance set $P_\rho = P$, $\rho = 1$. Although in this second case, a single AAC

search on any instance would suffice, it is not plausible in general, since it can contradict the No Free Lunch Theorems.

We believe that a more interesting formulation, like the one we propose in the present work, is a trade-off between both scenarios. And more important, it presents a useful characteristic regarding flexibility: based on the level of instance decomposition, i.e. the value of N_{obj}, the proposal guides the formulation toward one of each scenario.

As shown in Figure 9, in the proposed formulation there are three levels of optimization being performed.

Figure 9 – A MOAAC/D overview: base level optimization on the top left side, AAC level optimization on the top right, and MOAAC/D on the bottom.



Source: The author.

In the base level (top left), assuming decision variables $\mathbf{x}$, the MH optimizes the problem formulated in (1) aiming to produce a solution $\mathbf{x}^*$ that minimizes the objective function g (fitness)

for a single problem instance. It should be noted that in the proposal, the base problem can be multi-objective, i.e., formulated as in (2). In this case, the set of solutions $\{\mathbf{x}^*\}$ in the Pareto set are expected to minimize different compromises between the multiple objective functions $\mathbf{g} = (g_1, \ldots, g_{N_{obj}})$ of the base problem. The mapping of $\{\mathbf{x}^*\}$ into the objective space provides the base $\mathcal{PF}(\{\mathbf{x}^*\})$, which is the Pareto Front associated with the base problem.

In the AAC level (top right), a tuner adjusts and evaluates a configuration $\boldsymbol{\theta}$ based on its utility u_ρ, where the utility value depends on a given set of instances P_ρ of the base problem. When the base problem is multi-objective, the utility function must be provided for AAC as a unique indicator value calculated over the base Pareto Front, i.e., $u_\rho = indicator(\mathcal{PF}(\{\mathbf{x}^*\}))$.

In the MOAAC/D level (bottom), a multi-objective tuner[1] adjusts and evaluates configurations $\boldsymbol{\theta}$ considering multiple base problem sets $\{P_1, P_2, \ldots, P_{N_{obj}}\}$, decomposed according to the problem features. A multi-objective optimization algorithm must therefore be considered to find good configurations improving utility functions u_ρ for each problem subsets P_ρ. Therefore, we refer to a problem or instance as being related to a particular objective when it has similar features to the instances used to compute the objective's utility function. In the next section, we describe the approach conceived to solve the MOAAC/D formulation proposed here.

3.3.2 iMOEA/D: irace+MOEA/D for MOAAC/D

In this section, we present a practical method to solve AAC problems using the problem space decomposition. For that we use a decomposition-based MOEA with a proposed local search procedure based on a given AAC. We present this proposal in terms of the MOEA/D and irace algorithms, but it could be easily adapted for other strategies for the AAC and MOEA.

The configuration space Θ can be complex, containing different types of parameters, dependencies and constraints. Aiming to apply the multi-objective formulation from Equation 13, we can consider state-of-the-art MOEAs based on decomposition and, at the same time, existing successful AAC search strategies. Described in the algorithm in Figure 10, in the hybrid tuner named iMOEA/D, we consider that MOEA/D embeds irace as a local search operator capable of leveraging the decomposition aspects of MOEA/D.

The iMOEA/D general framework described in Figure 10 and works as follows. First, the population is initialized with N_{pop} randomly generated configurations. Then, as in the original MOEA/D, each weight vector $\mathbf{w}_i \in W$ with N_{obj} elements in $[0,1]$ is set in a way that W provides

[1] Any multi-objective optimizer could be adopted here.

Figure 10 – iMOEA/D

1: **procedure** IMOEA/D(P, N_{pop}, N_{gen}, N_{neigh}, $N_{samples}$, N_{evals})
2: $\Theta_{pop} \leftarrow N_{pop}$ random configurations from Θ^f
3: $W \leftarrow$ UNIFORMWEIGHTS(N_{pop})
4: Calculate $\mathbf{u}(\boldsymbol{\theta}_i), i = 1, \ldots, N_{pop}$
5: $B \leftarrow$ ASSIGNNEIGHBORS($\{\mathbf{u}(\boldsymbol{\theta}_i)\}, W, N_{neigh}$)
6: $\{P_1, \ldots, P_{N_{obj}}\} \leftarrow$ DECOMPOSE(P, N_{obj})
7: $gen \leftarrow 1$
8: **while** $gen < N_{gen}$ **do**
9: **for** $\boldsymbol{\theta}_i \in \Theta_{pop}$ **do**
10: $\boldsymbol{\theta}_i^{new} \leftarrow$ IRACELS($B_i, \mathbf{w}_i, \{P_1, \ldots, P_{N_{obj}}\}, N_{samples}, N_{evals}$)
11: Calculate $\mathbf{u}(\boldsymbol{\theta}_i^{new})$
12: Update non-dominated archive
13: $\vartheta^* \leftarrow \min\{\vartheta^*, \mathbf{u}(\boldsymbol{\theta}_i^{new})\}$
14: **for** $\boldsymbol{\theta}_k^{ngh} \mid k \in B_i$ **do**
15: **if** $g^{tche}(\mathbf{u}(\boldsymbol{\theta}_i^{new})|\mathbf{w}_i,\vartheta^*) \leq g^{tche}(\mathbf{u}(\boldsymbol{\theta}_k^{ngh})|\mathbf{w}_i,\vartheta^*)$ **then**
16: $\boldsymbol{\theta}_k^{ngh} \leftarrow \boldsymbol{\theta}_i^{new}$
17: $\mathbf{u}(\boldsymbol{\theta}_k^{ngh}) \leftarrow \mathbf{u}(\boldsymbol{\theta}_i^{new})$
18: **end if**
19: **end for**
20: **end for**
21: $gen \leftarrow gen + 1$
22: **end while**
23: **return** Non-dominated archive
24: **end procedure**

Source: The author.

a uniform distribution in the objective space. Each individual configuration $\boldsymbol{\theta}_i$ in the population is evaluated (line 4) and associated (line 5) with a weight $\mathbf{w}_i$, and indexes of the closest N_{neigh} weights form the neighborhood set (B_i) of each configuration $\boldsymbol{\theta}_i$. Next, as an exclusive step of the proposed approach, the set of problem instances P is decomposed into non-overlapping subsets P_ρ, $\rho = 1, \ldots, N_{obj}$.

The main loop starts with every configuration in the population being modified to generate a new configuration. The modification is performed by an irace-based local search. After that, the new configuration $\boldsymbol{\theta}_i^{new}$ is evaluated. A non-dominated solution archive is maintained, and is updated whenever irace finds out a non-dominated configuration. The new solution replaces all the solutions in the archive that contain worse aggregation function values. In the sequence, an aggregation function $g^{tche}(\cdot)$ compares, in a mono-objective way, $\boldsymbol{\theta}_i^{new}$ with every neighbor configuration $\boldsymbol{\theta}_k^{ngh}$. There is an update in the k-th neighbor whenever it is outperformed by the new configuration. In this case, a new solution can replace any number of parents. The main loop repeats for N_{gen} generations. Since in our case the objective function is quite costly (evaluate a configuration on several problems), we use low values for the population size (8 to 20 individuals) and number of generations (50 when using irace as local search). At the end, all the

configurations present in the non-dominated solutions archive are returned as the approximated Pareto Set.

We can use any aggregation function to evaluate the vector $\mathbf{u}(\boldsymbol{\theta}) = (u_1(\boldsymbol{\theta}), \ldots, u_m(\boldsymbol{\theta}))$ of configuration utilities according to the weight $\mathbf{w}_i$. For example. the Tchebycheff aggregation function is shown in (14).

$$g^{tche}(\mathbf{u}(\boldsymbol{\theta})|\mathbf{w}_i,\vartheta^*) = \max_{1\leq p\leq N_{obj}} \{w_{pi} \left|u_p(\boldsymbol{\theta}) - \vartheta_p^*\right|\} \tag{14}$$

where ϑ^* is the current reference point, composed of the best value of each configuration utility.

The algorithm in Figure 11 describes the local search (LS) using irace. It is performed on $\boldsymbol{\theta}_i$ at every generation of Figure 10 (see Line 7). During the local search, irace considers a maximum of N_{evals} configurations tests, each test involving the optimization of the base problem through the MH parameterized by the candidate configuration.

Besides the neighborhood B_i and the associated weight $\mathbf{w}_i$, Figure 11 also requires the decomposed instance sets $\{P_1, \ldots, P_{N_{obj}}\}$, and two additional hyper-parameters: the total number of instances to be sampled ($N_{samples}$) and the maximum number of configuration tests (N_{evals}) performed by irace. At each run, irace uses the particular vector ($\mathbf{w}_i$) and its neighborhood (B_i) information to produce new offspring solutions. Moreover, it considers instances sampled from the different decomposed sets to perform the local search.

Figure 11 – irace Local Search.

1: **procedure** IRACELS(B_i,$\mathbf{w}_i$,$\{P_1, \ldots, P_{N_{obj}}\}$, $N_{samples}$,N_{evals})
2: $P_{irace} \leftarrow \emptyset$
3: **for** $\rho = 1, \ldots, N_{obj}$ **do**
4: $P_{aux} \leftarrow$ sample $w_{pi} \times N_{samples}$ instances from P_ρ
5: $P_{irace} \leftarrow P_{irace} \bigcup P_{aux}$
6: **end for**
7: $\Theta_{init} \leftarrow \emptyset$
8: **for** $\boldsymbol{\theta}^{ngh} \in B_i$ **do**
9: $\Theta_{init} \leftarrow \Theta_{init} \bigcup \{\boldsymbol{\theta}^{ngh}\}$
10: **end for**
11: $\boldsymbol{\theta} \leftarrow$ IRACE($\Theta_\alpha, P_{irace}, \Theta_{init}, N_{evals}$)
12: **return** $\boldsymbol{\theta}$
13: **end procedure**

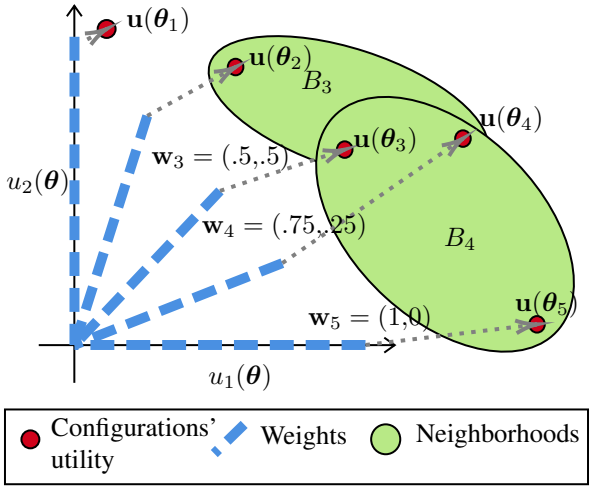
Source: The author.

In the first steps (Lines 2 to 6), the algorithm in Figure 11 samples instances from each set P_ρ, proportional to the ρ-th component $w_{\rho i}$ in the weight vector $\mathbf{w}_i$. After that, the algorithm in Figure 11 takes solutions from the neighborhood B_i as the initial configurations (Θ_{init}) for irace. Starting with the initial configurations, irace iteratively samples new configurations from

a distribution. The new configurations are evaluated and raced against each other. Statistically better configurations proceed to the next iterations and the distribution is updated. In a sense, one could think of this local search as evolving the irace search space itself. However, the internal irace' models are reset every time.

Figure 12 shows an example of the objective space on which iMOEA/D (Figure 10) could run.

Figure 12 – An example of objective space decomposition: bi-objective formulation ($u_1(\boldsymbol{\theta}), u_2(\boldsymbol{\theta})$) with five individuals in the population ($\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_5$), five corresponding weights ($\mathbf{w}_1, \ldots, \mathbf{w}_5$), $\boldsymbol{\theta}_3$ and $\boldsymbol{\theta}_4$ and neighborhoods ($\mathbf{B}_3 = \{\boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4\}$, $\mathbf{B}_4 = \{\boldsymbol{\theta}_3, \boldsymbol{\theta}_4, \boldsymbol{\theta}_5\}$).



Source: The author.

It shows a MOAAC/D formulation with $N_{obj} = 2$ objectives and $N_{pop} = 5$ individuals of Θ_{pop}. Every configuration $\boldsymbol{\theta}$ is associated in the objective space with a 2D coordinate:

$$\mathbf{u}(\boldsymbol{\theta}) = (u_1(\boldsymbol{\theta}|\Theta_\alpha, P_1, C_{\boldsymbol{\theta}}, t), u_2(\boldsymbol{\theta}|\Theta, P_2, C_{\boldsymbol{\theta}}, t))$$

which represents the utility of algorithm's (α) configuration $\boldsymbol{\theta}$ on the problem instance sets P_1 and P_2, respectively. In the MOEA/D framework, each configuration $\boldsymbol{\theta}_i \in \Theta_{pop}$ has also a corresponding weight $\mathbf{w}_i$. In the example shown in Figure 12, the proposed irace-based local search (Figure 11) when performed on $\boldsymbol{\theta}_4$ would sample 75% problems from P_1 and 25% problems from P_2 to provide P_{irace}. It would also use the neighbors $\{\boldsymbol{\theta}_3, \boldsymbol{\theta}_4, \boldsymbol{\theta}_5\}$ as the initial configurations for irace. Notice that MOEA/D is proposed for continuous problems, therefore in its canonical version, it generates new individuals with simulated binary crossover and polynomial mutation.

Therefore, in the proposed approach, two levels of decomposition occur: (i) one performed off-line on the problem space to formulate the MOAAC/D (each decomposed subset gives

rise to a particular utility in the multi-objective objective space); (ii) other performed online by the multi-objective algorithm to solve the MOAAC/D problem (each decomposed sub-problem, associated to a weight vector $\mathbf{w}_i$ can be solved in a traditional way by means of MOEA/D enhanced by irace). The original irace algorithm performs the search focusing on instances partition subsets whose weight components are higher and uses the neighborhood solutions as the initial configurations. For example, for weights located exactly on each dimension u_ρ, it performs the race exclusively for this objective, using nearby solutions, therefore, optimizing a specialist configuration to the decomposed set P_ρ. For weights whose components are balanced, it performs the race on central regions of the objective space looking for generalist configurations with good trade-offs between the decomposed sets in the problem space. When the base problem is multi-objective and we also consider a decomposition-based MOEA to solve it, a third level of decomposition might occur. However, as discussed in the next section, in the present work, the addressed base problem is a mono-objective combinatorial optimization problem.

We can notice that the MOAAC/D formulation is general and can be solved by any MOEA. Also, the proposed iMOEA/D mixes the original MOEA/D with irace, but any decomposition based algorithm and AAC could be used in the same manner. For such, in Chapter 5 we present results for MOAAC/D formulation using the original MOEA/D where the proposed irace-based local search is replaced by the original genetic operators.

In summary, in MOAAC/D recommendation, the training phase uses iMOEA/D to find the approximated Pareto-set of solutions. The problem space partitioning might be done manually, if the user knows or is interested in a given separation. Alternatively, given the problem features, the problem space can be partitioned into D dimensional principal components using PCA. Finally, the prediction is done by:

1. Applying the new instance's features to the PCA model to obtain its coordinates in the reduced dimensionality space;

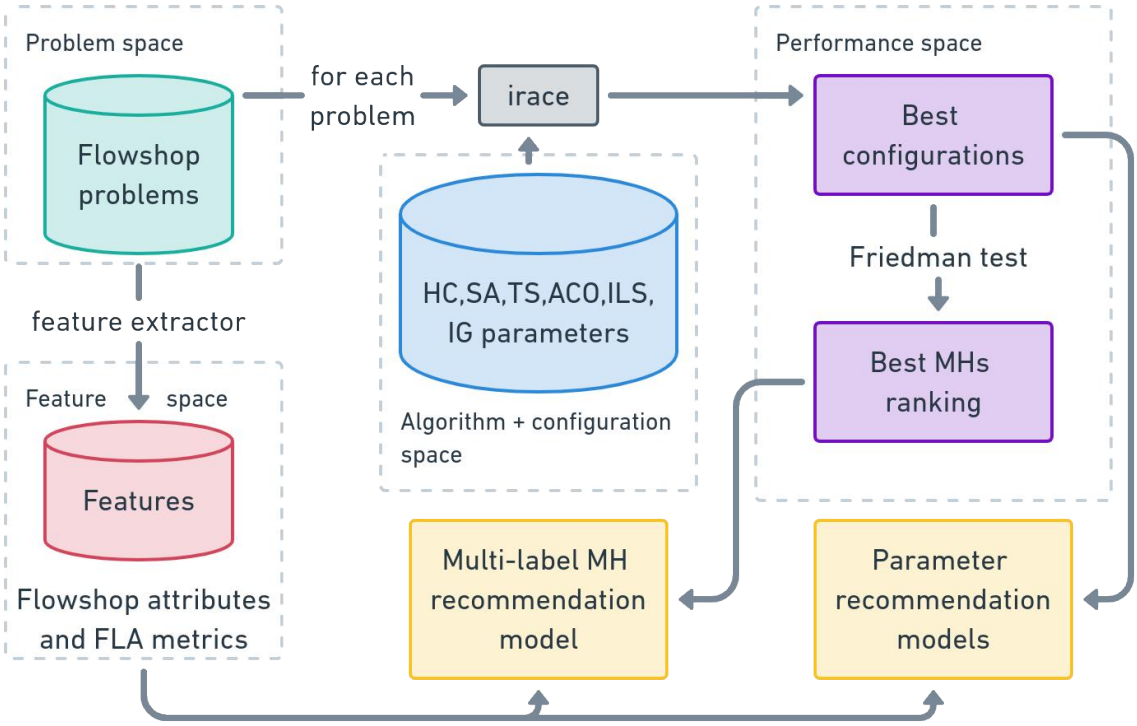
2. Computing the values of a given PCA metric to evaluate how suitable is the new coordinates to the first D components;

3. Using an aggregation function to find the configuration that best fits the metric values.

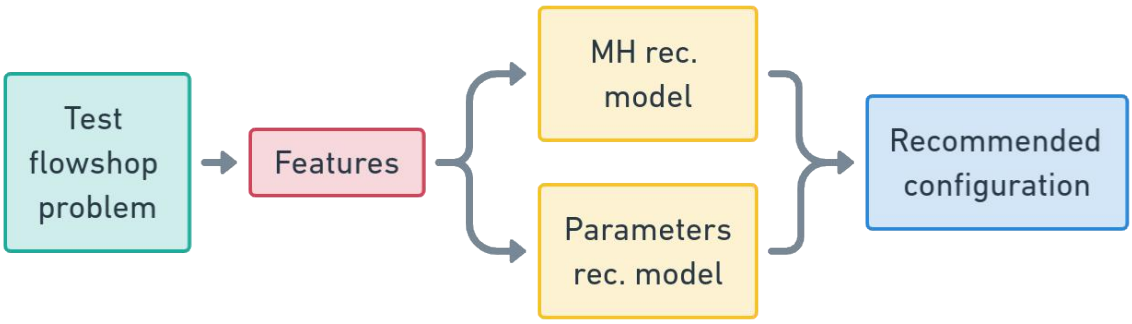
4 PIAC ON FLOWSHOP PROBLEMS

In this chapter we describe the application of the proposed approaches to several instances of FlowShop Problems (FSPs). As discussed in Section 2.5, FlowShop Problems (FSPs) have been studied since the 1950s, with many applications, variants and proposed MHs. In the present work we explore these problems from the perspective of a practitioner looking for a suitable MH and its configuration.

Figure 13 – The proposed framework for MetaL PIAC applied to flowshop problems.



(a) Training phase.

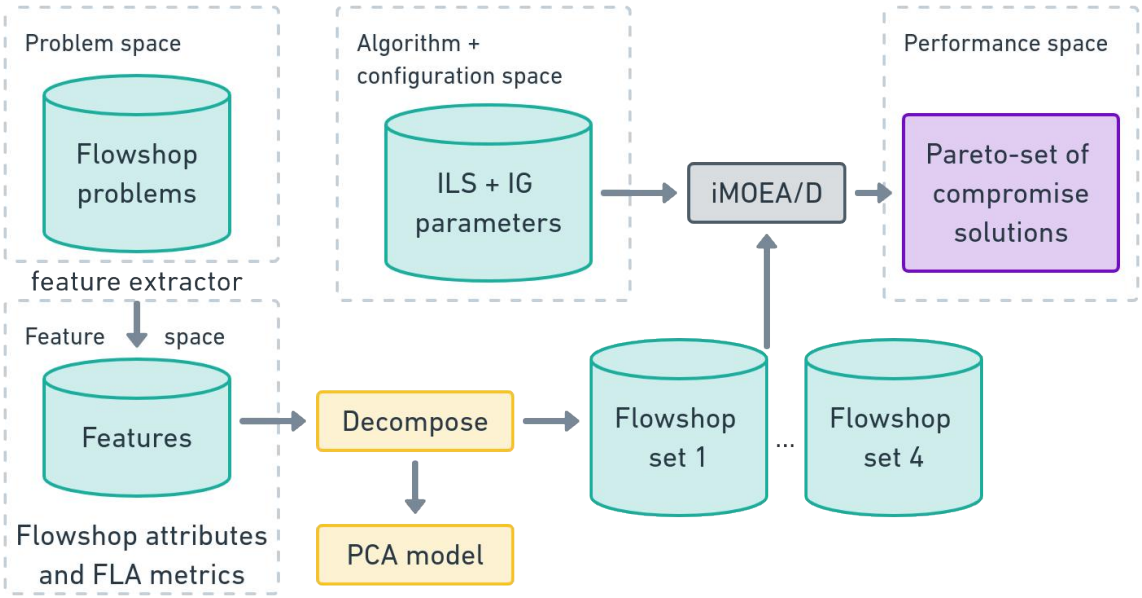


(b) Testing phase.

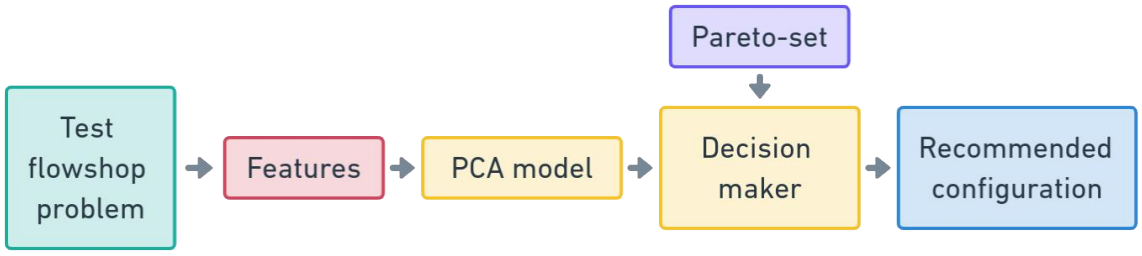
Source: The author.

As shown in Figure 13, the main components of the proposed MetaL PIAC framework. These components are associated with the same four spaces defined in the Rice's framework (algorithm, problem, features, performance) and consider the FSP as the base-level problem. Besides FSP, the figure also highlights the type of addressed algorithms (MHs for solving FSP), meta-features (basic and optimization plus FLA features), the statistical test used to set the recommended MH and its configurations.

Figure 14 – The proposed framework for MOAAC/D applied to flowshop problems.



(a) Training phase.



(b) Testing phase.

Source: The author.

As shown in Figure 14 the main components of the proposed iMOEA/D framework encompasses irace as local search and MOEA/D as the multi-objective algorithm based on decomposition used to find configurations for FSP problem. The diagram also highlights the principal components model used in the multi-objective decomposition and ILS and IG as the main MHs being analyzed.

In the next sections, we describe the set up for problem, feature, algorithm and performance spaces of the PIAC formulation proposed in the present work. From now on we will refer to configuration performance as the utility of a configuration $\boldsymbol{\theta} \in \Theta_{\alpha}^{f}$ for a particular algorithm α (or metaheuristic MH) on a specific FSP problem p (or instance $inst$).

All experiments are performed on Intel i7 machines with 16 GB of RAM. We implement the algorithms using R language and packages like `future` to use multiple cores wherever the proposals allow parallel execution, i.e. building MetaL PIAC performance data and evaluating iMOEA/D configurations. The base flowshop MHs are implemented in C++ using ParadisEO framework (HUMEAU *et al.*, 2013). The time to reproduce all experiments in the thesis would be two weeks using 16 cores. The source code and instances data are available at https://github.com/lucasmpavelski/FlowshopSolveR.

4.1 BUILDING THE PROBLEM SPACE (P)

FSP problems model a production line where jobs are processed by sequential machines (see Section 2.5). To perform the experiments we divide the attributes of the problem space into two main categories:

- **Basic** attributes: the number of jobs (J), the number of machines (M), the processing times distribution ($distribution$), and correlation type ($corr$);
- **Optimization** attributes: the objective function (g) to be optimized, the budget size ($budget$) for solving the problem, and the stop condition ($stopC$).

4.1.1 Setting Up the **Basic** Attributes

Several works point that the number of jobs influences the flowshop problem difficulty on uniformly random instances (REEVES, 1995). The number of machines is usually smaller in practice and the difficulty seems to depend mostly on the number of jobs.

The times taken to process each job on each machine are given as a positive matrix whose element $p_{j,m}$ is the time to process job j on machine m. In this work, processing times are generated randomly from different multivariate distributions to simulate several different instances, modeling different production-line environments.

The parameters used to generate different flowshop instances are the sizes (J and M),

the processing time distribution, and the correlation among processing times of different jobs and machines. More specifically, we use all combinations of attributes shown in Table 3.

Table 3 – FSP basic attributes.

Feature	Description	Numerical/Categorical values
J	The number of jobs	$\{10, 20, 30, 50\}$
M	The number of machines	$\{5, 10, 20\}$
$distribution$	Processing time distribution	{uniform, binomial, exponential}
$corr$	Correlation type	{uncorrelated, machine-correlated, job-correlated}

Source: The author.

All distributions have a mean processing time of 50 time units. For the job-correlated (machine-correlated) processing times each line (column) of the processing times matrix is generated with a correlation of 0.95 from other lines (columns). Figure 15 shows examples for instances with 100 jobs and 2 machines using different correlations types and processing times distributions. Without loss of generality, we generate integer time steps (rouded to the nearest integer in case of the exponential distribution), since they are faster to perform comparisons and sum.

Given the processing time distributions, in the experiments we consider multiple instances samples, and use multiple random number generator seeds to create similar instances.

4.1.2 Setting up the **Optimization** Attributes

Other attributes can be used to define the Problem Space P. In the present work, as optimization plays a fundamental role, we consider attributes associated with different aspects of base problem optimization.

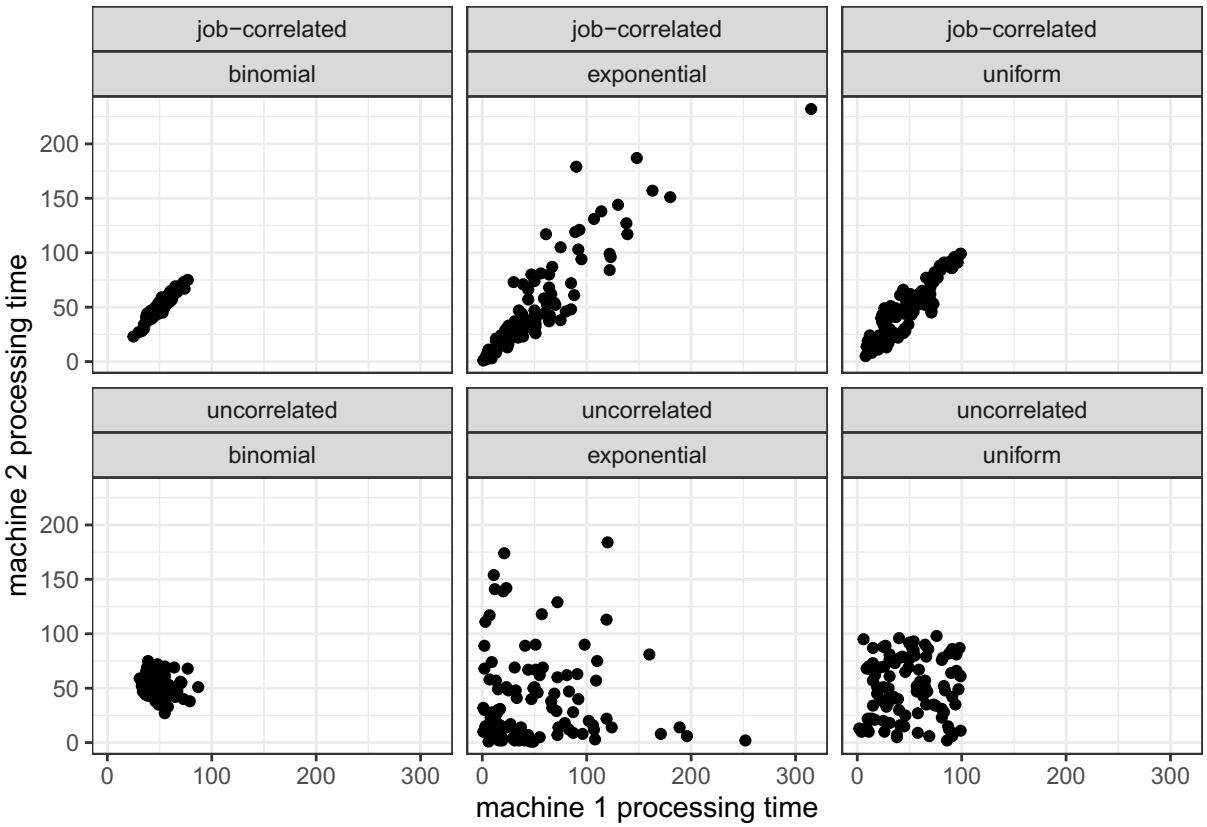
Given an instance and its processing times, FSP can be modeled as:

$$\begin{aligned} &\text{minimize } g(p_{j,m}(\mathbf{x})), \text{ for all j } \in \{1, \ldots, J\} \text{ and m } \in \{1, \ldots, M\}) \\ &\text{subject to } \mathbf{x} \text{ is a permutation} \\ &\text{subject to a maximum } budget \text{ of } stopC \end{aligned} \tag{15}$$

where $g(.)$, $budget$, $stopC$ are the objective function, computational budget, and stopping criterion, respectively.

In the experiments we consider all the attributes addressed in (15), i.e., the type of objective function g, stopping criterion $stopC$, budget $budget$ to solve the problem, as shown in Table 4.

Figure 15 – Example of processing times distributions for an instances with $J = 100$ and $M = 2$ using two correlations (uncorrelated and job-correlated) and three different distributions (binomial, exponential and uniform)



Source: The author.

Table 4 – FSP optimization attributes.

Feature	Description	Categorical Values
function g	The objective (to be minimized)	{makespan, total flowtime}
$stopC$	The stopping criterion	{number of evaluations (eval) time limit (time)}
$budget$	The budget to solve the problem	{low, medium, high}

Source: The author.

Table 5 shows how we set in the experiments numerical values for the three budget types (low, medium, high) considered in Table 4.

Table 5 – Maximum budgets for each stopping criterion.

Budget	Evaluations	Time (ns)
low	$J \times M \times 10$	$J^2 \times M \times 0.2$
med	$J \times M \times 100$	$J^2 \times M \times 2$
high	$J \times M \times 1000$	$J^2 \times M \times 20$

Source: The author.

4.1.3 FSP Problem Space: Basic × Optimization Attributes

As discussed in the previous sections, the basic subspace contains FSP problems with a given number of jobs, number of machines, processing times distribution, and processing times correlations. The optimization subspace is composed of problems with different objective functions, budgets, and stopping criteria.

Instances with the same basic attributes can be solved with different optimization attributes. Therefore the FSP space P can be defined as the Cartesian product of basic *versus* optimization subspaces.

For example, assuming $J = 10$, $M = 5$, the fitness, i.e., the objective function $g(.)$, as makespan, $stopC$ as the total number of evaluations and the $budget = 10{,}000$ evaluations of makespan, a specific instance of the base problem could be described as:

$$\begin{aligned} &\text{minimize } \textsc{makespan}(p_{10\times5}) \\ &\text{subject to a maximum of } 10000 \textsc{ evaluations} \end{aligned} \tag{16}$$

All combinations of problem attributes yield a total of $1296 \times 5 = 6480$ problems, where 5 is the number of instance samples, generated with the same attributes but different seeds. Despite only considering small instances ($J \leq 50$, $M \leq 20$) due to the high computational cost of solving the instances several times, we believe that these problems cover a representative portion of the problem space. Moreover, experiments on bigger instances found in the literature have been performed in a publication by the authors (PAVELSKI *et al.*, 2021b), which has a focus slightly different from the one of the present work since that work is focused on the use of NEH for solving FSP with MetaL PIAC.

4.2 BUILDING THE FEATURE SPACE (F)

The feature space F considered in this work is composed of two feature types: problem-based features and fitness landscape-based features. The problem-based features are the attributes described in Section 4.1: number of jobs, number of machines, distribution, correlation, objective, budget, and stopping criterion. The more advanced features considered are based on FLA.

The features that compose the meta-data in meta-learning experiments are:

- 5 simple problem features: number of jobs, number of machines, objective, budget and

stopping criterion;

- 5 simple computed features: the ratio between the number of jobs and machines, the mean deviation between job processing times, the mean deviation between machine processing times, mean correlation between processing times, mean correlation between machine ties;
- 10 solution statistic FLA metrics: percentage of the plateau, ledge, slope, local maximum or minimum, strict local maximum or minimum solutions and up, down, side edges;
- 7 random-walk FLA metrics: autocorrelation with delay 1, 2 and 4, entropy, partial information, information stability, and density basin;
- 22 adaptive-walk FLA metrics: mean number of steps, FDC using the seven different distances from Table 1 and three local search procedures: first improvement hill-climbing, best improvement hill-climbing, and best insertion.

We do not include the instance processing time and correlation type because the practitioner might not know these parameters exactly in the test phase.

For training better models, the feature dataset is pre-processed based on the following steps:

- Create numerical dummy variables: transform discrete features like budget and objective into a number of numeric indicator features (for example 0 for makespan objective and 1 for flowtime) to facilitate further processing methods;
- Zero variance feature elimination: filter features like frequency of plateau, local minima and maxima solutions, which are constant for all instances;
- Highly correlated features: recursive elimination of features with an absolute correlation higher than 0.95. Highly correlated features include mean walk length with the tree different strategies, random-walk autocorrelation with 2 and 4 delays, FDC with precedence distances that are similar to FDC with shift distance;
- Linear combinations elimination: remove features that can be approximated by a linear combination of other features: like solution and edge statistics that always sum to 1, therefore one of the features can be removed;

Additionally, the experiments include another reduction procedure based on information gain feature selection. The information gain measures how much information (entropy) is given by the inputs considering the target output values (MITCHELL, 1997). Features with less than 0.001 information gain are eliminated with respect to the output for each meta-learning task.

Besides providing the meta-data, a side benefit of using FLA is that we can also gain insights on the problem space, such as the effects of different operators and hardness. In Appendix C we analyze four types of FLA metrics, based on solution and edge type statistics, random walks, adaptive walks, and local optima networks. The main insights obtained from the analysis are that:

- Ruggedness metrics like the random walk auto-correlation are proportional to the number of jobs;
- Mean-distance between local optima and basin size is dependent on the types of instance correlation (job correlated instances have small number of very connected local optima and machine correlated instances have many local optima in large spread regions);
- Considering the proportion of improvement of a state-of-the-art IG over a simple HC as a difficulty measure, the best FLA metrics are the ones based on compressed LONs (VEREL *et al.*, 2018), such as the mean network size (number of neutral regions) and mean node size (size of the neutral region).

Despite having good descriptive quality, LON-based metrics have high computational cost as meta-features. This is particularly critical for small instances, since the cost of calculating the LON is bigger than the cost of solving the problem several times. Therefore, although they contribute to the problem understanding, FLA metrics based on LONs have not been included in the meta-data of the present work. Nevertheless, details of the analysis on LON-based metrics are discussed in Pavelski *et al.* (2021c).

4.3 BUILDING THE ALGORITHM AND CONFIGURATION SPACE: ($A + \Theta$)

In this work, the whole space of the algorithms encompasses not only the algorithms (MHs) themselves but also all possible configurations of its parameters. Therefore, the task of selecting from a portfolio of algorithms configured *a priori* is generalized to the task of selecting the algorithm and its configuration from the expanded space $A + \Theta$.

In the experiments, we consider six different MHs: Hill Climbing (HC) (RUSSELL; NORVIG, 2003), Simulated Annealing (SA) (KIRKPATRICK *et al.*, 1983), Tabu Search (TS) (GLOVER, 1989), Ant Colony Optimisation (ACO) (STÜTZLE, 1998a), Iterated Local Search (ILS) (STÜTZLE, 1998b), and Iterated Greedy (IG) (RUIZ; STÜTZLE, 2007). All MHs and operators are implemented using the ParadisEO framework (HUMEAU *et al.*, 2013).

The main reason to address all these MHs, is that they have been successfully used to solve FSPs. We use the original implementations of these MHs, but we also consider some extensions like NEH initialization, three types of cooling schedules for SA, TS with different types of tabu list, ACO with single-step iterations, IG with partial solution optimization (Dubois-Lacoste *et al.*, 2017) and others. The full list of parameters for each MH is described in Appendix B.

It is worth mentioning that the configuration space Θ for flexible MHs like ILS and IG is quite complex. Parameters can be categorical, integer, or real-valued. There are dependencies between values; for example, the acceptance temperature for ILS is only necessary for the acceptance criterion based on Metropolis-Hastings (other possibilities are to accept any or accepting only improving solutions). Moreover, as discussed in the next chapter, ILS and IG present the best performance for almost all the instances addressed by MetaL PIAC. Therefore, in the experiments associated with the second proposal (MOAAC/D), we focus on these two MHs analysis.

4.4 BUILDING THE PERFORMANCE SPACE (U)

Using the problem, feature and algorithm spaces described in the previous sections, the next step for solving the PIAC problem formulated under the Rice's framework involves building the performance space U, which associates, to the feature vector ϕ extracted from every candidate configuration $\boldsymbol{\theta}$, a performance metric value $u(\boldsymbol{\theta})$ or a set of performance metric values $\mathbf{u}(\boldsymbol{\theta})$. A challenge in this task is defining the best metric and how many evaluations (N. of tests) are enough for a good recommendation. As detailed in the next sections, the performance space U is modeled, and thus built, differently for MetaL PIAC and MOAAC/D.

4.4.1 MetaL PIAC performance space

In the first proposal, the space of performance on the permutation FSP is built, for all MHs and their configurations, using irace as the AAC component in the meta-learning

framework (see Figure 13). For this, irace runs with its default hyper-parameters from the R package version 2, aiming at solving the AAC problem given by (5). In the experiments performed in the present work, the problem space P encompasses a set of training ($\{inst\}_{tr}$) an testing instances ($\{inst\}_{ts}$), and the function $t : P \rightarrow \mathbb{R}$, which associates the computation time/resource (budget) allocated for each base problem $p \in P$, follows the rules presented in Table 5. During its run, irace distributes internally the maximum number of configuration evaluations ($N.tests$), among the races. So, it is not possible to identify the limit of evaluations established for each configuration.

An example of configurations data provided by irace on each problem addressed in the present work is analyzed in Appendix A. The main observations are that some parameters values are very correlated, therefore they might interact with each other and some parameter values like random initialization are not used, NEH-based initialization is preferred (see Section 2.5 for more details).

As mentioned in Section 3.2, in the process of indicating the best configurations for each MH at the performance space of MetaL PIAC, a statistical test is used to compare the best final configurations for each problem (instance) aiming to determine the best MH. In the experiments, the distribution costs are estimated using $S = 50$ different seeds, and the samples achieved by the different MHs are compared using the non-parametric Friedman test (CONOVER, 1999). If the test considers the samples of the costs distribution different with a statistical confidence of 95%, an all-pairs post-hoc Conover test is used. The "recommended" MHs are the ones that have the best mean values for cost distributions or are statistically equivalent to the MH with best mean. For the configuration recommendation, all configurations found by irace are considered for building the performance space.

4.4.2 MOAAC/D Performance Space

In the second proposal, the space of performance on the permutation FSP is built, for ILS and IG and their configurations, using iMOEA/D as the MOAAC/D component (see Figure 14). For this, MOEA/D and irace run with the hyper-parameters presented in Section 4.5.2, aiming at solving the MOAAC/D formulation given by (13). In the experiments performed in the present work, the function t set as the budget also follows the rules presented in Table 5. The problem space P also encompasses a set of training $\{inst\}_{tr}$ and testing $\{inst\}_{te}$ instances. However, differently from the MetaL PIAC, here the problem space associated with the training

instances is decomposed into N_{obj} non-overlapping sets P_ρ, $\rho = 1, \ldots, N_{obj}$ and each objective $u_p(\boldsymbol{\theta})$ is associated with problem partition $P_\rho \in \{inst\}_{tr}$. As in the first proposal, $u_p(\boldsymbol{\theta})$ can be estimated based on sample costs (drawn from $C_{\boldsymbol{\theta}}$).

Differently from the MetaL PIAC, where a statistical test compares directly the samples $c_{s,\boldsymbol{\theta}}$, $s = 1, \ldots, S$, of different cost distributions $C_{\boldsymbol{\theta}}$, here an average cost calculated based on S samples is used to assess the utility value. In the experiments, we consider the utility metric given by Average Relative Performance Difference (ARPD), which is based on the relative difference between the fitness $g(\mathbf{x}_{s,\boldsymbol{\theta},p})$ found using the configuration $\boldsymbol{\theta}$ and seed s and the best known fitness $g(\mathbf{x}^*_p)$ for each problem p. ARPD can be averaged over several instances, e.g. over all instances of partition subset P_ρ, and S seed samples, resulting in an estimation of the configuration utility value for a problem partition subset ρ given by:

$$u_\rho(\boldsymbol{\theta}) = ARPD(\boldsymbol{\theta}) = \frac{1}{|P_\rho|} \sum_{p \in P_\rho} \frac{1}{S} \sum_{s=1}^{S} \frac{g(\mathbf{x}_{s,\boldsymbol{\theta},p}) - g(\mathbf{x}^*_p)}{g(\mathbf{x}^*_p)} \tag{17}$$

4.5 INDIVIDUAL EVALUATION OF THE PIAC PROPOSALS

In the previous sections, the set up performed in every space of both PIAC proposals provided a general framework for experiments in which the problem space P associated with the FSP problem encompasses more than 6000 generated instances. Moreover, six MHs and their parameters compose the algorithm+configuration space of MetaL PIAC, and a hybrid MH joining ILS and IG, as well as the associated parameters, compose the algorithm+configuration space for MOAAC/D. Simple metrics like basic and optimization attributes, all of them extract from the FSP instances, are joined with more complex FLA-based metrics to form the feature space F. The performance space U is built differently for the two proposals: i) MetaL PIAC considers the results of irace (López-Ibáñez *et al.*, 2016) to compose the configuration performance space and compares these results using the Friedman statistical test to provide the recommended MHs; ii) MOAAC/D considers either the results of irace running as a local search or the modifications performed by the genetic operators of MOEA/D to provide in the Pareto set $\mathcal{PS}$ specialists and generalists configurations for a particular MH, which are mapped to the extremes or middle of the same Pareto Front $\mathcal{PF}$.

In this section, we describe how each proposal builds and uses the mapping h to recommend the MH configurations. Sections 4.5.1 and 4.5.2 show that the mapping h is built

exclusively from machine learning models for the first proposal and from PCA plus Multi-objective decision-making for the second one. We also describe the experiments designed to individually evaluate the proposals.

In Section 4.5.1, we describe the experiment in which MetaL PIAC is evaluated under a machine learning perspective when performing regression or classification tasks. From an AAC perspective and considering iMOEA/D as the proposed multi-objective framework, in Section 4.5.2 we discuss how MOAAC/D can be evaluated on manually performed partitions of the problem space and compared with irace-based baselines. We compare, as part of our proposal, the iMOEA/D and the original MOEA/D algorithms for solving the MOAAC/D formulation. Therefore, the purpose of this comparisons are to show that MOAAC/D formulation is capable of improving the irace mono-objective baselines, as well as the effectiveness of iMOEA/D local search.

In Section 4.5.2.2, an experiment focused on the PIAC task is described. It aims at evaluating iMOEA/D when recommending configurations over a problem space that is automatically partitioned. Finally, in Section 4.6, we describe the experiment that compares both approaches with two alternatives to perform the PIAC task. In all the proposed experiments, the approaches run on a particular set ($\{inst\}_{tr}$) that is randomly chosen for training and another one ($\{inst\}_{ts}$), randomly chosen for testing. Whenever it is applicable, a third set ($\{inst\}_{val}$) is chosen as validation to tune the hyper-parameters of the approaches.

4.5.1 Building and using the mapping h in MetaL PIAC

Aiming to build the mapping h for MetaL PIAC, we train the recommendation models using the meta-data that encompasses the problem features (Section 4.2) as inputs and the tuned configurations of the recommended MH (Sections 4.3 and 4.4) as outputs. As it might be quite useful to provide explainable models for algorithm and problem insights, we use the following tree-based models for the recommendation tasks:

- **Classification and Regression Tree (CART) (BREIMAN *et al.*, 1984):** simple binary decision trees built by greedily splitting the data using Gini index function for classification or squared error for regression. An important parameter is the tree complexity;
- **Random Forest (RF) (BREIMAN, 2001):** bagged ensemble of decision trees, where each split is made using the best variable of a random subset of features. Some parameters

are the number of trees and size of the random feature subsets;

- **eXtreme Gradient Boosting (XGB) (FRIEDMAN, 2001):** boosting ensemble of trees that uses splitting functions based on gradient values derived from a wide range of objective functions. An improved and fast implementation is named as the XGBoost algorithm proposed by Chen and Guestrin (2016). XGB has many parameters like number of iterations, objective function, learning rate, etc.

CARTs are explainable as simple decision trees, and RFs and XGBs present 'feature importance' metrics based on the mean decreased impurity of nodes. Another advantage of these models is that they are capable of solving classification and regression problems, used for categorical and numeric parameter recommendations, respectively.

4.5.1.1 Evaluating MetaL PIAC from a machine learning perspective

For all tests performed for MetaL PIAC, the model hyper-parameters are the ones shown in Table 6, which have been defined based on a 10-fold cross-validation performed on the training set. The final parameters are chosen as the best average macro-AUC.

Table 6 – Learning model parameters.

Model	Fixed parameters	Tuned parameters
CART		Complexity $\in \{0.0005, 0.001, 0.005, 0.01, 0.05\}$
RF	Number of trees $= 300$	Number of features per split $\in \{6, 13\}$
XGB	$\eta = 0.3$, $\gamma = 0$, 50% features sampled	Max. depth $\in \{3,6,12\}$
	75% sub-sampling, 300 rounds	

Source: The author.

In classification problems, AUC and average prediction metrics are used. The AUC metric is useful because it can measure the amount of information the model expresses (values of 0.5 or lower for single-class models) (BRADLEY, 1997). For regression problems, the metric used is the Mean Absolute Error (MAE). The MAE is a simple mean of the absolute prediction error, similar to the Root Mean Squared Error but not as sensible to outliers.

Training and validation phases correspond, respectively to building the meta-data to further train the machine-learning models and tuning the models hyper-parameters of tree-base and multi-label models based on a 10-fold cross-validation. During the testing phase, data not used during training and validation phases, is used to test the recommendation: presents it at the built model and get the output with the recommended MH or its configuration. For the MH

recommendation on MetaL PIAC, we test the 17 multi-label strategies described in Section 2.3. All strategies use CART, RF and XGB as the underlying learning models.

4.5.2 Building and using the mapping h in MOAAC/D

The MOAAC/D proposal involves finding good configurations on a multi-objective formulation of the AAC problem. First it decomposes the problem space P into different partition subsets. Then it solves the multi-objective problem of multiple partitions by providing, for each algorithm, specialist and compromise configurations within a Pareto set. Further it evaluates how similar is a new problem p' to each objective argument, i.e., to each partition subset, and chooses the configuration whose compromise with each objective is most similar with the compromises estimated for the test problem p'.

In the experiments, we test four different scenarios based on different decomposition possibilities of the FSP problems. As detailed in the next sections, in the first three scenarios a manual partition decomposes the problem space aiming to evaluate different strategies in terms of AAC performance. The last scenario is used to test the performance of the iMOEA/D algorithm performing the PIAC task.

4.5.2.1 Evaluating MOAAC/D from an AAC perspective

The first three scenarios used to evaluate the multi-objective proposed approach as a simple AAC approach, consider the following alternatives to decompose the problem space:

- a) P decomposed by means of FSP **objective function type** into two partition subsets ($\rho = 1,2$), one for each objective: makespan and flowtime. All problems have 50 jobs;
- b) P decomposed by means of **correlation** into two partition subsets ($\rho = 1,2$), one for instances with job- or machine-correlated processing times, and another for instances with uncorrelated processing times. All problems have 50 jobs and makespan objective;
- c) P decomposed by means of **instance size** into three partition subsets ($\rho = 1,2,3$) defined by the number of jobs: small ($J = 10$), medium ($J = 30$) and large ($J = 50$). All problems have makespan objective;

In all these scenarios, we use a reduced set of problems, only including random and

exponential distributions, $budget$ set as high, the stopping criterion $stopC$ set as time-based. This is done to focus on the particular features of the manual partitions. Including all instances might induce different configurations that maximize performance (for different budgets for example), also, it would require more computational time to evaluate the objective functions.

As shown in Section 5.2, in the context of AAC, we compare four different approaches. Two of them are based on irace, (i) the original irace algorithm used to provide a generalist configuration (G-IRACE), and (ii) Original irace algorithm optimizing a particular Extreme of the Pareto front (E-IRACE), and two are the variants of the proposed approach running with and without irace as local search:

- G-IRACE: single-objective AAC tuner (original irace algorithm) with default hyper-parameters;
- E-IRACE: single-objective AAC tuner (original irace algorithm), but it runs individually on each instance set I_p (to solve only its associated objective u_p), i.e., one whole execution for each $I_1, \ldots, I_m$ with the total budget N_{evals} divided by the number of objectives m;
- MOEA/D: the MOAAC/D formulation solved using the MOEA/D algorithm with default genetic operators (all parameters are considered as real-valued);
- iMOEA/D: the proposed MOEA/D with irace as local search operator on the MOAAC/D formulation.

In additional to the scalarization function set as g^{tche}, the remaining hyper-parameters established for iMOEA/D are shown in Table 7. These parameter values have been set based on the original MOEA/D (ZHANG; LI, 2007).

Table 7 – iMOEA/D parameters.

Scenario	$\|P\|$	N_{obj}	N_{pop}	S	$N_{samples}$	N_{neigh}	N_{evals}	N_{gen}		Total
							iMOEA/D	iMOEA/D	MOEA/D	no. tests
By objective	8	2	8	4					206	52800
By correlation	8	2	8	4	4	2	100	50	206	52800
By size	12	3	16	2					260	99200

Source: The author.

Notice that the first four hyper-parameters are scenario-dependent, including the total number of seeds S used to calculate the utility value for each candidate configuration in Figure 10 (lines 4 and 5). N_{evals} and N_{gen} are MH-dependent, with $N_{evals} = 100$ for iMOEA/D. Since this hyper-parameter defines the total tests performed by irace running as local search it is not

applicable for MOEA/D. $N_{samples}$ and N_{neigh} are set as 4 and 2, respectively, for all scenarios and all algorithms. The value of N_{gen} for MOEA/D is set to guarantee a fair comparison between its performance and that of iMOEA/D; MOEA/D evolves for more generations, since it does not use irace local search. In the last column, the total number of configurations tests performed by MOEA/D and iMOEA/D is given by $N_{pop}N_{gen}(N_{evals} + (|P|\, S))$.

In the experiments, irace baselines (G-IRACE and E-IRACE) run with default hyper-parameters and the same maximum number of tests. The irace hyper-parameters are chosen to provide reasonable computational costs (small population and few generations), while still providing good demonstration of the proposed approaches (robust performance with $N_{samples} \geq 2$ and $N_{evals} = 100$ evaluations for irace local search in iMOEA/D).

Finally, for all the three AAC scenarios previously described, the final configurations of every comparison approach is evaluated with $S = 100$ on each instance p. This should give a better accuracy to validate the performance of each baseline and the proposed approach iMOEA/D. The final Pareto fronts $\mathcal{PF}$ produced by each approach (G-IRACE, E-IRACE, MOEA/D and iMOEA/D) are evaluated in terms of dominance and performance on each group of instances.

4.5.2.2 Evaluating MOAAC/D from a PIAC Perspective

In Section 3.3 we describe how the MOAAC/D formulation can be used to solve the PIAC problem, given a membership function $mshp : P \rightarrow \boldsymbol{\psi} \in [0,1]^{N_{obj}}$ indicating the compromise of each problem p with each objective u_p. We also discussed how the feature space can be used to automatically decompose the problem space. In these experiments, we use the proposed iMOEA/D as the strategy for solving the MOAAC/D formulation during the PIAC training phase. It is chosen as its irace-base local search gives good results compared to the MOEA/D with genetic operators during the evaluation from the AAC perspective.

In this fourth and last scenario considered for the MOAAC/D proposal, P is automatically decomposed into four partition subsets ($\rho = 1,2,3,4$), one for each principal component of the PCA model. For this, we perform a space reduction in the pre-processed F data by means of PCA, providing $\Re^4$ feature space. Notice that the choice of four dimensions seems a good compromise between not degenerating iMOEA/D performance and also allowing a suitable partition in the problem space. It is known that the number of objectives affects the performance of MOEAs (DEB; JAIN, 2012) since they require exponentially larger populations to cover

the whole objective space. In this automated partitioning approach, the number of objectives is considered a parameter and the aspects of MOEAs' performance, quality of the decomposition (like PCA's percentage of variance explained), and the expected amount of compromise configurations could be investigated in future works.

After building a PCA model to reduce the feature space to four dimensions we select the problems that best represent each of the first four axis. The metric cos^2 is commonly used with PCA to identify such inputs. Therefore the membership function $mshp$ is defined as:

$$mshp(p) = \boldsymbol{\psi}_p = \cos^2(PCAprediction(f(p))) \tag{18}$$

In the experiments, the $mshp$ defines the problem partition subsets P_1, P_2, P_3, and P_4 by selecting the top 100 problems that have the highest $mshp$ values for each of the four dimensions.

The approach chosen to perform the MOAAC/D task in this last scenario is iMOEA/D. It is tuned using validation data to find a good approximation of the Pareto-Set. The iMOEA/D parameters are: $N_{pop} = 20$, $N_{gen} = 400$, $N_{eval} = 400$ and $N_{neigh} = 2$. These configurations use roughly the same number of tests performed for building MetaL PIAC performance space.

Training and validation phases correspond, respectively to performing the multi-objective optimization and setting the hyper-parameters of iMOEA/D by means of irace. During the testing phase, data not used during training or testing is used to recommend configuration for an unseen problem as:

$$\begin{aligned} h^*(f(p')|\mathcal{PS}) &= \arg\min_{\boldsymbol{\theta}\in\mathcal{PS}} g^{tche}(\mathbf{u}(\boldsymbol{\theta})|mshp(f(p')),\vartheta^*) \\ &= \arg\min_{\boldsymbol{\theta}\in\mathcal{PS}} g^{tche}(\mathbf{u}(\boldsymbol{\theta})|\cos^2(PCAprediction(f(p'))),\vartheta^*) \end{aligned} \tag{19}$$

where $g^{tche}(.)$ is the Tchebycheff aggregation function (Equation 4) and ϑ^* is the reference point, obtained from the Pareto-Front as the best utility for each partition subset.

4.6 COMPARING PIAC APPROACHES: AN OPTIMIZATION PERSPECTIVE

The previous sections have described the experiments designed to evaluate individually the MetaL PIAC in machine learning perspective, MOAAC/D in AAC perspective and its adaptation to PIAC. Next, we discuss an experiment focused on PIAC comparison that tests the configuration recommendations. The experiments consider MetaL PIAC, MOAAC/D and

baseline methods performing the PIAC task. This section aims at describing the last experiments designed to compare different PIAC approaches.

The process of training and testing the proposed PIAC approaches is based on a 5-fold cross-validation procedure. Every recommendation model obtained from both proposals is trained using the meta-data, which is built using the training set $\{inst\}_{tr}$ and joins the feature space with the performance values. The results are evaluated on the test set $\{inst\}_{ts}$ using the ARPD metric (Equation 17) considering the best known solution $g(\mathbf{x}_p^*)$ as the ones found by irace while building MetaL PIAC's performance space. Therefore, for all strategies compared for the PIAC task under an optimization perspective, the same training problems are used and the final comparison depends on the performance on all problems from the test set. For this, we use the final PIAC recommendation schemes learned from the 5-fold cross validation and evaluate its recommendation quality when solving (optimizing) problems from the test set. Aiming to reduce uncertainty, we evaluate the final per-instance configurations with $S = 30$ samples.

For a fair comparison, all PIAC strategies when tested on unseen instances from the test set use roughly the same number of configurations tests $1000|\{inst_{ts}\}|$. Since no previous work on AAC uses the same flowshop problem dataset, we compare the proposed recommendation models with two recommendation strategies:

- a baseline approach that randomly chooses MH and its configuration - Recommends a random configuration (randPIAC);
- an overall best solver G-IRACE achieved by the original irace, used to find the best MH and generalist configurations at once on all problems in the training set. It uses the same configurations evaluations budget ($1000\times$ the number of problems) as MetaL PIAC and iMOEA/D. The MH is an additional parameter for tuning, and there are dependencies between all parameters and their respective MH;

The best solver configuration found by G-IRACE running on the overall dataset is an IG with LSPS using random NEH initialization, random best hill-climbing local search, 63% of the neighborhood, single-step hill-climbing LSPS and acceptance criterion of only improving solutions. As expected, this configuration is similar to the state-of-the-art for flowshop problems (Dubois-Lacoste *et al.*, 2017). The best solvers for every single external fold have very similar configurations; therefore the best solver obtained by G-IRACE is simply determined over all problems.

In addition, we test two different tie-breaking strategies whenever statistical test used in the comparison phase of MetaL PIAC: random choice and Performance-Priority tie-breaking between recommended MHs (PP). In a PP strategy, MH with lowest parameters average training errors is chosen as the recommended MH. Besides the CART, RF, and XGB multi-label BR recommendation models, we also consider each model equipped with Feature selection (FS).

5 RESULTS

This chapter presents the results obtained by both proposals. First we present the results of MetaL PIAC, under a machine-learning perspective, i.e., considering the regression problem of ranking algorithms and classification/regression problem of recommending the algorithm's configuration depending on the parameter type. Then we present the results of MOAAC/D proposal running only as an AAC approach, i.e., it does not perform the PIAC task. Finally we compare the two PIAC strategies proposed in the present work, with one irace-based baseline and a random tuner.

5.1 METAL PIAC: RESULTS FROM A MACHINE-LEARNING PERSPECTIVE

The results for MetaL PIAC are presented in two parts: MH recommendation and parameter recommendation.

5.1.1 Metaheuristic recommendation

As described in the previous chapter, the meta-data for MH recommendation is composed of problem features (problem-based and FLA based features) and the recommended MH for each problem or instance. An MH is considered recommended if it is statistically superior to others or equivalent to the MH with the best mean final fitness value. Therefore, it is possible that multiple MHs are recommended at once, making it a multi-label classification problem. The evaluation is made using the multi-label metric macro-AUC and average precision.

Figure 16 shows the comparison between the performance of different multi-label strategies (from Ranking by Pairwise Comparison (RPC) to the Binary Relevance (BR) - see Section 2.3 for more details on these strategies) for the three addressed machine-learning algorithms (CART, RF, XGB) using or not the information gain feature selection procedure. Most of approaches present good performance for average-precision, and CART seems less effective than the others for macro-AUC. Regarding the machine-learning techniques, CART models are inferior, as expected for simple models, and RF and XGB have similar performance.

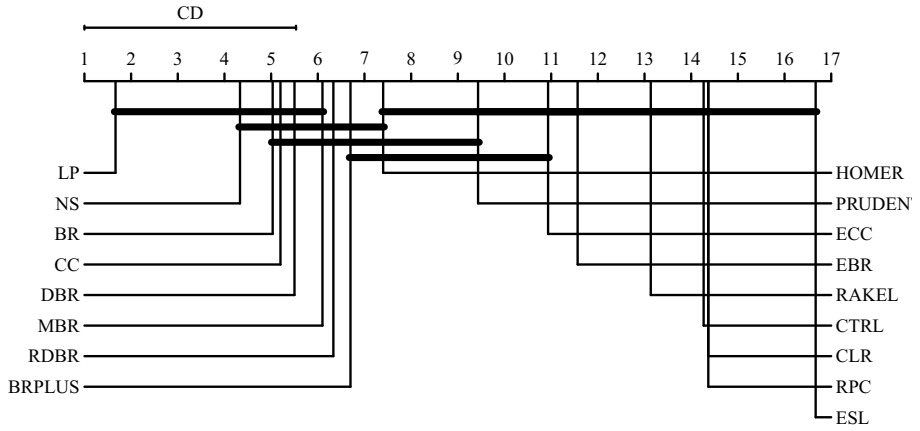
The boxplots shown in blue and red in Figure 16 represent metrics with and without feature selection, respectively. We can see no clear advantage of using feature selection. In

Figure 16 – Multi-label MH recommendation models performance for average precision (left) and AUC (right) for CART (top) , RF (middle) and XGB (bottom).

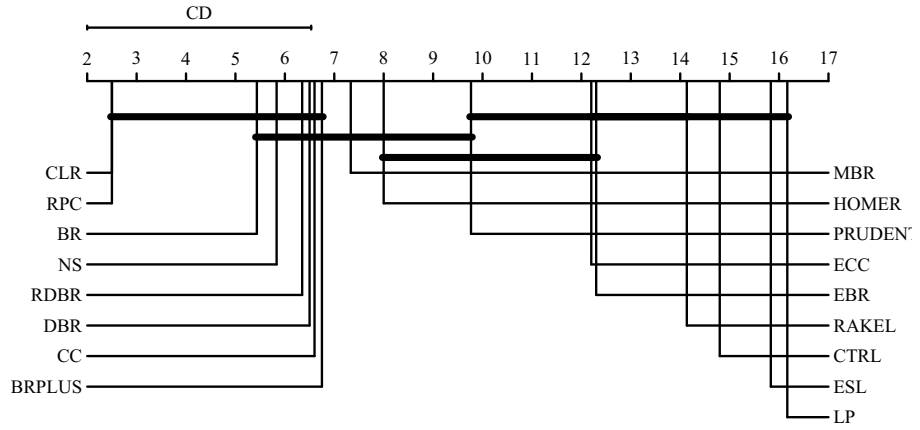
Source: The author.

some cases like macro-AUC of RF and Recursive Dependent Binary Relevance (RDBR) the feature selection is better. On the other hand, the feature selection presents worse results when we use LP on the same metric and base learner. Some of the features that are not included in feature selection are redundant metrics, like the percentage of local optima nodes (zero in most cases) and random walk correlation with multiple steps (highly correlated with the single step

Figure 17 – Critical difference plot of the Friedman post-hoc Nemeny test comparing different multi-label algorithms for MH recommendation.



(a) Macro-AUC.



(b) Average-precision.

Source: The author.

autocorrelation).

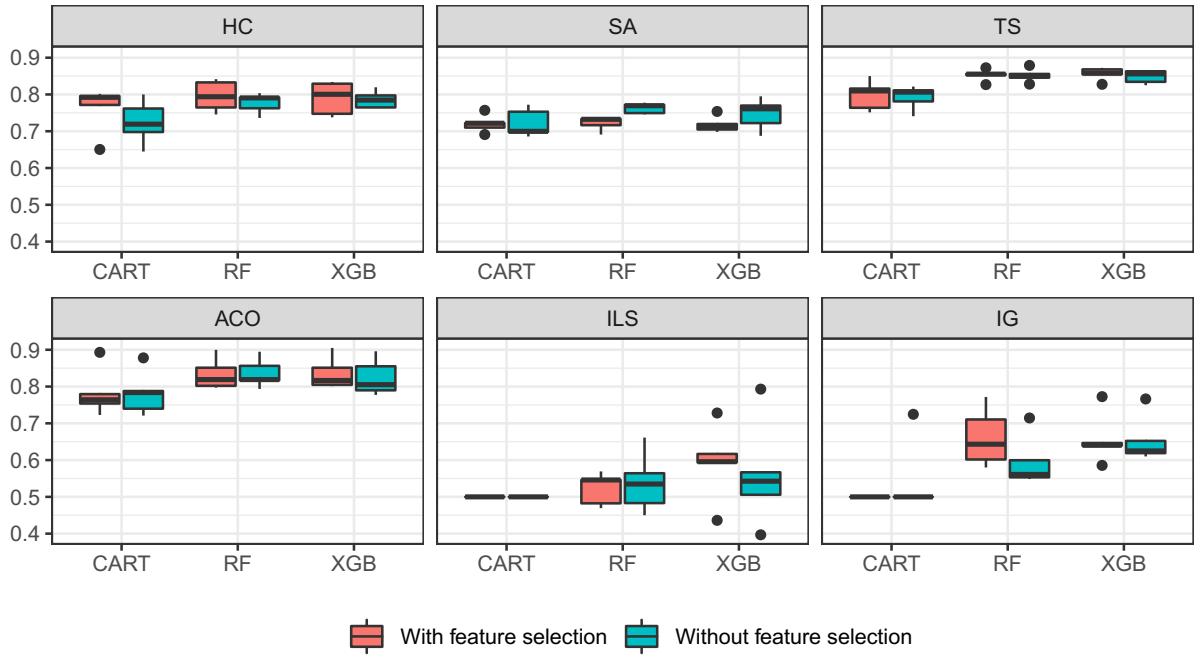
The main performance differences are between multi-label strategies. For example, the label powerset (LP) has good performance for macro-AUC but it is inferior regarding the average precision. Ensemble of Single Label (ELS) seems the worst of macro-AUC and is among the worst for precision.

We use the Friedman test to compare the different strategies. The critical difference plots on Figure 17 allow us to conclude that Binary Relevance (BR) strategies are as good as the best strategy for both metrics.

Figure 18 shows the performance of the binary relevance models for each possible label (recommended MH).

For the HC and SA recommendation, the models show similar performance (AUC is 0.75 ± 0.05). Models recommending TS and ACO have better recommendation performance

Figure 18 – Performance for each MH recommendation model using binary relevance multi-label strategy.



Source: The author.

(AUC is 0.82 ± 0.04). As ILS and IG algorithms are mostly recommended for the majority of instances, the training data is imbalanced and models are inferior in terms of AUC. For those, CART models have a single node, recommending ILS and IG for all instances.

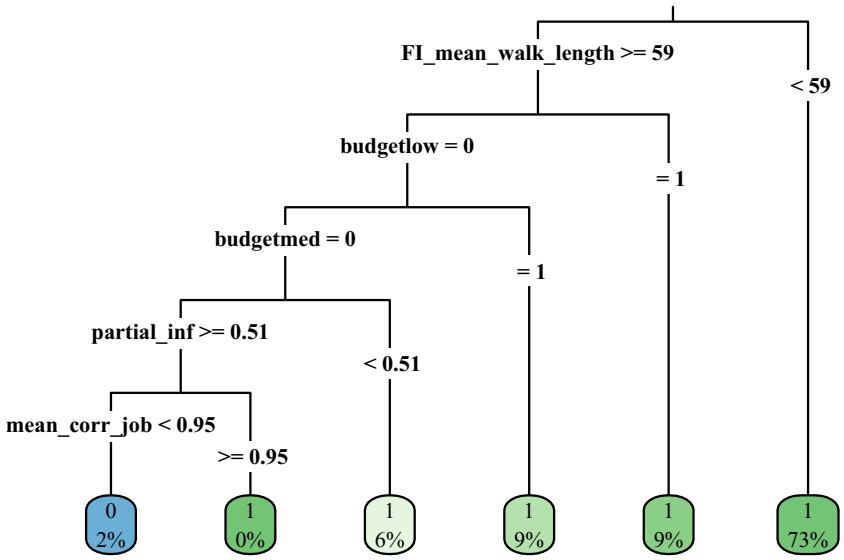
Regarding the model's interpretation, Figure 19 shows a CART model for TS recommendation and Figure 20 summarizes the model importance for all BR base RF models.

The most important features are the budget, stopping criterion, and the mean number of steps of an FI adaptive walk. The objective, instance size, and solution type frequencies have little contributions for the MH recommendation models. Also, since ILS and IG are recommended most of the time, the features are not very informative, and the decrease in node impurity is small.

5.1.2 Parameter recommendation

For each parameter, the CART, RF, and XGB models are trained for classification or regression tasks, according to the parameter domain. Since there is a model for each parameter, the full configuration is built by joining the predictions of all models. The same methodology for MH recommendation of 5 external folds and 10 internal folds for parameter setting is used for each parameter and model combination.

Figure 19 – CART model for TS recommendation.



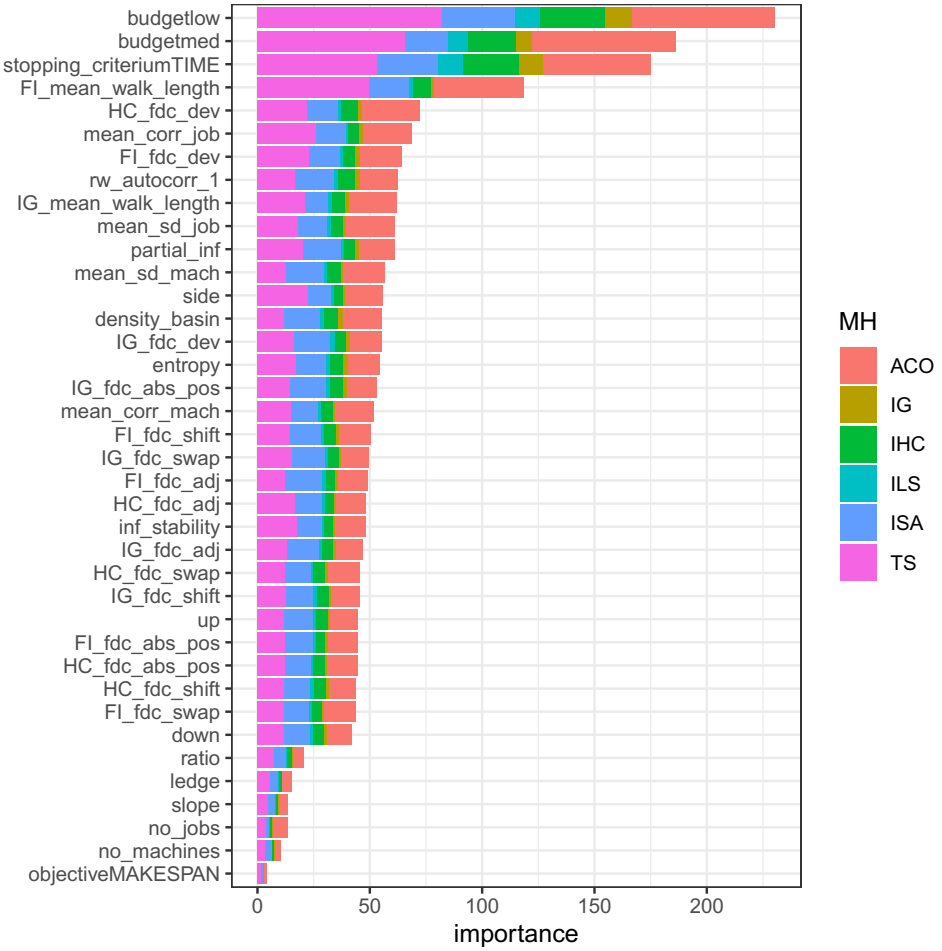
Source: The author.

Table 8 shows the results for categorical parameters in terms of the AUC metric. The generalized multi-class AUC proposed by Hand and Till (2001) evaluates parameters with multiple categories. For most HC parameters, SA initialization and cooling schedules, ACO single-step local search, ILS local search and neighborhood strategy, and IG acceptance and single-step local search, the performance was acceptable with most models presenting AUC higher than 0.5. Some parameters like IG perturbation have poor performance.

The performance of numerical parameters are shown in Table 9 in terms of MAE. For better comparison, the values are normalized between $[0,1]$. With some exceptions like HC and ACO neighborhood size and SA temperature decay, the error is around 0.3, which indicates high variation with respect to the best parameter values.

From the machine-learning perspective, we conclude that the proposed models induce h mappings capable of providing good recommendations for MH and their categorical parameters. However, for numerical parameters, the models induce mappings with high levels of regression errors. For example, the ILS parameter 'number of kicks' has above 30% variance in all models. This means that either the models are not well trained or the parameter recommendation is not suitable given the provided problem features. In Section 5.3, we present the results obtained with the testing configurations under optimization perspectives.

Figure 20 – RF models feature importance for each MH recommendation.



Source: The author.

5.2 MOAAC/D: RESULTS FROM AAC AND PIAC PERSPECTIVES

The results for the second proposal are presented in this section, considering first the three scenarios described in Section 4.5.2 and then the last scenario described in Section 4.5.2.2.

5.2.1 Results for Scenario a): flowshop instances decomposed by FSP objective

This scenario is particularly interesting considering that configurations with average performance between both objectives can be used in multi-objective formulations of flowshop that consider makespan and flowtime simultaneously (YENISEY; YAGMAHAN, 2014).

Figure 21 shows the final configurations performance with instances decomposed by objective. The original MOEA/D only finds low quality configurations. E-IRACE finds configurations with good utilities for makespan objective. The irace baseline finds configurations

Table 8 – Categorical parameter recommendation performance: AUC for each model with or without feature selection.

	No feature selection			Feature selection		
Parameter	CART	RF	XGB	CART	RF	XGB
HC local search (FI,HC,Random HC,IG)	0.625	0.669	0.668	0.631	0.658	0.650
HC comparison operator ($>$, $\geq$)	0.705	0.785	0.755	0.749	0.780	0.750
HC initialization (random,NEH,random NEH)	0.553	0.624	0.618	0.549	0.607	0.602
HC neighborhood strategy (ordered,random)	0.657	0.718	0.708	0.687	0.707	0.705
SA comparison operator ($>$, $\geq$)	0.508	0.493	0.488	0.508	0.493	0.488
SA initialization (random,NEH,random NEH)	0.609	0.680	0.665	0.626	0.677	0.666
SA cooling schedule (simple,dynamic,OsmanPotts)	0.693	0.751	0.740	0.692	0.698	0.676
TS aspiration (none,better)	0.481	0.507	0.506	0.495	0.491	0.488
TS comparison operator ($>$, $\geq$)	0.634	0.662	0.641	0.640	0.657	0.637
TS initialization (random,NEH,random NEH)	0.562	0.621	0.609	0.571	0.597	0.602
TS local search (FI,HC,HC random)	0.593	0.650	0.629	0.586	0.653	0.633
TS neighborhood strategy (ordered,random)	0.659	0.735	0.725	0.684	0.728	0.749
TS taboo list (index,random index,solution,neighbor)	0.494	0.525	0.518	0.494	0.523	0.523
ACO comparison operator ($>$, $\geq$)	0.601	0.631	0.619	0.628	0.624	0.634
ACO initialization (random,NEH,random NEH)	0.514	0.556	0.545	0.519	0.539	0.526
ACO neighborhood strategy (ordered,random)	0.611	0.657	0.651	0.631	0.647	0.638
ACO single-step local search (yes,no)	0.775	0.838	0.814	0.772	0.839	0.819
ILS acceptance (always,better,temperature)	0.595	0.660	0.640	0.632	0.643	0.647
ILS comparison operator ($>$, $\geq$)	0.528	0.567	0.555	0.541	0.558	0.563
ILS initialization (random,NEH,random NEH)	0.556	0.615	0.601	0.572	0.600	0.594
ILS local search (FI,HC,Random HC or IG)	0.600	0.664	0.625	0.613	0.663	0.624
ILS neighborhood strategy (ordered,random)	0.607	0.627	0.614	0.605	0.611	0.595
ILS neutral escape strategy (restart,destruction,kick)	0.427	0.521	0.532	0.429	0.519	0.526
ILS perturb (restart,kick,NILS)	0.596	0.654	0.638	0.605	0.654	0.635
ILS restart strategy (random,NEH,random NEH)	0.504	0.572	0.570	0.518	0.578	0.577
ILS single-step local search (yes,no)	0.599	0.633	0.617	0.634	0.639	0.629
IG acceptance (always,better,temperature)	0.559	0.646	0.625	0.605	0.631	0.631
IG perturbation (RMS, LSPS)	0.494	0.445	0.445	0.501	0.474	0.467
IG comparison operator ($>$, $\geq$)	0.565	0.593	0.567	0.551	0.577	0.577
IG initialization (random,NEH,random NEH)	0.517	0.557	0.545	0.560	0.559	0.558
IG local search (FI,HC,Random HC,IG)	0.571	0.672	0.658	0.575	0.672	0.658
IG neighborhood strategy (ordered,random)	0.562	0.567	0.556	0.583	0.566	0.563
IG partial solution local search (FI,HC,Random HC,IG)	0.418	0.504	0.507	0.490	0.499	0.513
IG single-step local search (yes,no)	0.592	0.636	0.617	0.609	0.623	0.611

Source: The author.

with good compromise between objectives, while iMOEA/D mostly focus on the flowtime objective.

Some examples of configurations found are:

- Best for flowtime found by iMOEA/D: best-insertion local search, with destruction-construction with LSPS using single-step best improvement local search and $d = 7$, and accepting only improvements;
- Best for makespan found by E-IRACE: best-insertion local search on 73% of the neighborhood, with a single swap as perturbation, and acceptance criterion based on temperature with very low factor of $T = 0.015$;

Table 9 – Numerical parameter recommendation performance: MAE for each model with or without feature selection, normalized to each parameter range.

	No feature selection			Feature selection		
Parameter	CART	RF	XGB	CART	RF	XGB
HC neighborhood size (0-10%,...,90-100%)	0.195	0.179	0.217	0.186	0.184	0.200
SA initial temperature	0.283	0.247	0.284	0.259	0.255	0.279
SA neighborhood size (0-10%,...,90-100%)	0.204	0.193	0.232	0.202	0.194	0.229
SA final temperature (0,...,1)	0.247	0.228	0.248	0.239	0.231	0.257
SA initial temperature factor (0.1,...,10)	0.284	0.241	0.273	0.272	0.244	0.267
SA max. moves in span (50,...,500)	0.256	0.234	0.264	0.260	0.253	0.299
SA max. number of span moves (n/a,2,...,10)	0.345	0.311	0.360	0.300	0.315	0.316
SA max. trie in span (1000,...,7000)	0.299	0.253	0.282	0.271	0.266	0.283
SA Osman and Potts beta (0,...,1)	0.287	0.249	0.280	0.252	0.252	0.268
SA simple schedule span size (50,100)	0.318	0.270	0.316	0.276	0.274	0.345
SA temperature decay (0.1,...,1)	0.243	0.216	0.238	0.239	0.214	0.236
TS max. steps a move is taboo (1,...,10)	0.299	0.276	0.352	0.283	0.294	0.321
TS neighborhood size (0-10%,...,90-100%)	0.161	0.146	0.187	0.154	0.149	0.163
TS random taboo size (n/a,1,...,8)	0.331	0.282	0.320	0.305	0.291	0.337
TS taboo list maximum size (2,...,10)	0.317	0.279	0.315	0.305	0.276	0.310
TS max. moves without improvement	0.270	0.244	0.301	0.264	0.255	0.274
ACO initial pheromone level factor	0.282	0.261	0.325	0.256	0.264	0.265
ACO evaporation rate (0.01,...,1)	0.271	0.252	0.315	0.248	0.256	0.259
ACO neighborhood size (0-10%,...,90-100%)	0.239	0.220	0.271	0.230	0.228	0.247
ACO prob. of choosing best path (0,...,1)	0.274	0.252	0.305	0.256	0.251	0.275
ILS accept. temperature factor (0,...,5)	0.285	0.234	0.263	0.247	0.245	0.272
ILS kick perturb strength factor (0*N,...,1*N)	0.282	0.255	0.287	0.268	0.260	0.283
ILS neighborhood size (0-10%,...,90-100%)	0.264	0.234	0.280	0.262	0.235	0.285
ILS neutral number of kicks (n/a,1,2,3)	0.416	0.376	0.389	0.435	0.380	0.375
ILS neutral perturb destruction size (n/a,2,...,6)	0.316	0.289	0.334	0.298	0.294	0.334
ILS neutral perturb maximum number of steps (0,...,1000)	0.291	0.262	0.287	0.276	0.277	0.302
ILS number of kicks (n/a,0-0.1*N,...,0.9-0.1*N)	0.398	0.358	0.412	0.382	0.367	0.409
ILS restart perturb. patience (0,...,10)	0.325	0.298	0.317	0.318	0.310	0.358
IG accept. temperature factor (0,...,5)	0.304	0.256	0.270	0.299	0.269	0.313
IG destruction size (0*N,...,1*N)	0.241	0.218	0.267	0.230	0.222	0.261
IG neighborhood size (0-10%,...,90-100%)	0.266	0.242	0.295	0.256	0.246	0.292

Source: The author.

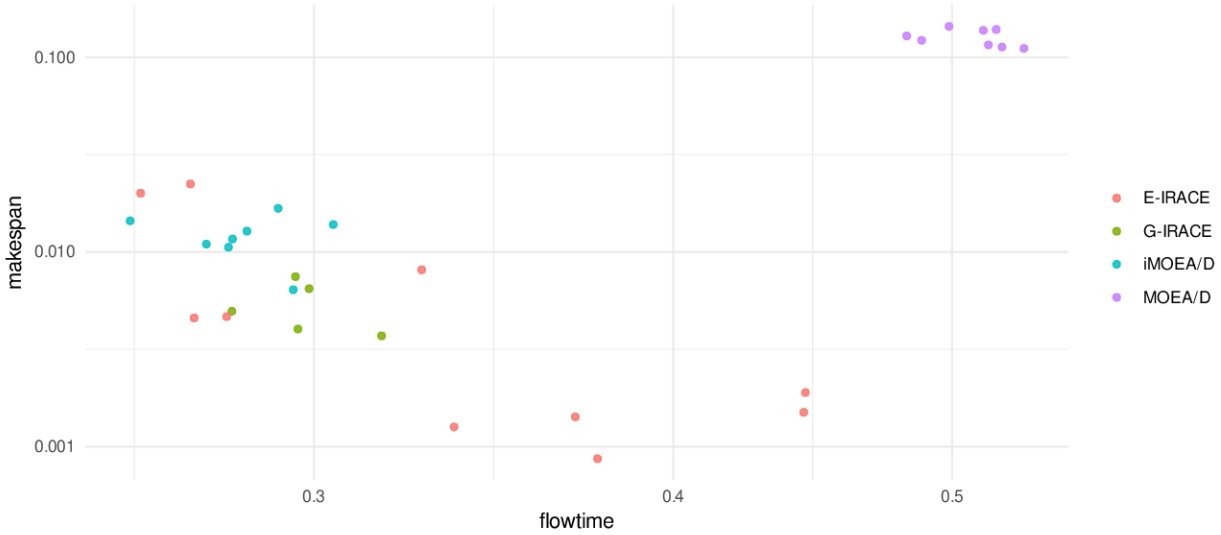
- Compromise solution by G-IRACE: similar to the state-of-the-art IG with LSPS configuration with $d = 4$ and accepting only improvements.

5.2.2 Results for Scenario b): flowshop instances decomposed by processing time correlation

Correlated instances are known to be easy to solve using simple hill-climbing methods (WATSON *et al.*, 1999). For this reason, most configurations that include a local search on the full neighborhood can perform well.

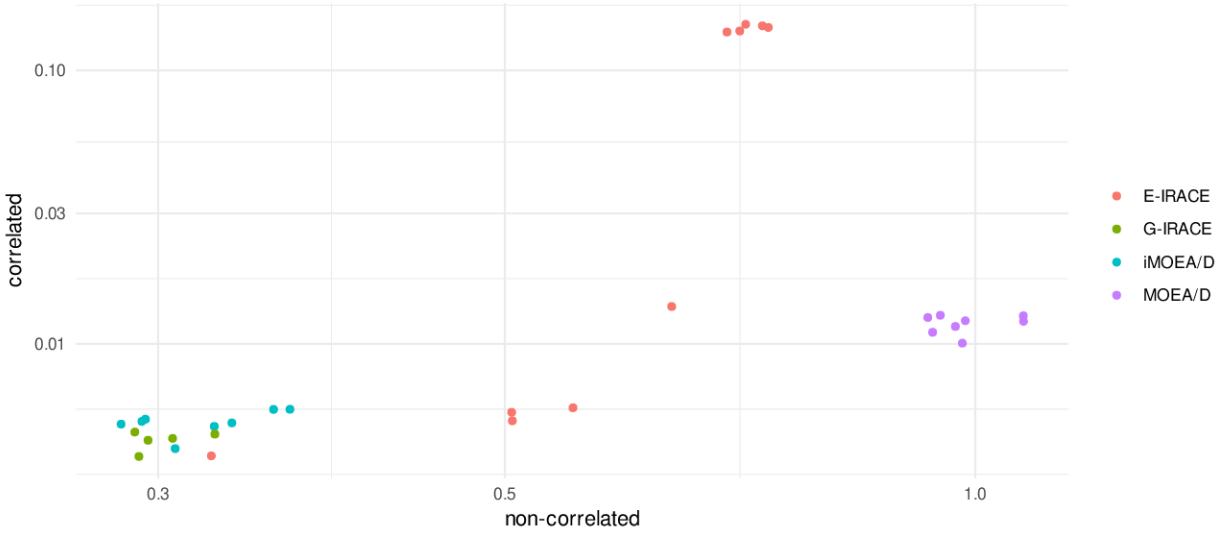
Figure 22 shows the configurations performance with instances decomposed by processing times correlation. We notice that irace and iMOEA/D are able to find good configurations for correlated processing times. The best configurations for each objective are very similar to the

Figure 21 – Final IG configurations performance on the flowshop problem whose instances are decomposed by objective functions makespan and flowtime, both assessed through ARPD values.



Source: The author.

Figure 22 – Final IG configurations performance on the flowshop problem decomposed by processing times correlation (correlated or not-correlated) given by ARPD values.



Source: The author.

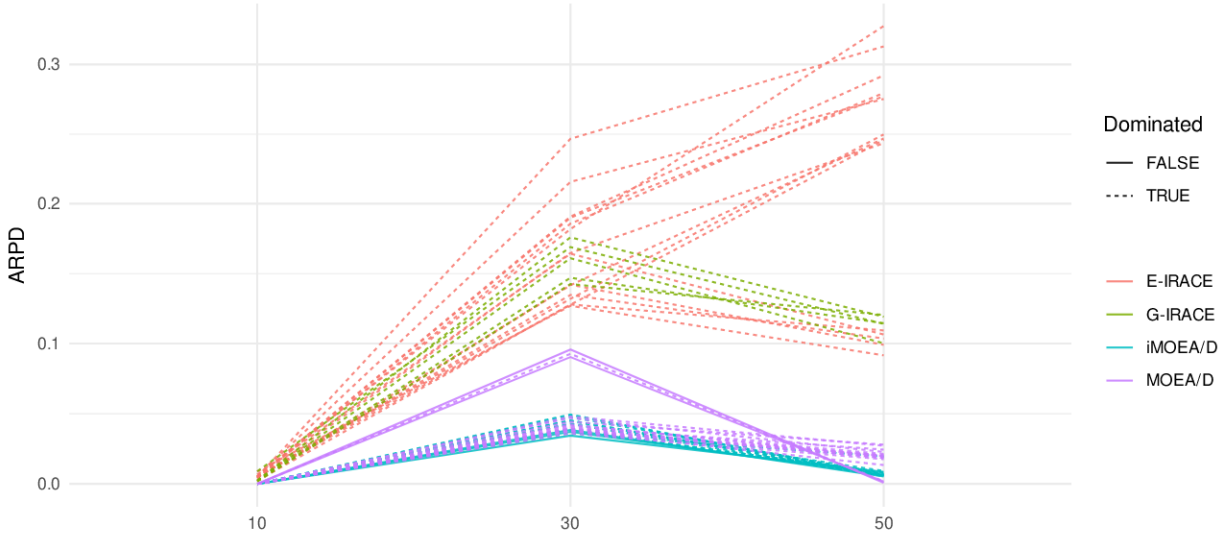
state-of-the-art IG with partial solution local search and the only difference is that correlated instances have higher destruction sizes:

- Best for result for correlated instances by iMOEA/D: best-insertion local search, destruction-construction perturbation with LSPS using best-insertion local search, destruction size $d = 4$, and acceptance criterion of only improvements;
- Best for result for non-correlated instances by G-IRACE: same configuration as above, but with $d = 5$.

5.2.3 Results for Scenario c): flowshop instances decomposed by FSP size

The number of jobs is the feature that most affect the heuristics performance considering random uniform instances (REEVES, 1999). For these experiments, the partition define three sizes: small ($J = 10$), medium ($J = 30$) and large ($J = 50$).

Figure 23 – Final IG configurations performance on the flowshop problem decomposed by size (10, 30 or 50 jobs) given by ARPD values.



Source: The author.

Figure 23 shows the configurations validation performance with instances decomposed by size. We see that all configurations reach the best known utility on small ($J = 10$) instances. iMOEA/D finds good configurations for medium ($J = 30$) and MOEA/D for large ($J = 50$) problems. Only iMOEA/D and MOEA/D configurations are non-dominated. Some of the configurations are:

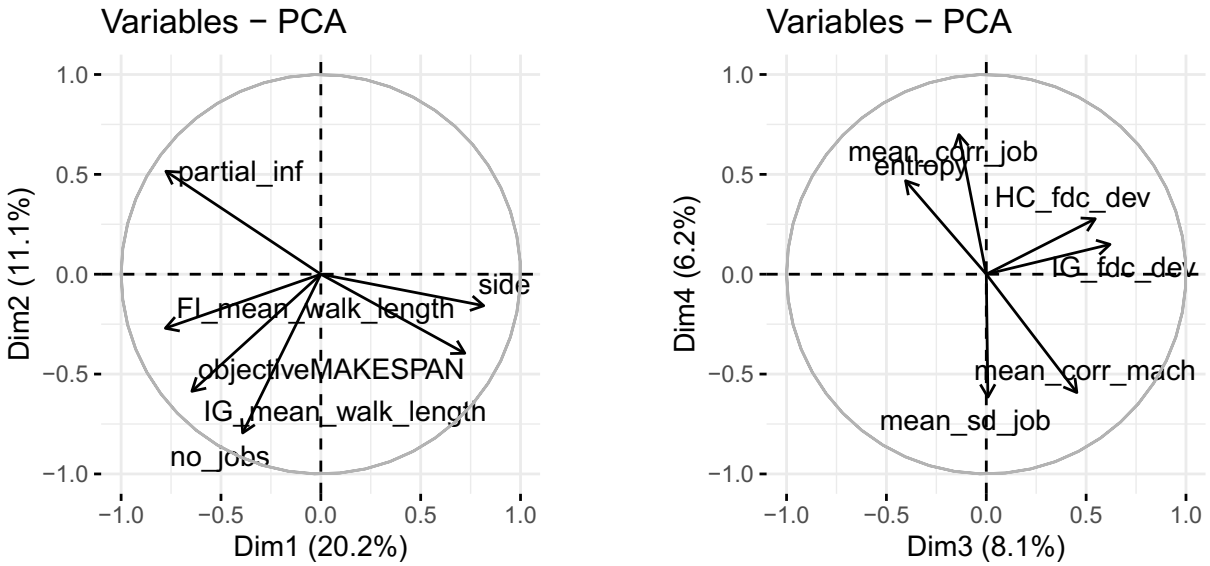
- Best for $J = 30$ found by iMOEA/D: best-insertion local search, with destruction-construction perturbation using $d = 8$, and acceptance criterion of only improvements;
- Best for $J = 50$ found by MOEA/D: no local-search, with destruction-construction perturbation using $d = 4$, and acceptance criterion based on temperature with very low factor of $T = 8e - 6$;
- Compromise solutions found by iMOEA/D: best-insertion local search with swap perturbation using 3 swaps, and accepting only improvements.

Overall, iMOEA/D is able to find configurations that are close to the best for each objective, as well as compromise configurations. The irace-extremes formulation is better in cases like

partition by objective, where there might not be compromises, only specialist configurations that share little information. This might also be the reason why MOEA/D with genetic operators does not perform well on objective scenario, since the genetic operators mostly generate individuals that share similarities with their parents.

5.2.4 Results for scenario d): automatic problem space decomposition by principal components

Figure 24 – PCA variable plot for flowshop features that most contribute on each axis. For better visualization, we only show the top six features.



Source: The author.

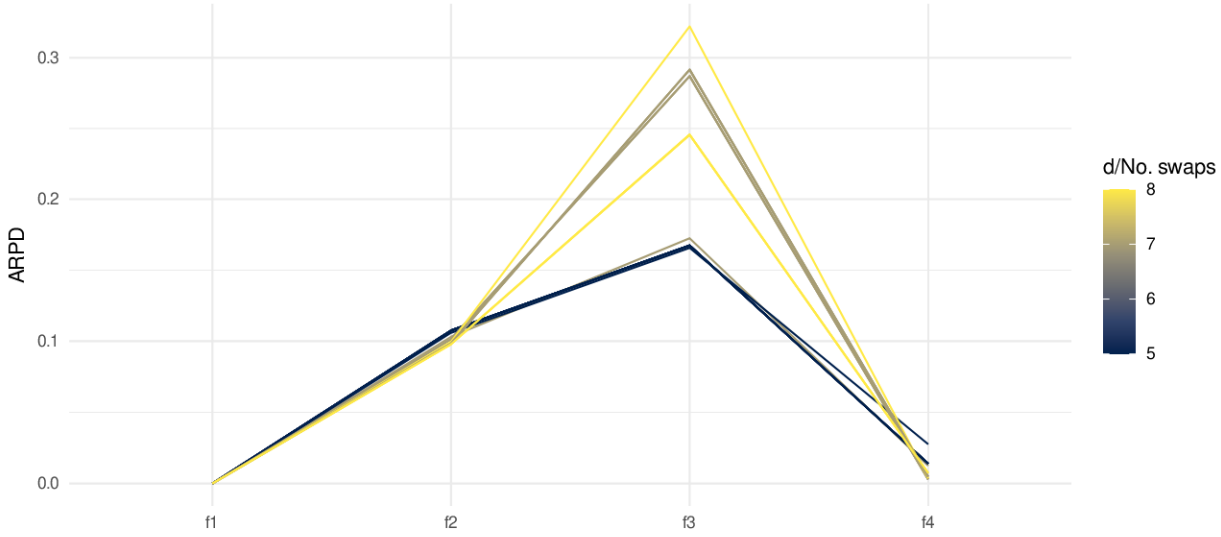
As described in Section 4.5.2.2, we can use a PCA model of the feature space to automatically decompose the problem space. PCA models provide the contribution measure for the problem features in relation to each component. Figure 24 shows the top six features that most contribute for the dimensionality reduction. In summary, the partitioning explains 45,6% of the data variability and follows:

- Dimension 1: instances with low number of jobs and makespan objective;
- Dimension 2: instances with many neutral (side) edges and low number of steps to reach local optima using first improvement;
- Dimension 3: job-correlated instances with high entropy on random walk fitness;

- Dimension 4: instances with high correlation between fitness and number of steps to reach local minimum (fitness-distance correlation) using best insertion and best improvement local searches.

Figure 25 shows the parallel coordinates plot for the final 20 configurations found by iMOEA/D on the decomposed flowshop problem space.

Figure 25 – Final IG configurations performance on the four dimensions PCA of problem features.



Source: The author.

We notice that all configurations found the best know results on instances of the first axis, and the third axis is possibly the hardest, with configurations that require low number of destructions.

5.3 COMPARING PIAC STRATEGIES: RESULTS FROM AN OPTIMIZATION PERSPECTIVE

Since the optimal solutions for all considered problems are not known we compare the final mean objective values to the mean objective of the best MH and configuration for each instance, i.e., the problem performance meta-data. Table 10 shows the percentage of configurations that are worse, equal or better than the instance-based best configuration from the dataset. For a small percentage, the recommendations achieved better performance and most configurations are equal or worse. In all tables presented in this section, we highlight the best results, and further discuss them in the text.

Table 10 – Percentage of better, equal, worse and better or equal configurations compared to the instance-based optimal configuration and MH.

Strategy	Better	Equal	Worse	Better or equal
randPIAC	0.59	30.14	69.27	30.73
G-IRACE	3.30	54.49	42.21	57.79
MetaL PIAC – CART	6.48	51.02	42.50	57.50
MetaL PIAC – RF	5.74	51.90	42.36	57.64
MetaL PIAC – XGB	5.52	49.68	44.80	55.20
MetaL PIAC – CART-FS	6.08	51.91	42.01	57.99
MetaL PIAC – RF-FS	5.26	50.88	43.86	56.14
MetaL PIAC – XGB-FS	5.31	49.27	45.42	54.58
MetaL PIAC – CART-PP	5.52	50.46	44.01	55.99
MetaL PIAC – RF-PP	6.94	53.75	**39.31**	60.69
MetaL PIAC – XGB-PP	4.92	48.95	46.13	53.87
MetaL PIAC – CART-FS-PP	6.36	52.04	41.60	58.40
MetaL PIAC – RF-FS-PP	6.56	52.31	41.13	58.87
MetaL PIAC – XGB-FS-PP	5.00	49.01	45.99	54.01
iMOEA/D	**15.56**	**60.65**	23.80	**76.20**

Source: The author.

Overall, the configurations found by iMOEA/D outperform all other strategies, reaching 76.20% configurations with better or equal performance, compared to the instance-based best configuration. MetaL PIAC's RF-PP model has the least worse configurations (39.31%), and 60.69% of the recommendations are better than or equal to the instance-base best.

Interestingly, even recommending algorithms that are simpler than the state-of-the-art IG, the PIAC models are more capable of producing configurations that are better than the instance-based optimal, for around 6% for MetaL PIAC models and 15.56% for iMOEA/D, considering all problems.

Table 11 shows the same percentages for instances with $J = 50$ jobs. For these harder instances, the MetaL PIAC models are mostly better than the overall best solver provided by G-IRACE: all percentages of better configurations of the first are higher than the later. Moreover, all percentages of worse configurations in MetaL PIAC are lower than the ones in G-IRACE. In these harder instances, iMOEA/D has the higher amount configurations that are better than the baseline (27.79). For equal, worse and better or equal, MetaL PIAC performs the best (37.28%, 51.17% and 48.83%, respectively). This difference might be due to the PCA model that running over the problem features in iMOEA/D focuses on only one axis (axis $f3$ in Figure 25) for probably harder instances (higher values of ARPD).

Figure 26 shows the critical difference comparison of the best models, MetaL PIAC RF-PP and iMOEA/D, with the random and best-solver baseline strategies.

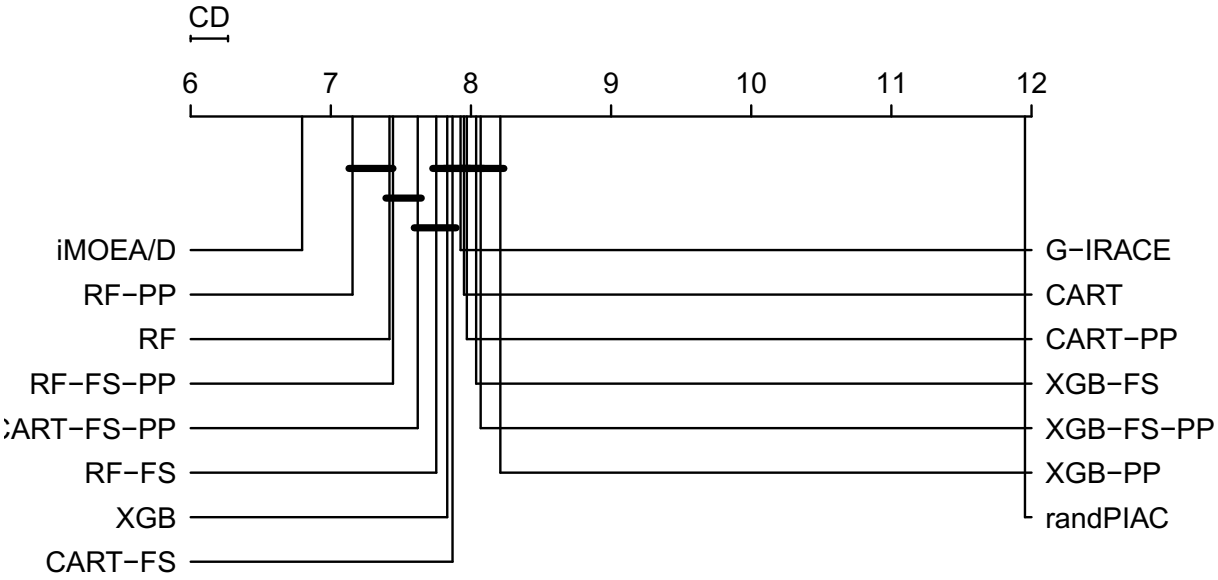
The plot shows the statistic values for the Friedman test with Nemenyi post-hoc and

Table 11 – Percentage of better, equal, worse and better or equal configurations compared to the instance-based optimal configuration and MH considering only instances with $J = 50$ jobs.

Strategy	Better	Equal	Worse	Better or equal
randPIAC	0.37	17.16	82.47	17.53
G-IRACE	3.95	35.86	60.19	39.81
MetaL PIAC – CART	10.80	35.99	53.21	46.79
MetaL PIAC – RF	8.89	35.62	55.49	44.51
MetaL PIAC – XGB	8.64	34.32	57.03	42.96
MetaL PIAC – CART-FS	9.81	37.04	53.15	46.85
MetaL PIAC – RF-FS	9.63	35.74	54.63	45.37
MetaL PIAC – XGB-FS	10.19	33.77	56.05	43.95
MetaL PIAC – CART-PP	8.83	34.75	56.42	43.58
MetaL PIAC – RF-PP	11.54	**37.28**	**51.17**	**48.83**
MetaL PIAC – XGB-PP	7.22	34.14	58.64	41.36
MetaL PIAC – CART-FS-PP	10.56	37.10	52.35	47.65
MetaL PIAC – RF-FS-PP	11.73	36.48	51.79	48.21
MetaL PIAC – XGB-FS-PP	9.81	33.58	56.60	43.40
iMOEA/D	**27.79**	10.19	62.04	37.96

Source: The author.

Figure 26 – Friedman test critical difference plot of relative improvement of recommendation strategies.



Source: The author.

the critical difference considering $\alpha = 0.05$ significance level. We can see that the iMOEA/D outperforms all strategies and RF-based models can mostly outperform the best solver and other models.

The values for the ARPD metric are shown for each problem J in Table 12 and for each problem correlation type in Table 13. Comparing the overall best solver provided by G-IRACE to the PIAC strategies, we see that the MetaL PIAC models are the best on $J = 50$ instances, while iMOEA/D have good performance on $J = 10$ and $J = 30$. The ARPD by processing time correlation shows that iMOEA/D performs better on instances with correlated processing

times, while RF models from MetaL PIAC have good recommendations on harder uncorrelated instances, and G-IRACE is better on job-correlated instances.

Table 12 – ARPD over the instance-based best configurations for different number of jobs.

Strategy	10	20	30	50
randPIAC	1.8621467	2.7769920	2.6985613	3.5113785
G-IRACE	0.0504838	**0.0707775**	0.0759169	0.1289456
MetaL PIAC – CART	0.2039080	0.4312832	0.1451264	0.0973404
MetaL PIAC – RF	0.1420477	0.1265833	0.2641105	**0.0649518**
MetaL PIAC – XGB	0.1284261	0.1518446	0.1933849	0.1186370
MetaL PIAC – CART-FS	0.2585760	0.1255681	0.1042376	0.0966631
MetaL PIAC – RF-FS	0.2065405	0.1414928	0.1851762	0.0928052
MetaL PIAC – XGB-FS	0.9328087	0.3294801	0.3945023	0.1112641
MetaL PIAC – CART-PP	0.2008428	0.1955303	0.1308527	0.1090830
MetaL PIAC – RF-PP	0.0449009	0.1598861	0.0935169	0.0675427
MetaL PIAC – XGB-PP	0.2859672	0.3372254	0.2436894	0.1257882
MetaL PIAC – CART-FS-PP	0.5580679	0.2423539	0.1055191	0.0877328
MetaL PIAC – RF-FS-PP	0.1542107	0.0881221	0.1037375	0.0747267
MetaL PIAC – XGB-FS-PP	0.9031691	0.3266154	0.3924017	0.1077634
iMOEA/D	**0.0332770**	0.0970083	**0.0718594**	0.1396014

Source: The author.

Table 13 – ARPD over the instance-based best configurations for different correlation types.

Strategy	job-correlated	machine-correlated	non-correlated
randPIAC	4.1018945	0.2717023	3.7632120
G-IRACE	**0.0121109**	0.0061273	0.2263547
MetaL PIAC – CART	0.4049182	0.0261983	0.2271270
MetaL PIAC – RF	0.2512856	0.0143181	0.1826663
MetaL PIAC – XGB	0.1721288	0.0336721	0.2384186
MetaL PIAC – CART-FS	0.0972662	0.0325028	0.3090146
MetaL PIAC – RF-FS	0.1646906	0.0298418	0.2749786
MetaL PIAC – XGB-FS	0.8856861	0.0488105	0.3915448
MetaL PIAC – CART-PP	0.1936907	0.0138164	0.2697246
MetaL PIAC – RF-PP	0.1149538	0.0134066	**0.1460247**
MetaL PIAC – XGB-PP	0.4533118	0.0231233	0.2680676
MetaL PIAC – CART-FS-PP	0.4876789	0.0176097	0.2399667
MetaL PIAC – RF-FS-PP	0.1038594	0.0148917	0.1968467
MetaL PIAC – XGB-FS-PP	0.8629502	0.0490871	0.3854249
iMOEA/D	0.0171043	**-0.0006360**	0.2398413

Source: The author.

Overall, we observe that there is a compromise between learning the models of recommendation with MetaL PIAC and exploring the compromise of distinct problem partition subsets with iMOEA/D. The iMOEA/D is better in general, probably because the problem space includes many small instances, where it performs better. However, the MetaL PIAC models are better for hard problems. This could be explained by the fact that features are more useful for defining the best operators and algorithms when the search space is large. Therefore it is able to recommend specialist configurations where otherwise iMOEA/D would recommend generalist ones because of the limited number of partitions.

6 CONCLUSIONS AND FUTURE WORK

This work has formalized, discussed and analyzed two proposals for the Per-Instance Algorithm Configuration (PIAC) problem. One namely MetaL PIAC is based on an extension to Rice's meta-learning model, and other is inspired on the Multi-objective Automatic Algorithm Configuration based on problem space Decomposition (MOAAC/D). The strategies have been tested on the flowshop scheduling problem using more than $6{,}000$ instances. The problem's features include simple attributes and fitness landscape analysis metrics.

MetaL PIAC has been evaluated for recommending six metaheuristics (HC, SA, TS, ACO, ILS, and IG) and their parameters using tree-based models (CART, RF, and XGB). The results from a machine learning perspective showed that the models are able to predict the best metaheuristic using multi-label binary relevance classifiers. The parameters recommendation, specially real-valued parameters, like ILS number of kicks, have large variance. Some possible reasons for this are:

- a single parameter value dominates, leading to an unbalanced dataset;
- the MH performance is insensitive to the parameter value, and the output is close to random;
- parameter interactions can influence MH performance.

MOAAC/D proposal is evaluated in the task of finding good configurations for a hybrid MH encompassing ILS and IG algorithms in different problem space partition types: by objective, by correlation and by size. The final configurations are compared with single-objective AAC like G-IRACE and E-IRACE, focusing on generalist and specialist configurations for each partition subset. The results have shown that the proposed iMOEA/D, i.e. MOEA/D with or without irace as local search, performs well finding non-dominated configurations in most scenarios.

Disadvantages of the MetaL PIAC strategy:

Number of configuration evaluations spent on easy instances: the MetaL PIAC has to build a database of the best configurations for several problems in order to build its models (for this work we use 1000 configuration evaluations for each problem). Easy instances, like the ones with correlated processing times, must be solved during the training phase

with the same budget as harder instances. Ideally, the search could focus on regions of the problem space where more sophisticated configurations and better tuning of the parameters are necessary;

Equivalent configurations: as one might expect, there may be several ways to solve a given optimization problems that yield the same performance. For the MetaL PIAC, this means that the performance dataset can contain noise from the best configurations search process. This imposes difficulties in choosing the appropriate models on the training phase;

Interpretability: the models produced during the training phase might not be easily interpretable. Tree-based models like CART do not produce the best results. Also, since there is a separate model for each parameter, the understating of why a full configurations is chosen requires aggregating several interpretations involving different problem features.

Multi-label model choice: the result have shown that the impact of multi-label models on the MetaL PIAC performance is higher for multi-label models than for inner models. As there are several options, the problem of selecting such model can be harder than recommending the configuration.

Advantages of the MetaL PIAC strategy:

Good performance on harder instances: the results show that the MetaL PIAC models are better on harder instances, like the ones with larger sizes and uncorrelated processing times. This could be due to the fact that the performance data on harder instances is more reliable, with well defined parameters;

Reusability: the parameter recommendation models could also be used to, for example, automatically adapt the configuration during the search. By getting features online, the adapt could check in the meta-data which is the better configuration and updates it. Another application is to use it as a guide to understating instances features and how they affect the performance on certain algorithms, operators or parameter values.

MOAAC/D PIAC disadvantages:

Problem space partition: the practitioner should know previously how to partition the problem space. Alternatively, one can use the problem features and possibly a dimensionality

reduction technique to automatically find the problem space partition. Nonetheless, it introduces complexity, specially when the problem is not well known;

Search space limitations: since testing configurations is usually a computationally expensive process, the number of individuals and generations should be small. Therefore, the multi-objective process might not be able to explore large configuration spaces;

MOAAC/D PIAC advantages:

Compromise configurations: as the strategy includes a Pareto-set of good configurations, the practitioner can choose the most adequate according to different criteria;

Flexibility: since the space of problems is partitioned, it is possible to model different features or combinations of features and directly search for specialist and generalist configurations.

The final comparison involves testing the configuration models in terms of the objective function value against a random recommendation and an overall best-solver provided by G-IRACE. The overall best-solver is a state-of-the-art IG with parameters tuned for all instances in the problem dataset. The results show that both proposed approaches are advantageous, specially MetaL PIAC for larger instances and iMOEA/D overall.

Regarding the hypothesis, we can empirically conclude that the mono-objective approach based on meta-learning and the multi-objective approach based on problem space decomposition can provide good configurations even for unseen instances. However, this conclusion is particular for the flowshop scenarios addressed in the present work. Moreover, it was not possible to identify whether the advantages and drawbacks of each proposal were due to mono/multi formulation or classification-regression/optimization modeling.

6.1 FUTURE WORK

The current work can be improved in many directions:

1. **Expand meta-data:** including different flowshop variants, as well as generating harder instances (VALLADA *et al.*, 2015);
2. **Improve meta-features:** studying the cost of LON meta-features (HERNANDO *et al.*, 2017), specially for harder problems, when the use of costly meta-features is acceptable, or else reduce its computation cost;

3. **Alternative utility metrics:** using different cost measures like empirical mean run time (HOOS; STÜTZLE, 2004) to compare algorithms while building the performance data;

4. **Use and compare with other AACs** both proposals use irace as main component, other competitive AACs, like SMAC and ParamILS (HUTTER *et al.*, 2014), could yield better results;

5. **Use and compare with other MOEAs**, the multi-objective proposal (MOAAC/D) is implemented by means of the MOEA/D framework. Other state-of-the-art MOEAs and even Many-objective Optimization Algorithms (MaOEAs) - algorithms developed to deal with 4 or more objectives - could be considered;

6. **Explore generated models and configurations:** MetaL PIAC models and iMOEA/D final Pareto-set can be interpreted and matched against the problem features could lead to insights on the base problem;

7. **Scale to larger problems**: further investigations on how to scale the MetaL PIAC framework to address real0world instances could include sampling small sets of large problems and using few-shot learners (WANG *et al.*, 2020) for efficient recommendation with less meta-data.

8. **Explore the parameters of MOAAC/D with automated partitioning**: we consider the number of objectives as a parameter and it could be further investigated. This is an important parameter, since it influences the quality of the configurations with better partitions, but also aspects of many-objective MOEAs' performance. Also, we do not explore the percentage of data variation provided by PCA models, as well as alternative clustering methods.

In the big picture, the present work points toward the increasing use of problem information for automatically solve optimization problems. We think that there is a large room for improvements and PIAC could, in the future, be used in several complex real-world problems, where there are many complexities and costs involved. Moreover, we can also see a hybridization of both proposed approaches with MetaL PIAC providing initial solutions for iMOEA/D.

REFERENCES

Alfaro-Fernández, Pedro; RUIZ, Rubén; PAGNOZZI, Federico; STÜTZLE, Thomas. Automatic Algorithm Design for Hybrid Flowshop Scheduling Problems. **European Journal of Operational Research**, v. 282, n. 3, p. 835–845, May 2020. ISSN 0377-2217.

ALLAHVERDI, Ali; NG, C. T.; CHENG, T. C. E.; KOVALYOV, Mikhail Y. A survey of scheduling problems with setup times or costs. **European Journal of Operational Research**, v. 187, n. 3, p. 985–1032, 2008. ISSN 0377-2217.

BAKER, Kenneth R.; TRIETSCH, Dan. Appendix A: Practical Processing Time Distributions. *In*: **Principles of Sequencing and Scheduling**. *[S.l.]*: John Wiley & Sons, Ltd, 2009. p. 445–458. ISBN 978-0-470-45179-3.

Ben-Daya, M.; Al-Fawzan, M. A tabu search approach for the flow shop scheduling problem. **European Journal of Operational Research**, v. 109, n. 1, p. 88–95, 1998. ISSN 0377-2217.

BIRATTARI, Mauro. **Tuning Metaheuristics: A Machine Learning Perspective**. First. *[S.l.]*: Springer-Verlag Berlin Heidelberg, 2009. (Studies in Computational Intelligence 197). ISBN 3-642-00482-2 978-3-642-00482-7 978-3-642-00483-4.

BISCHL, Bernd; KERSCHKE, Pascal; KOTTHOFF, Lars; LINDAUER, Marius; MALITSKY, Yuri; FRÉCHETTE, Alexandre; HOOS, Holger; HUTTER, Frank; Leyton-Brown, Kevin; TIERNEY, Kevin; VANSCHOREN, Joaquin. ASlib: A benchmark library for algorithm selection. **Artificial Intelligence**, v. 237, p. 41–58, Aug. 2016. ISSN 00043702.

BISCHL, Bernd; MERSMANN, Olaf; TRAUTMANN, Heike; PREUSS, Mike. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. *In*: **Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation**. New York, NY, USA: ACM, 2012. p. 313–320.

BISKUP, Dirk. A state-of-the-art review on scheduling with learning effects. **European Journal of Operational Research**, v. 188, n. 2, p. 315–329, 2008. ISSN 0377-2217.

BLOT, Aymeric; HOOS, Holger H.; JOURDAN, Laetitia; Kessaci-Marmion, Marie-Éléonore; TRAUTMANN, Heike. MO-ParamILS: A Multi-objective Automatic Algorithm Configuration Framework. *In*: FESTA, Paola; SELLMANN, Meinolf; VANSCHOREN, Joaquin (Ed.). **Learning and Intelligent Optimization**. Cham: Springer International Publishing, 2016. v. 10079, p. 32–47. ISBN 978-3-319-50348-6 978-3-319-50349-3.

BONNEY, M. C.; GUNDRY, S. W. Solutions to the Constrained Flowshop Sequencing Problem. **Journal of the Operational Research Society**, v. 27, n. 4, p. 869–883, 1976. ISSN 1476-9360.

BOSSEK, Jakob; KERSCHKE, Pascal; TRAUTMANN, Heike. A multi-objective perspective on performance assessment and automated selection of single-objective optimization algorithms. **Applied Soft Computing**, v. 88, p. 105901, Mar. 2020. ISSN 15684946.

BRADLEY, Andrew P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. **Pattern recognition**, Elsevier, v. 30, n. 7, p. 1145–1159, 1997.

BRAZDIL, Pavel; CARRIER, Christophe Giraud; SOARES, Carlos; VILALTA, Ricardo. **Metalearning: Applications to Data Mining**. Berlin, Heidelberg: Springer Science & Business Media, 2008.

BREIMAN, Leo. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct. 2001. ISSN 1573-0565.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and Regression Trees**. Monterey, CA: Wadsworth and Brooks, 1984.

BURKE, Edmund K.; HYDE, Matthew R.; KENDALL, Graham; OCHOA, Gabriela; ÖZCAN, Ender; WOODWARD, John R. A Classification of Hyper-Heuristic Approaches: Revisited. *In*: GENDREAU, Michel; POTVIN, Jean-Yves (Ed.). **Handbook of Metaheuristics**. Cham: Springer International Publishing, 2019. v. 272, p. 453–477. ISBN 978-3-319-91085-7 978-3-319-91086-4.

CAMPBELL, Herbert G.; DUDEK, Richard A.; SMITH, Milton L. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. **Management Science**, INFORMS, v. 16, n. 10, p. B630–B637, 1970. ISSN 00251909, 15265501.

CHEN, R-M; SANDNES, F. E. Flow shop scheduling based on a novel cooling temperature driven simulated annealing algorithm. **Scientia Iranica.Transaction B, Mechanical Engineering**, v. 22, n. 4, p. 1545–1553, 2015.

CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. *In*: **Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2.

CHICANO, Francisco; WHITLEY, L. Darrell; ALBA, Enrique. A Methodology to Find the Elementary Landscape Decomposition of Combinatorial Optimization Problems. **Evolutionary Computation**, v. 19, n. 4, p. 597–637, 2011.

CONOVER, W.J. **Practical Nonparametric Statistics**. Third. New York: John Wiley & Sons, 1999.

CZOGALLA, Jens; FINK, Andreas. Fitness landscape analysis for the no-wait flow-shop scheduling problem. **Journal of Heuristics**, v. 18, n. 1, p. 25–51, Feb. 2012. ISSN 1572-9397.

DANNENBRING, David G. An Evaluation of Flow Shop Sequencing Heuristics. **Management Science**, v. 23, n. 11, p. 1174–1182, 1977.

DANTAS, A. L.; POZO, A. T. R. A Meta-Learning Algorithm Selection Approach for the Quadratic Assignment Problem. *In*: **2018 IEEE Congress on Evolutionary Computation (CEC)**. *[S.l.: s.n.]*, 2018. p. 1–8.

DEB, Kalyanmoy. Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. *In*: WANG, Lihui; NG, Amos H. C.; DEB, Kalyanmoy (Ed.). **Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing**. London: Springer London, 2011. p. 3–34. ISBN 978-0-85729-617-7 978-0-85729-652-8.

DEB, K.; JAIN, H. Handling many-objective problems using an improved NSGA-II procedure. *In*: **2012 IEEE Congress on Evolutionary Computation (CEC)**. *[S.l.: s.n.]*, 2012. p. 1–8.

DONG, Xingye; HUANG, Houkuan; CHEN, Ping. An improved NEH-based heuristic for the permutation flowshop problem. **Computers & Operations Research**, v. 35, n. 12, p. 3962–3968, 2008. ISSN 0305-0548.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: Optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 26, n. 1, p. 29–41, Feb. 1996. ISSN 1083-4419.

DRÉO, Johann. Using performance fronts for parameter setting of stochastic metaheuristics. *In*: **Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference - GECCO '09**. Montreal, Québec, Canada: ACM Press, 2009. p. 2197. ISBN 978-1-60558-505-5.

Dubois-Lacoste, Jérémie; PAGNOZZI, Federico; STÜTZLE, Thomas. An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. **Computers & Operations Research**, v. 81, p. 160–166, 2017. ISSN 0305-0548.

DYMOND, Antoine S.; ENGELBRECHT, Andries P.; KOK, Schalk; HEYNS, P. Stephan. Tuning Optimization Algorithms Under Multiple Objective Function Evaluation Budgets. **IEEE Transactions on Evolutionary Computation**, v. 19, n. 3, p. 341–358, Jun. 2015. ISSN 1089-778X, 1089-778X, 1941-0026.

DYMOND, Antoine S.; KOK, Schalk; HEYNS, P. Stephan. The sensitivity of multi-objective optimization algorithm performance to objective function evaluation budgets. *In*: **2013 IEEE Congress on Evolutionary Computation**. Cancun, Mexico: IEEE, 2013. p. 1868–1875. ISBN 978-1-4799-0454-9 978-1-4799-0453-2 978-1-4799-0451-8 978-1-4799-0452-5.

DYMOND, Antoine S.; KOK, Schalk; HEYNS, P. Stephan. MOTA: A Many-Objective Tuning Algorithm Specialized for Tuning under Multiple Objective Function Evaluation Budgets. **Evolutionary Computation**, v. 25, n. 1, p. 113–141, Mar. 2017. ISSN 1063-6560, 1530-9304.

EMMONS, George Vairaktarakis (auth.) Hamilton. **Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications**. First. *[S.l.]*: Springer US, 2013. (International Series in Operations Research & Management Science 182). ISBN 978-1-4614-5151-8 978-1-4614-5152-5.

ENGELBRECHT, Andries P. **Computational Intelligence: An Introduction**. Second. *[S.l.]*: Wiley Publishing, 2007. ISBN 0-470-03561-7.

FEO, Thomas A.; RESENDE, Mauricio G. C. Greedy Randomized Adaptive Search Procedures. **Journal of Global Optimization**, v. 6, n. 2, p. 109–133, 1995. ISSN 1573-2916.

Fernandez-Viagas, Victor; FRAMINAN, Jose M. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. **Computers & Operations Research**, v. 45, p. 60–67, 2014. ISSN 0305-0548.

Fernandez-Viagas, Victor; RUIZ, Rubén; FRAMINAN, Jose M. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. **European Journal of Operational Research**, v. 257, n. 3, p. 707–721, 2017. ISSN 0377-2217.

FEURER, Matthias; SPRINGENBERG, Jost Tobias; HUTTER, Frank. Initializing Bayesian Hyperparameter Optimization via Meta-learning. *In*: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. *[S.l.]*: AAAI Press, 2015. (AAAI'15), p. 1128–1135. ISBN 0-262-51129-0.

FLETCHER, R. **Practical Methods of Optimization**. *[S.l.]*: Wiley, 2013. ISBN 978-1-118-72318-0.

FRIEDMAN, Jerome H. Greedy function approximation: A gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001.

GAREY, M. R.; JOHNSON, D. S.; SETHI, Ravi. The Complexity of Flowshop and Jobshop Scheduling. **Mathematics of Operations Research**, INFORMS, v. 1, n. 2, p. 117–129, 1976. ISSN 0364765X, 15265471.

GIBAJA, Eva; VENTURA, Sebastián. A Tutorial on Multilabel Learning. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 47, n. 3, p. 52:1–52:38, Apr. 2015. ISSN 0360-0300.

GLOVER, Fred. Tabu Search—Part I. **ORSA Journal on Computing**, v. 1, n. 3, p. 190–206, 1989.

GOLDBERG, D.E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. *[S.l.]*: Addison-Wesley, 1989.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. Rinnooy. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *In*: HAMMER, E.L. Johnson P.L.; KORTE, B. H. (Ed.). **Discrete Optimization In Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium Co-Sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver**. *[S.l.]*: Elsevier, 1979, (Annals of Discrete Mathematics, v. 5). p. 287–326.

GREFENSTETTE, John. Optimization of Control Parameters for Genetic Algorithms. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 16, n. 1, p. 122–128, Jan. 1986. ISSN 0018-9472.

Gutierrez-Rodríguez, Andres E.; Conant-Pablos, Santiago E.; Ortiz-Bayliss, José C.; Terashima-Marín, Hugo. Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. **Expert Systems with Applications**, v. 118, p. 470–481, 2019. ISSN 0957-4174.

HAND, David J.; TILL, Robert J. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. **Machine Learning**, v. 45, n. 2, p. 171–186, Nov. 2001. ISSN 1573-0565.

HERNANDO, Leticia; DAOLIO, Fabio; VEERAPEN, Nadarajen; OCHOA, Gabriela. Local Optima Networks of the Permutation Flowshop Scheduling Problem: Makespan vs. total flow time. *In*: **2017 IEEE Congress on Evolutionary Computation (CEC)**. San Sebastian, Spain: IEEE, 2017. p. 1964–1971.

HOOKER, J. N. Needed: An Empirical Science of Algorithms. **Operations Research**, v. 42, n. 2, p. 201–212, Apr. 1994. ISSN 0030-364X, 1526-5463.

HOOKER, J. N. Testing heuristics: We have it all wrong. **Journal of Heuristics**, v. 1, n. 1, p. 33–42, Sep. 1995. ISSN 1381-1231, 1572-9397.

HOOS, Holger H; STÜTZLE, Thomas. **Stochastic Local Search: Foundations and Applications**. San Francisco, USA: Elsevier, 2004.

HUMEAU, Jérémie; LIEFOOGHE, Arnaud; TALBI, El-Ghazali; VEREL, Sébastien. **ParadisEO-MO: From Fitness Landscape Analysis to Efficient Local Search Algorithms**. *[S.l.]*, 2013.

HUTTER, Frank; López-Ibáñez, Manuel; FAWCETT, Chris; LINDAUER, Marius; HOOS, Holger H.; Leyton-Brown, Kevin; STÜTZLE, Thomas. AClib: A Benchmark Library for Algorithm Configuration. *In*: PARDALOS, Panos M.; RESENDE, Mauricio G.C.; VOGIATZIS, Chrysafis; WALTEROS, Jose L. (Ed.). **Learning and Intelligent Optimization**. Cham: Springer International Publishing, 2014. p. 36–40. ISBN 978-3-319-09584-4.

ISHIBUCHI, Hisao; MISAKI, Shinta; TANAKA, Hideo. Modified simulated annealing algorithms for the flow shop sequencing problem. **European Journal of Operational Research**, v. 81, n. 2, p. 388–398, 1995. ISSN 0377-2217.

JIN, Feng; SONG, Shiji; WU, Cheng. Structural Property and Meta-heuristic for the Flow Shop Scheduling Problem. *In*: CHAKRABORTY, Uday K. (Ed.). **Computational Intelligence in Flow Shop and Job Shop Scheduling**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–20. ISBN 978-3-642-02836-6.

JOHNSON, S. M. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 1, n. 1, p. 61–68, 1954. ISSN 1931-9193.

JONES, Terry *et al.* **Evolutionary Algorithms, Fitness Landscapes and Search**. 1995. Phd Thesis (PhD Thesis) — University of New Mexico, 1995.

KADIOGLU, Serdar; MALITSKY, Yuri; SELLMANN, Meinolf; TIERNEY, Kevin. ISAC –Instance-Specific Algorithm Configuration. *In*: **Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence**. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010. p. 751–756. ISBN 978-1-60750-605-8.

KALCZYNSKI, Pawel Jan; KAMBUROWSKI, Jerzy. On the {NEH} heuristic for minimizing the makespan in permutation flow shops. **Omega**, v. 35, n. 1, p. 53–60, 2007. ISSN 0305-0483.

KALCZYNSKI, Pawel J.; KAMBUROWSKI, Jerzy. An Improved NEH Heuristic to Minimize Makespan in Permutation Flow Shops. **Comput. Oper. Res.**, Elsevier Science Ltd., Oxford, UK, UK, v. 35, n. 9, p. 3001–3008, Sep. 2008. ISSN 0305-0548.

KALCZYNSKI, Pawel J.; KAMBUROWSKI, Jerzy. An empirical analysis of the optimality rate of flow shop heuristics. **European Journal of Operational Research**, v. 198, n. 1, p. 93–101, 2009. ISSN 0377-2217.

KANDA, Jorge; CARVALHO, Andre; HRUSCHKA, Eduardo; SOARES, Carlos. Selection of algorithms to solve traveling salesman problems using meta-learning. **International Journal of Hybrid Intelligent Systems**, IOS Press, v. 8, n. 3, p. 117–128, 2011.

KANDA, Jorge; CARVALHO, Andre de; HRUSCHKA, Eduardo; SOARES, Carlos; BRAZDIL, Pavel. Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features. **Neurocomputing**, v. 205, p. 393–406, 2016. ISSN 0925-2312.

KANDA, Jorge; SOARES, Carlos; HRUSCHKA, Eduardo; CARVALHO, Andre De. A meta-learning approach to select meta-heuristics for the traveling salesman problem using MLP-Based label ranking. *In*: **International Conference on Neural Information Processing**. Doha, Qatar: Springer, 2012. p. 488–495.

KERSCHKE, Pascal; HOOS, Holger H.; NEUMANN, Frank; TRAUTMANN, Heike. Automated Algorithm Selection: Survey and Perspectives. **Evolutionary Computation**, v. 27, n. 1, p. 3–45, 2019.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 0036-8075.

KOULAMAS, Christos. A new constructive heuristic for the flowshop scheduling problem. **European Journal of Operational Research**, v. 105, n. 1, p. 66–71, 1998. ISSN 0377-2217.

LAGEWEG, BJ; LENSTRA, Jan Karel; KAN, AHG Rinnooy. A general bounding scheme for the permutation flow-shop problem. **Operations Research**, INFORMS, v. 26, n. 1, p. 53–67, 1978.

LEMKE, Christiane; BUDKA, Marcin; GABRYS, Bogdan. Metalearning: A survey of trends and technologies. **Artificial intelligence review**, Springer, v. 44, n. 1, p. 117–130, 2015.

LINN, Richard; ZHANG, Wei. Hybrid flow shop scheduling: A survey. **Computers & industrial engineering**, Elsevier, v. 37, n. 1, p. 57–61, 1999.

LIU, Hongcheng; GAO, Liang; PAN, Quanke. A Hybrid Particle Swarm Optimization with Estimation of Distribution Algorithm for Solving Permutation Flowshop Scheduling Problem. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 4, p. 4348–4360, Apr. 2011. ISSN 0957-4174.

López-Ibáñez, Manuel; Dubois-Lacoste, Jérémie; CÁCERES, Leslie Pérez; STÜTZLE, Thomas; BIRATTARI, Mauro. The irace Package: Iterated Racing for Automatic Algorithm Configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

LÜ, Zhipeng; GLOVER, Fred; HAO, Jin-Kao. A hybrid metaheuristic approach to solving the UBQP problem. **European Journal of Operational Research**, v. 207, n. 3, p. 1254–1262, 2010. ISSN 0377-2217.

MADJAROV, Gjorgji; KOCEV, Dragi; GJORGJEVIKJ, Dejan; DŽEROSKI, Sašo. An extensive experimental comparison of methods for multi-label learning. **Pattern Recognition**, v. 45, n. 9, p. 3084–3104, 2012. ISSN 0031-3203.

MALECK, Christian; NIEKE, Gottfried; BOCK, Karlheinz; PABST, Detlef; STEHLI, Marcel. A comparison of an CP and MIP approach for scheduling jobs in production areas with time constraints and uncertainties. *In*: **2018 Winter Simulation Conference (WSC)**. *[S.l.: s.n.]*, 2018. p. 3526–3537.

MANNE, Alan S. On the Job-Shop Scheduling Problem. **Operations Research**, v. 8, n. 2, p. 219–223, 1960.

MARIUSZ, Makuchowski. Permutation, no-wait, no-idle flow shop problems. **Archives of Control Sciences**, v. 25, n. 2, p. 189, 2015.

MARMION, Marie-Eléonore; DHAENENS, Clarisse; JOURDAN, Laetitia; LIEFOOGHE, Arnaud; VEREL, Sébastien. NILS: A Neutrality-Based Iterated Local Search and Its Application to Flowshop Scheduling. *In*: **Evolutionary Computation in Combinatorial Optimization: 11th European Conference, EvoCOP 2011, Torino, Italy, April 27-29, 2011. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 191–202. ISBN 978-3-642-20364-0.

MARMION, Marie-Eléonore; DHAENENS, Clarisse; JOURDAN, Laetitia; LIEFOOGHE, Arnaud; VEREL, Sébastien. On the Neutrality of Flowshop Scheduling Fitness Landscapes. *In*: COELLO, Carlos A. Coello (Ed.). **Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 238–252. ISBN 978-3-642-25566-3.

MARMION, Marie-Eléonore; MASCIA, Franco; López-Ibáñez, Manuel; STÜTZLE, Thomas. Automatic Design of Hybrid Stochastic Local Search Algorithms. *In*: BLESA, María J.; BLUM, Christian; FESTA, Paola; ROLI, Andrea; SAMPELS, Michael (Ed.). **Hybrid Metaheuristics**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 144–158. ISBN 978-3-642-38516-2.

MARMION, Marie-Eléonore; Regnier-Coudert, Olivier. Fitness Landscape of the Factoradic Representation on the Permutation Flowshop Scheduling Problem. *In*: DHAENENS, Clarisse; JOURDAN, Laetitia; MARMION, Marie-Eléonore (Ed.). **Learning and Intelligent Optimization: 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers**. Cham: Springer International Publishing, 2015. p. 151–164. ISBN 978-3-319-19084-6.

MASCIA, Franco; López-Ibáñez, Manuel; Dubois-Lacoste, Jérémie; MARMION, Marie-Éléonore; STÜTZLE, Thomas. Algorithm Comparison by Automatically Configurable Stochastic Local Search Frameworks: A Case Study Using Flow-Shop Scheduling Problems. *In*: BLESA, Maria J.; BLUM, Christian; VOSS, Stefan (Ed.). **Hybrid Metaheuristics**. Cham: Springer International Publishing, 2014. p. 30–44. ISBN 978-3-319-07644-7.

MASCIA, Franco; López-Ibáñez, Manuel; Dubois-Lacoste, Jérémie; STÜTZLE, Thomas. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. **Computers & Operations Research**, v. 51, p. 190–199, 2014. ISSN 0305-0548.

MASTROLILLI, Monaldo; SVENSSON, Ola. Hardness of approximating flow and job shop scheduling problems. **Journal of the ACM (JACM)**, ACM, v. 58, n. 5, p. 20, 2011.

MELAB, N.; CHAKROUN, I.; MEZMAZ, M.; TUYTTENS, D. A GPU-accelerated Branch-and-Bound Algorithm for the Flow-Shop Scheduling Problem. *In*: **Proceedings of the 2012 IEEE International Conference on Cluster Computing**. Washington, DC, USA: IEEE Computer Society, 2012. (CLUSTER '12), p. 10–17. ISBN 978-0-7695-4807-4.

MITCHELL, T.M. **Machine Learning**. *[S.l.]*: McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 978-0-07-115467-3.

MOCCELLIN, J.; SANTOS, M. An adaptive hybrid metaheuristic for permutation flowshop scheduling. **Control and Cybernetics**, Vol. 29, no 3, p. 761–771, 2000.

MONTAÑES, Elena; SENGE, Robin; BARRANQUERO, Jose; QUEVEDO, José Ramón; COZ, Juan José del; HÜLLERMEIER, Eyke. Dependent binary relevance models for multi-label classification. **Pattern Recognition**, v. 47, n. 3, p. 1494–1508, 2014. ISSN 0031-3203.

MONTERO, Elizabeth; RIFF, María-Cristina; NEVEU, Bertrand. A beginner's guide to tuning methods. **Applied Soft Computing**, v. 17, p. 39–51, Apr. 2014. ISSN 15684946.

NAGANO, M. S.; MOCCELLIN, J. V. A High Quality Solution Constructive Heuristic for Flow Shop Sequencing. **The Journal of the Operational Research Society**, Palgrave Macmillan Journals, v. 53, n. 12, p. 1374–1379, 2002. ISSN 01605682, 14769360.

NAWAZ, Muhammad; ENSCORE, E. Emory; HAM, Inyong. A Heuristic Algorithm for the m-Machine, n-Job Flow-Shop Sequencing Problem. **Omega**, v. 11, n. 1, p. 91–95, 1983. ISSN 0305-0483.

NOCEDAL, J.; WRIGHT, S. **Numerical Optimization**. *[S.l.]*: Springer New York, 2006. (Springer Series in Operations Research and Financial Engineering). ISBN 978-0-387-40065-5.

NOWICKI, Eugeniusz; SMUTNICKI, Czeslaw. A Fast Taboo Search Algorithm for the Job Shop Problem. **Management Science**, v. 42, n. 6, p. 797–813, 1996.

OCHOA, Gabriela; MALAN, Katherine. Recent advances in fitness landscape analysis. *In*: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. Prague Czech Republic: ACM, 2019. p. 1077–1094. ISBN 978-1-4503-6748-6.

OCHOA, Gabriela; TOMASSINI, Marco; VÉREL, Sebástien; DARABOS, Christian. A study of NK landscapes' basins and local optima networks. *In*: **Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation - GECCO '08**. Atlanta, GA, USA: ACM Press, 2008. p. 555. ISBN 978-1-60558-130-9.

OSMAN, Ibrahim H; POTTS, CN. Simulated annealing for permutation flow-shop scheduling. **Omega**, Elsevier, v. 17, n. 6, p. 551–557, 1989.

ÖZGÜVEN, Cemal; ÖZBAKIR, Lale; YAVUZ, Yasemin. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. **Applied Mathematical Modelling**, Elsevier, v. 34, n. 6, p. 1539–1548, 2010.

PAGNOZZI, Federico; STÜTZLE, Thomas. **Automatic Design of Hybrid Stochastic Local Search Algorithms for Permutation Flowshop Problems**. *[S.l.]*, 2017.

PALMER, D. S. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time—A Quick Method of Obtaining a Near Optimum. **Journal of the Operational Research Society**, v. 16, n. 1, p. 101–107, 1965. ISSN 1476-9360.

PAN, S.J.; ZHENG, V.W.; YANG, Q.; HU, D.H. Transfer learning for WiFi-based indoor localization. *In*: **AAAI Workshop - Technical Report**. *[S.l.: s.n.]*, 2008. WS-08-13, p. 43–48.

PAPPA, Gisele L.; OCHOA, Gabriela; HYDE, Matthew R.; FREITAS, Alex A.; WOODWARD, John; SWAN, Jerry. Contrasting Meta-learning and Hyper-heuristic Research: The Role of Evolutionary Algorithms. **Genetic Programming and Evolvable Machines**, Kluwer Academic Publishers, Hingham, MA, USA, v. 15, n. 1, p. 3–35, 2014. ISSN 1389-2576.

PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore. Meta-Learning for Optimization: A Case Study on the Flowshop Problem Using Decision Trees. *In*: **2018 IEEE Congress on Evolutionary Computation (CEC)**. *[S.l.: s.n.]*, 2018. p. 1–8.

PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore. Meta-learning on Flowshop Using Fitness Landscape Analysis. *In*: **Proceedings of the Genetic and Evolutionary Computation Conference**. New York, NY, USA: ACM, 2019. (GECCO '19), p. 925–933. ISBN 978-1-4503-6111-8.

PAVELSKI, Lucas Marcondes; DELGADO, Myriam; KESSACI, Marie-Éléonore; FREITAS, Alex A. Stochastic local search and parameters recommendation: A case study on flowshop problems. **International Transactions in Operational Research**, p. itor.12922, Dec. 2020. ISSN 0969-6016, 1475-3995.

PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Recommending Meta-Heuristics and Configurations for the Flowshop Problem via Meta-Learning: Analysis and Design. *In*: **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**. *[S.l.: s.n.]*, 2018. p. 163–168.

PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Dynamic Learning in Hyper-Heuristics to Solve Flowshop Problems (to appear). *In*: **2021 7th Brazilian Conference on Intelligent Systems (BRACIS)**. *[S.l.: s.n.]*, 2021. p. 1–15.

PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Flowshop NEH-Based Heuristic Recommendation. *In*: ZARGES, Christine; VEREL, Sébastien (Ed.). **Evolutionary Computation in Combinatorial Optimization**. Cham: Springer International Publishing, 2021. v. 12692, p. 136–151. ISBN 978-3-030-72903-5 978-3-030-72904-2.

PAVELSKI, Lucas Marcondes; KESSACI, Marie-Éléonore; DELGADO, Myriam. Local Optima Network Sampling for Permutation Flowshop. *In*: **2021 IEEE Congress on Evolutionary Computation (CEC)**. Kraków, Poland: IEEE, 2021. p. 1131–1138. ISBN 978-1-72818-393-0.

PINEDO, Michael; SCHRAGE, Linus. Stochastic shop scheduling: A survey. *In*: **Deterministic and Stochastic Scheduling**. Netherlands: Springer, 1982. p. 181–196.

PINEDO, Michael L. **Scheduling: Theory, Algorithms, and Systems**. Fifth. New York, NY, USA: Springer, 2016.

PITZER, Erik; AFFENZELLER, Michael. Recent Advances in Intelligent Engineering Systems. *In*: . Berlin, Heidelberg: Springer, 2012, (Studies in Computational Intelligence, v. 378). chap. A Comprehensive Survey on Fitness Landscape Analysis, p. 161–186.

POTTS, Chris N; STRUSEVICH, Vitaly A. Fifty years of scheduling: A survey of milestones. **Journal of the Operational Research Society**, Springer, v. 60, n. 1, p. S41–S68, 2009.

PRABHAHARAN, G.; KHAN, B. Shahul Hamid; RAKESH, L. Implementation of GRASP in flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 30, n. 11, p. 1126–1131, 2006. ISSN 1433-3015.

RAJENDRAN, Chandrasekharan; ZIEGLER, Hans. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. **European Journal of Operational Research**, v. 155, n. 2, p. 426–438, 2004. ISSN 0377-2217.

REEVES, C.R. Landscapes, operators and heuristic search. **Annals of Operations Research**, v. 86, n. 0, p. 473–490, Jan. 1999. ISSN 1572-9338.

REEVES, Colin R. Improving the Efficiency of Tabu Search for Machine Sequencing Problems. **The Journal of the Operational Research Society**, Palgrave Macmillan Journals, v. 44, n. 4, p. 375–382, 1993. ISSN 01605682, 14769360.

REEVES, Colin R. A genetic algorithm for flowshop sequencing. **Computers & Operations Research**, v. 22, n. 1, p. 5–13, 1995. ISSN 0305-0548.

REEVES, Colin R.; YAMADA, Takeshi. Genetic Algorithms, Path Relinking, and the Flowshop Sequencing Problem. **Evolutionary Computation**, v. 6, n. 1, p. 45–60, 1998.

RIBAS, Imma; COMPANYS, Ramon; Tort-Martorell, Xavier. Comparing three-step heuristics for the permutation flow shop problem. **Computers & Operations Research**, v. 37, n. 12, p. 2062–2070, 2010. ISSN 0305-0548.

RICE, John R. The Algorithm Selection Problem. *In*: RUBINOFF, Morris; YOVITS, Marshall C. (Ed.). **Advances in Computers**. Washington, DC, USA: Elsevier, 1976, (Advances in Computers, v. 15). p. 65–118.

RICHTER, Hendrik; ENGELBRECHT, Andries P. (Ed.). **Recent Advances in the Theory and Application of Fitness Landscapes**. Berlin Heidelberg: Springer, 2014. (Emergence, Complexity and Computation, Vol. 6). ISBN 978-3-642-41887-7.

RIVOLLI, Adriano; CARVALHO, Andre C. P. L. F. de. The utiml Package: Multi-label Classification in R. **The R Journal**, v. 10, n. 2, p. 24–37, 2018.

RUIZ, Rubén; MAROTO, Concepción. A comprehensive review and evaluation of permutation flowshop heuristics. **European Journal of Operational Research**, v. 165, n. 2, p. 479–494, 2005. ISSN 0377-2217.

RUIZ, Rubén; MAROTO, Concepción. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. **European Journal of Operational Research**, v. 169, n. 3, p. 781–800, 2006. ISSN 0377-2217.

RUIZ, Rubén; STÜTZLE, Thomas. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 177, n. 3, p. 2033–2049, 2007.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. Second. New Jersey, USA: Pearson Education, 2003. ISBN 0-13-790395-2.

SMITH-MILES, Kate; BAATAR, Davaatseren; WREFORD, Brendan; LEWIS, Rhyd. Towards objective measures of algorithm performance across instance space. **Computers & Operations Research**, v. 45, p. 12–24, May 2014. ISSN 03050548.

SMITH-MILES, Kate; LOPES, Leo. Measuring instance difficulty for combinatorial optimization problems. **Computers & Operations Research**, v. 39, n. 5, p. 875–889, 2012. ISSN 0305-0548.

SMITH-MILES, Kate; WREFORD, Brendan; LOPES, Leo; INSANI, Nur. Predicting metaheuristic performance on graph coloring problems using data mining. *In*: **Hybrid Metaheuristics**. Berlin, Heidelberg: Springer, 2013. p. 417–432.

SMITH-MILES, K. A. Towards insightful algorithm selection for optimisation using meta-learning concepts. *In*: **2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)**. Hong-Kong: IEEE, 2008. p. 4118–4124. ISSN 2161-4393.

SMITH-MILES, Kate A. Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 41, n. 1, p. 6:1–6:25, 2009. ISSN 0360-0300.

SMITH-MILES, Kate A.; JAMES, Ross J. W.; GIFFIN, John W.; TU, Yiqing. A Knowledge Discovery Approach to Understanding Relationships between Scheduling Problem Structure and Heuristic Performance. *In*: **Learning and Intelligent Optimization: Third International Conference, LION 3, Trento, Italy, January 14-18, 2009. Selected Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 89–103. ISBN 978-3-642-11169-3.

SÖRENSEN, Kenneth. Metaheuristics-the metaphor exposed. **International Transactions in Operational Research**, v. 22, n. 1, p. 3–18, Jan. 2015. ISSN 09696016.

STADLER, Peter F. Landscapes and their correlation functions. **Journal of Mathematical Chemistry**, v. 20, n. 1, p. 1–45, Mar. 1996. ISSN 1572-8897.

STADLER, Peter F. Fitness landscapes. *In*: LÄSSIG, Michael; VALLERIANI, Angelo (Ed.). **Biological Evolution and Statistical Physics**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. v. 585, p. 183–204. ISBN 978-3-540-43188-6.

STÜTZLE, Thomas. An Ant Approach to the Flow Shop Problem. *In*: **In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)**. Aachen, Germany: Verlag, 1998. p. 1560–1564.

STÜTZLE, Thomas. **Applying Iterated Local Search to the Permutation Flow Shop Problem**. *[S.l.]*, 1998.

SWAN, Jerry; ADRIAENSEN, Steven; BROWNLEE, Alexander E.I.; HAMMOND, Kevin; JOHNSON, Colin G.; KHEIRI, Ahmed; KRAWIEC, Faustyna; MERELO, J.J.; MINKU, Leandro L.; ÖZCAN, Ender; PAPPA, Gisele L.; García-Sánchez, Pablo; SÖRENSEN, Kenneth; VOSS, Stefan; WAGNER, Markus; WHITE, David R. Metaheuristics "In the Large". **European Journal of Operational Research**, p. S0377221721004707, Jun. 2021. ISSN 03772217.

TAILLARD, Éric. Some efficient heuristic methods for the flow shop sequencing problem. **European Journal of Operational Research**, v. 47, n. 1, p. 65–74, 1990. ISSN 0377-2217.

TSENG, Fan T.; JR, Edward F. Stafford; GUPTA, Jatinder N. D. An empirical analysis of integer programming formulations for the permutation flowshop. **Omega**, v. 32, n. 4, p. 285–293, 2004. ISSN 0305-0483.

TSOUMAKAS, Grigorios; KATAKIS, Ioannis; VLAHAVAS, Ioannis. Mining multi-label data. *In*: **Data Mining and Knowledge Discovery Handbook**. *[S.l.]*: Springer, 2009. p. 667–685.

TZENG, Yeu-Ruey; CHEN, Chun-Lung; CHEN, Chuen-Lung. A hybrid EDA with ACS for solving permutation flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 60, n. 9, p. 1139–1147, 2012. ISSN 1433-3015.

VALLADA, Eva; RUIZ, Rubén; FRAMINAN, Jose M. New hard benchmark for flowshop scheduling problems minimising makespan. **European Journal of Operational Research**, v. 240, n. 3, p. 666–677, Feb. 2015. ISSN 0377-2217.

VASSILEV, Vesselin K; FOGARTY, Terence C; MILLER, Julian F. Information characteristics and the structure of landscapes. **Evolutionary computation**, MIT Press, v. 8, n. 1, p. 31–60, 2000.

VEN, Gido M Van de; TOLIAS, Andreas S. Three scenarios for continual learning. **arXiv preprint arXiv:1904.07734**, 2019.

VEREL, S.; COLLARD, P.; TOMASSINI, M.; VANNESCHI, L. Neutral Fitness Landscape in the Cellular Automata Majority Problem. *In*: YACOUBI, Samira El; CHOPARD, Bastien; BANDINI, Stefania (Ed.). **Cellular Automata**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 258–267. ISBN 978-3-540-40932-8.

VEREL, Sébastien; DAOLIO, Fabio; OCHOA, Gabriela; TOMASSINI, Marco. Sampling Local Optima Networks of Large Combinatorial Search Spaces: The QAP Case. *In*: **Parallel Problem Solving from Nature (PPSN XV)**. Coimbra, Portugal: Springer, 2018. p. 257–268.

WAGNER, Harvey M. An integer linear-programming model for machine scheduling. **Naval Research Logistics Quarterly**, Wiley Subscription Services, Inc., A Wiley Company, v. 6, n. 2, p. 131–140, 1959. ISSN 1931-9193.

WAGNER, Markus; LINDAUER, Marius; MISIR, Mustafa; NALLAPERUMA, Samadhi; HUTTER, Frank. A case study of algorithm selection for the traveling thief problem. **Journal of Heuristics**, v. 24, n. 3, p. 295–320, Jun. 2018. ISSN 1572-9397.

WANG, Ling; ZHENG, Da-Zhong. A modified evolutionary programming for flow shop scheduling. **The International Journal of Advanced Manufacturing Technology**, v. 22, n. 7, p. 522–527, 2003. ISSN 1433-3015.

WANG, Yaqing; YAO, Quanming; KWOK, James T; NI, Lionel M. Generalizing from a few examples: A survey on few-shot learning. **ACM computing surveys (csur)**, ACM New York, NY, USA, v. 53, n. 3, p. 1–34, 2020.

WATSON, Jean-Paul. An Introduction to Fitness Landscape Analysis and Cost Models for Local Search. *In*: GENDREAU, Michel; POTVIN, Jean-Yves (Ed.). **Handbook of Metaheuristics**. Boston, MA: Springer US, 2010. p. 599–623. ISBN 978-1-4419-1665-5.

WATSON, Jean-Paul; BARBULESCU, Laura; HOWE, Adele E; WHITLEY, L Darrell. Algorithm Performance and Problem Structure for Flow-shop Scheduling. *In*: **AAAI/IAAI**. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999. p. 688–695.

WATSON, Jean-Paul; WHITLEY, Darrell; HOWE, Adele. Linking Search Space Structure, Run-Time Dynamics, and Problem Difficulty: A Step Toward Demystifying Tabu Search. **J. Artif. Intell. Res. (JAIR)**, v. 24, p. 221–261, Jul. 2005.

WEINBERGER, Edward D. Correlated and uncorrelated fitness landscapes and how to tell the difference. **Biological Cybernetics**, v. 63, n. 5, p. 325–336, Sep. 1990. ISSN 1432-0770.

WIDMER, Marino; HERTZ, Alain. A new heuristic method for the flow shop sequencing problem. **European Journal of Operational Research**, v. 41, n. 2, p. 186–193, 1989. ISSN 0377-2217.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, Apr. 1997. ISSN 1089-778X.

WU, Xi-Zhu; ZHOU, Zhi-Hua. A Unified View of Multi-label Performance Measures. *In*: **Proceedings of the 34th International Conference on Machine Learning - Volume 70**. *[S.l.]*: JMLR.org, 2017. (ICML'17), p. 3780–3788.

XU, Lin; HUTTER, Frank; HOOS, Holger H; Leyton-Brown, Kevin. SATzilla: Portfolio-based algorithm selection for SAT. **Journal of artificial intelligence research**, v. 32, p. 565–606, 2008.

YENISEY, Mehmet Mutlu; YAGMAHAN, Betul. Multi-objective permutation flow shop scheduling problem Literature review, classification and current trends. **Omega**, v. 45, p. 119–135, 2014. ISSN 0305-0483.

ZEGORDI, Seyed Hessameddin; ITOH, Kenji; ENKAWA, Takao. Minimizing makespan for flow shop scheduling by combining simulated annealing with sequencing knowledge. **European Journal of Operational Research**, v. 85, n. 3, p. 515–531, 1995. ISSN 0377-2217.

ZHANG, Qingfu; LI, Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. **IEEE Transactions on Evolutionary Computation**, v. 11, n. 6, p. 712–731, 2007.

ZHANG, Tiantian; GEORGIOPOULOS, Michael; ANAGNOSTOPOULOS, Georgios C. S-Race: A multi-objective racing algorithm. *In*: **Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference - GECCO '13**. Amsterdam, The Netherlands: ACM Press, 2013. p. 1565. ISBN 978-1-4503-1963-8.

ZHANG, Tiantian; GEORGIOPOULOS, Michael; ANAGNOSTOPOULOS, Georgios C. SPRINT Multi-Objective Model Racing. *In*: **Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation**. Madrid Spain: ACM, 2015. p. 1383–1390. ISBN 978-1-4503-3472-3.

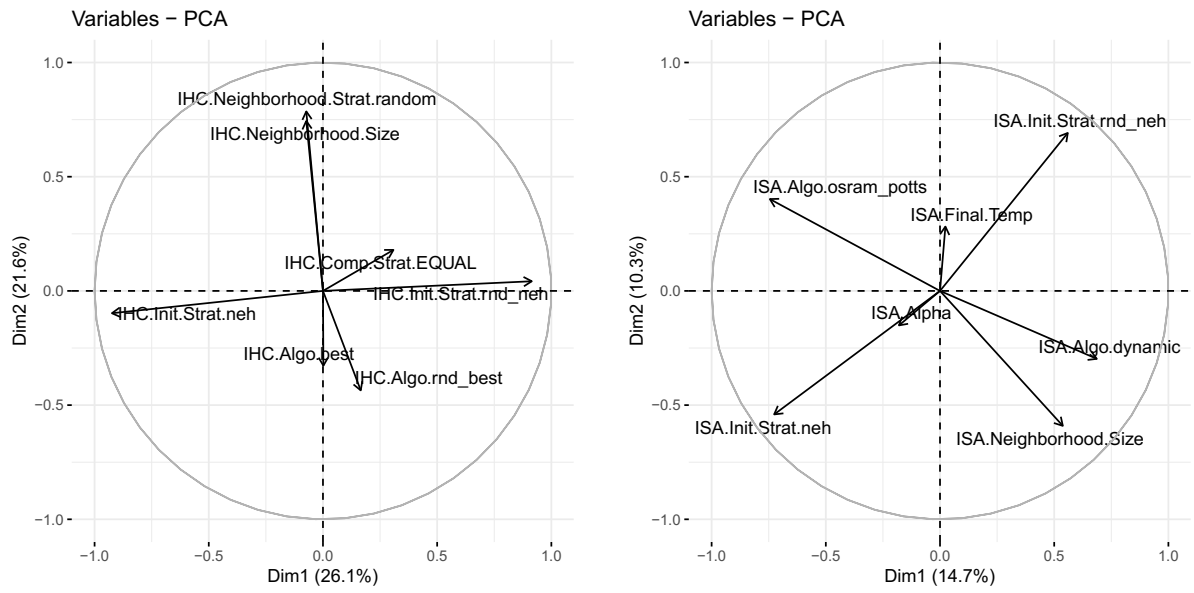
APPENDIX

APPENDIX A – METAHEURISTIC AAC ANALYSIS

In this appendix, we investigate the instance-based algorithm configuration data, using irace for 1296 groups of instances: $J \in \{10,20,30,50\}$, $M \in \{10,20,30\}$, 3 processing time distributions (uniform, exponential and binomial), 3 processing time correlation (job-, machine- and non-correlated), 2 objectives (makespan and flowtime), 2 stopping criteria (number of evaluations and time) and 3 budgets (low, medium and high). For each feature combination, irace uses as the training set 5 instances with different processing times samples.

Figures 27 to 29 show PCA variable plots for each MH with the best projected ($cos^2 > 0.05$) parameters. For that, all categorical parameters are transformed into k dummy indicator parameters, where k is the number of parameter values.

Figure 27 – PCA for HC and SA parameters.



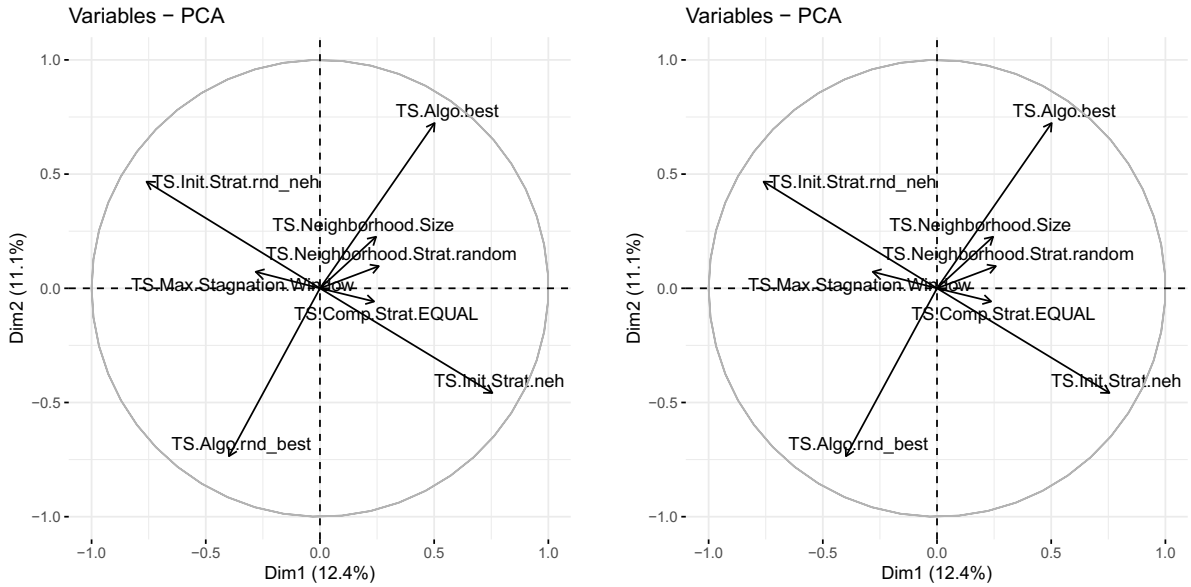
(a) HC configurations PCA. **(b) SA configurations PCA.**

Source: The author.

From HC parameters (Figure 27(a)), we notice that random neighborhood are larger and local searches like best and random best are used with smaller neighborhoods. Figure 27(b) shows that Osman and Potts (1989) cooling schedule uses smaller neighborhoods, while dynamic cooling schedule uses large neighborhoods. SA initial temperatures and span sizes are not well projected.

Regarding the TS and ACO parameters from Figure 28, we notice that most TS parameters are not well projected. For ACO on the other hand, we notice that random best local search is usually accompanied with larger neighborhood sizes and single-step local searches while

Figure 28 – PCA for TS and ACO parameters.

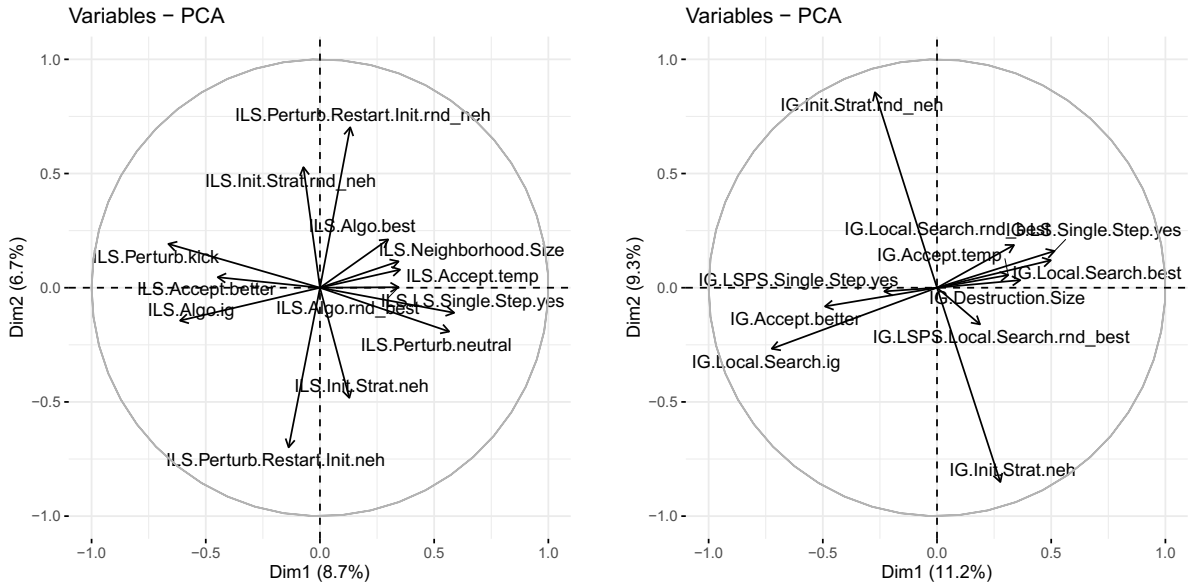


(a) TS configurations PCA. **(b) ACO configurations PCA.**

Source: The author.

the greedy insert local search of IG uses small neighborhoods and better or equal comparison operator.

Figure 29 – PCA for ILS and IG parameters.



(a) ILS configurations PCA. **(b) IG configurations PCA.**

Source: The author.

Figure 29 shows the PCA for ILS and IG configuration spaces. For ILS, we notice that metropolis hastings acceptance criterion is mostly used with larger neighborhoods and random-best local searches, while greedy insertion local searches uses kick perturbation accepting only

improving solutions (similar to the ILS configuration proposed by Stützle (1998b)). For IG, the local search based on best solutions on the neighborhood is correlated with single-step local searches and, like ILS, configurations with greedy insertion local search mostly uses the acceptance criterion on only improving solutions.

Overall, we notice that parameters like local search, acceptance criterion are better projected. Some categorical parameters indicators like random initialization strategy are never shown, mostly because the best configurations almost never use random initialization. Comparison operators and most numerical parameters are not well projected, which implies that either their values setting do not depend on other parameters, or they assume the same value for most problems.

APPENDIX B – METAHEURISTICS PARAMETERS

This appendix describes the parameter for each MH implemented in this work for solving flowshop problems. There are two basic parameter types or domains: categorical (like initialization and local search strategies) and numerical (like neighborhood size and temperature decay factor). Some parameters are also conditional to each other; for example, the ILS accept temperature factor is only used when the metropolis hastings acceptance criterion is used. All MHs and operators are implemented using the ParadisEO framework (HUMEAU *et al.*, 2013).

The parameters, domain and conditionals are described in Tables 14 to 19. HC is a simple MH using a local search strategy and restarting until the evaluations budget is exhausted. SA has a simple, dynamic and a cooling schedule based on Osman and Potts (1989). TS can use different local searches and tabu list types. ACO is based on the proposal presented by Stützle (1998a). ILS is mainly based on the approach proposed by Stützle (1998b), including new perturbation operators like neutral walk perturbation (MARMION *et al.*, 2011b). Finally, IG is implemented providing two variants, the classic IG from Ruiz and Stützle (2007) and the state-of-the-art IG based on partial solution optimization from (Dubois-Lacoste *et al.*, 2017).

Table 14 – HC parameters.

Parameters	Descriptions (possible values) and dependencies
Initialization	(random, NEH or randomized NEH)
Neighborhood type	How neighborhood is explored (ordered or random)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
HC variant	First improvement (FI), local search the best (BEST), random local search for the best (RANDOM_BEST)

Source: The author.

Table 15 – SA parameters

Parameters	Descriptions (possible values) and dependencies
Initialization	(random, NEH or randomized NEH)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
SA variant	SA variant Classic SA (SIMPLE) or with dynamic cooling schedule (DYNAMIC)
Initial temperature	(Real between 5.0 and 10.0)
Final temperature	(Real between 0.1 and 1.0) used if SA variant is SIMPLE
Alpha	Temperature decay rate (Real between 0.1 e 1.0)
Span	Temperature update span (Integer between 5 e 100) used if SA variant is SIMPLE
Maximum span, maximum moves and maximum updates	Maximum number of tries (integer between 1000 and 7000), moves (integer between 50 and 500) and updates (integer between 2 e 10) with the same temperature, used if SA variant is DYNAMIC
Stagnation window	Maximum number of moves with same fitness until a restart (integer between 1 and budget size)

Source: The author.

Table 16 – TS parameters

Parameter	Description (and possible values)
Initialization	(random, NEH or randomized NEH)
Neighborhood type	How neighborhood is explored (ordered or random)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
TS variant	TS local search strategy of first improvement (FI), best neighbor (BEST) or randomized best neighbor (RANDOM_BEST)
Tabu list type	Simple tabu list (INDEXED), with variable tabu period (RANDOM_INDEXED), neighbors tabu (NEIGHBOR) or solutions tabu (SOL)
Aspiration function	None (NONE) or best improvement (BEST)
Tabu list size	Maximum tabu list size (integer between 2 and 10), used if tabu list type is NEIGHBOR or SOL
Tabu length	Number of iterations a move is considered tabu (integer between 1 and 10)
Tabu length variance	Maximum number of iterations added to the tabu length (integer between 1 and 8), used if tabu list type is RANDOM_INDEXED (LÜ *et al.*, 2010)

Source: The author.

Table 17 – ACO parameters

Parameter	Description (and possible values)
Initialization	(random, NEH or randomized NEH)
Neighborhood type	How neighborhood is explored (ordered or random)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
Local Search	ACO local search strategy of first improvement (FI), best neighbor (BEST), randomized best neighbor (RANDOM_BEST) or greedy insert (IG)
Single-step local search	Indicates (true or false) if the local search executes for a Single-step
Pheromone decreasing factor ρ	Factor to update the current pheromone levels $\tau_{t+1} \leftarrow \rho\tau_t$ (real value between 0 and 1)
Minimum pheromone factor s	Scaling of the maximum pheromone to define the minimum level $\tau_{min} = s\tau_{max}$ (real value between 0 and 1) where $\tau_{max} = ((1-\rho)f_0)^{-1}$ and f_0 is the initial solution fitness.
Random mutation probability	Probability of choosing a random job during the construction phase (real value between 0 and 1)

Source: The author.

Table 18 – ILS parameters

Parameter	Description (and possible values)
Initialization	(random, NEH or randomized NEH)
Neighborhood type	How neighborhood is explored (ordered or random)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
Local search	First improvement (FI), best (BEST), randomized best (RANDOM_BEST), greedy insertion (IG)
Perturbation	Restart (RESTART), solution kick with random swaps (KICK) (STÜTZLE, 1998b) or neutral walk perturbation strategy (NILS) (MARMION *et al.*, 2011b)
Acceptance criterion	Always accept (ALWAYS), accept improvements (BETTER) or metropolis hastings (TEMPERATURE) (STÜTZLE, 1998b)
Accept temperature	Temperature factor (real between 0 and 5.0), used if acceptance criterion is TEMPERATURE
Single-step local search	Indicates (true or false) if the local search executes for a Single-step
Restart initialization	Initialization used on restart perturbation (random, NEH or randomized NEH), used if perturbation is RESTART
Restart threshold	Number of non-improving steps until restart operator is applied (integer between 0 and 10), used if perturbation is RESTART
Number of kicks	Number of random swaps (integer between 1 and 3), used if perturbation is KICK
Kick strength factor ks	Maximum positional distance of a swap, relative to the job size (real between 0 and 1), used if perturbation is KICK
Neutral walk number of steps	Neutral walk's maximum number of moves (integer between 0 and 1000), used if perturb is NILS
Neutral escape perturbation	Perturbation applied in NILS (random, NEH, randomized NEH, IG destruction or solution kick), used if perturb is NILS
Neutral escape destruction size	Number of destructions of IG style perturbation on NILS escape, used if perturb is NILS and NILS perturb is IG destruction
Neutral escape number of kicks	Number of random swaps of kick perturbation on NILS escape (integer between 1 and 3), used if perturb is NILS and NILS perturb is solution kick

Source: The author.

Table 19 – IG parameters

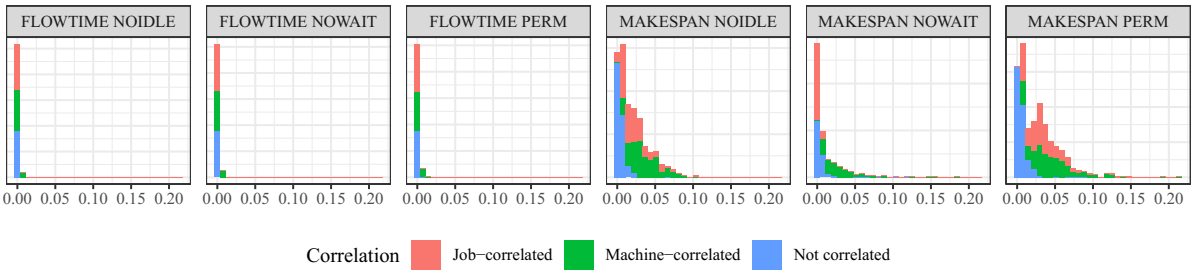
Parameter	Description (and possible values)
Initialization	(random, NEH or randomized NEH)
Neighborhood type	How neighborhood is explored (ordered or random)
Comparison	Used to compare solutions (strict $>$, or equal $\geq$)
Neighborhood size	(Integers between 1 and 2^{N-1} normalized to 0%-100%)
Local search	First improvement (FI), best (BEST), randomized best (RANDOM_BEST), greedy insert (IG)
Single-step local search	Indicates (true or false) if the local search executes for a Single-step
Acceptance criterion	Always accept (ALWAYS), accept improvements (BETTER) or metropolis hastings (TEMPERATURE) (STÜTZLE, 1998b)
Accept temperature	Temperature factor (real between 0 and 5.0), used if acceptance criterion is TEMPERATURE
Destruction size factor s_d	IG destruction size n_d relative to the total number of jobs J, i.e, $n_d = s_d J$
IG variant	Original IG or IG with local search on partial solutions (Dubois-Lacoste *et al.*, 2017) (IG or IG-LSPS)
Single-step partial local search	Indicates (true or false) if the partial local search executes for a Single-step

Source: The author.

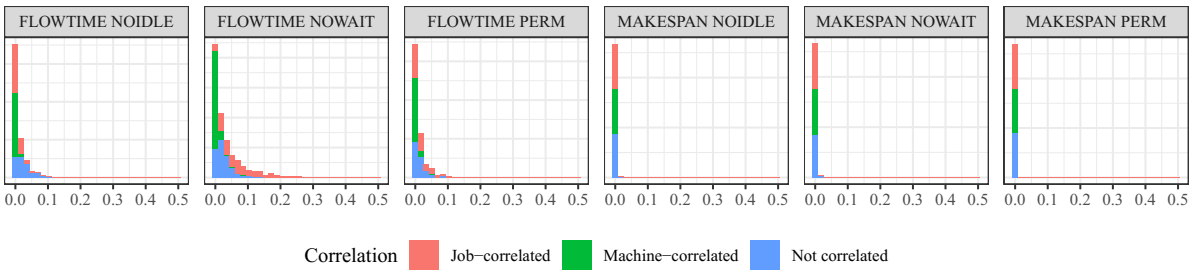
APPENDIX C – FLOWSHOP FITNESS LANDSCAPE ANALYSIS

Aiming to show insights on the addressed FSP instances, in this chapter, we show the values obtained using different FLA metrics on 3240 flowshop instances with: $J \in \{10,20,30,50\}$, $M \in \{10,20,30\}$, 3 processing time distributions (uniform, exponential and binomial), 3 processing time correlation (job-, machine- and non-correlated), 2 objectives (makespan and flowtime), 3 flowshop variants (permutation, no-wait and no-idle) and 5 different random seeds for each feature combination.

Figure 30 – Histograms of edge and solution type FLA metrics per objective and flowshop type.



(a) Proportion of side edge types.



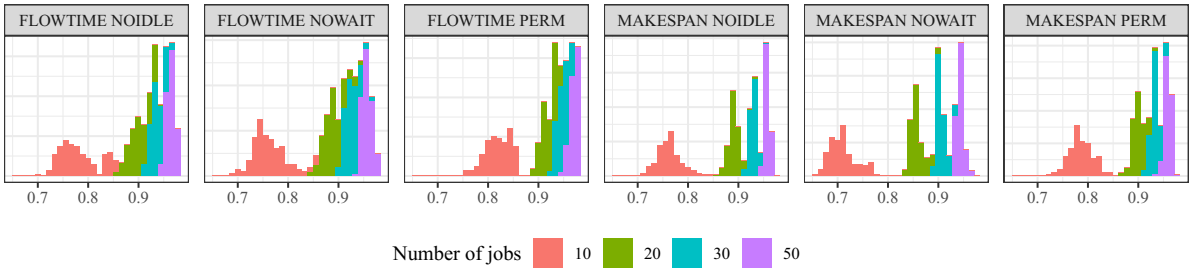
(b) Proportion of SLOPE solution types.

Source: The author.

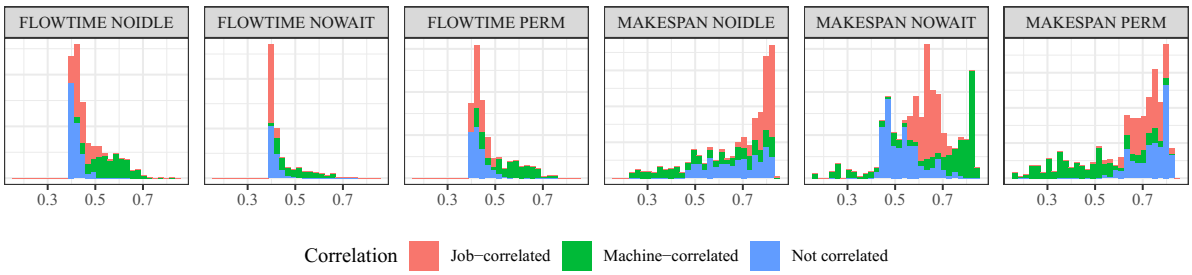
Regarding solution and edge type statistics, we measure the type of 1000 samples of the search space of each instance, flowshop type and objective. The proportion of up and down edges are similar for all problems, normally distributed around 50%. The proportion of side edges is higher on makespan problems and job-correlated processing times, as shown in Figure 30(a). Plateau, local minima or maxima solutions are rarely present. Slope solutions are more common in job-correlated and non-correlated solutions with flowtime objective, as shown in Figure 30(b).

Other FLA metrics are based on a 10,000 step random walks starting from a random solution. From this, we measure the random walk autocorrelation with delays of 1, 2, and 4, as well as the other information theory measures from Section 2.2.2. The autocorrelation with delay 1 mainly depends on the number of jobs, as seen in Figure 31(a), higher delay values show the same

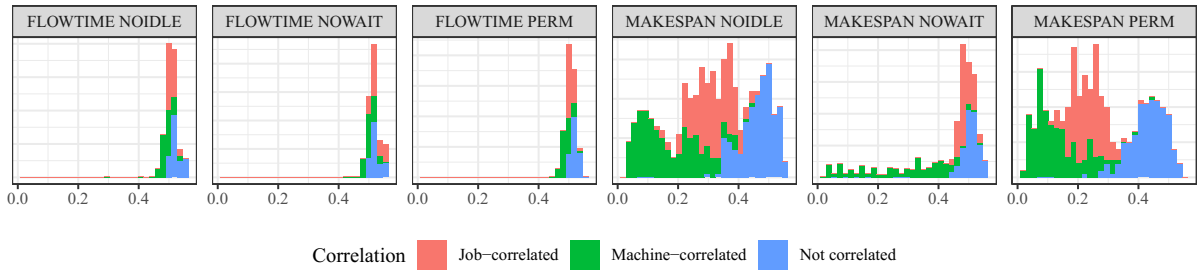
Figure 31 – Histograms of FLA metrics based on random walks per objective and flowshop type.



(a) Random walk autocorrelation with delay of 1.



(b) Entropy of random walk fitness values.



(c) Partial information of random walk fitness values.

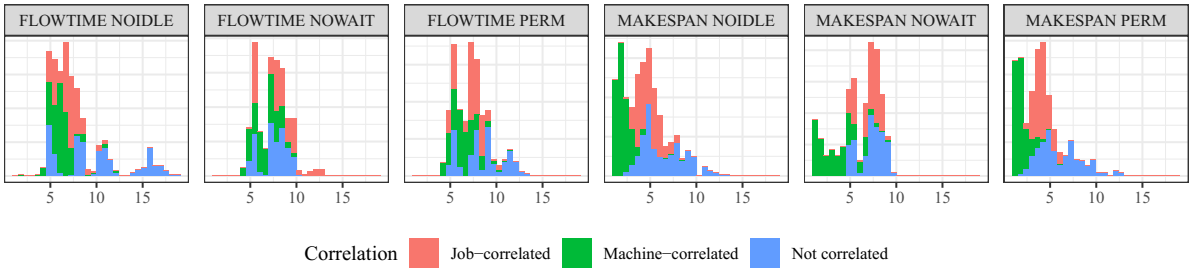
Source: The author.

behavior. As seen in Figure 31(b), entropy is high mostly in job-correlated and non-correlated problems of no-idle and permutation problems. For no-wait flowshop, the machine-correlated instances show high entropy but low partial information, as seen in Figure 31(c). Information stability is dependent on the fitness range and is lower for makespan objective and high for instances with exponentially distributed processing times.

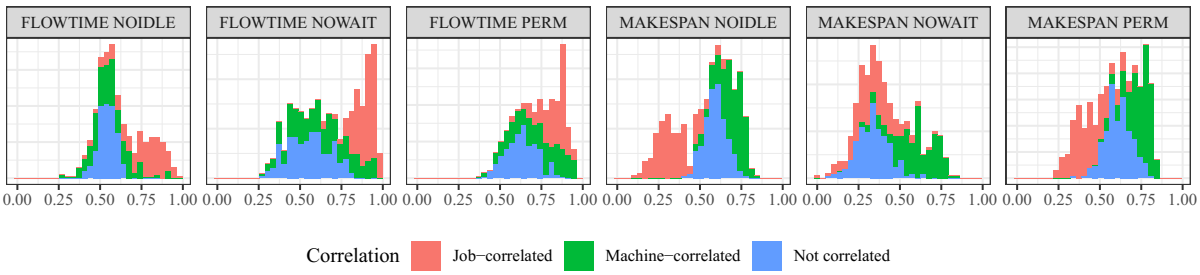
One of the most informative FLA metric types are local optima networks (LONs). We use the snowball procedure with $l = 15$ random walk steps, $m = 10$ local optima neighbors and depth $d = 3$. The local search used is a random best HC, and the perturbation operator performs two random swaps. From the sampling, we also obtain other FLA metrics such as FDCs, adaptive walk lengths, and mean distances between local optima.

From the performed adaptive walks, we observe in Figure 32(a) that the mean length to reach a local optimum is lower on makespan objective and machine-correlated instances.

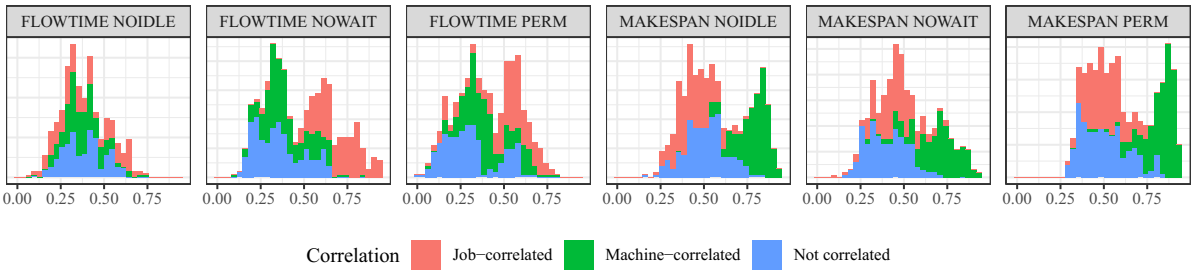
Figure 32 – Histograms of FLA metrics based on adaptive walks per objective and flowshop type.



(a) Mean number of steps to reach a local optima solution.



(b) FDC using job-precedence distance.



(c) FDC using shift distance.

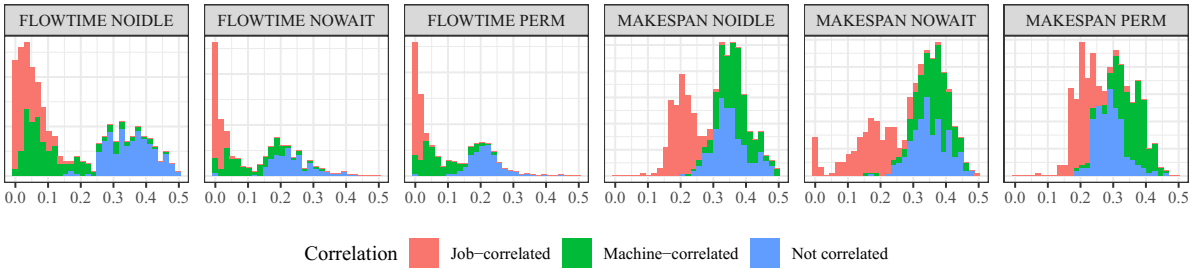
Source: The author.

The FDC values, considering job-precedence distance, is high for job-correlated instances with flowtime objective and machine-correlated instances with makespan objective. Other distances also follow this pattern as seen for shift distance in Figure 32(c). Although, in some cases like makespan no-idle flowshop the different FDCs complement each other on distinguishing job- and machine-correlated instances.

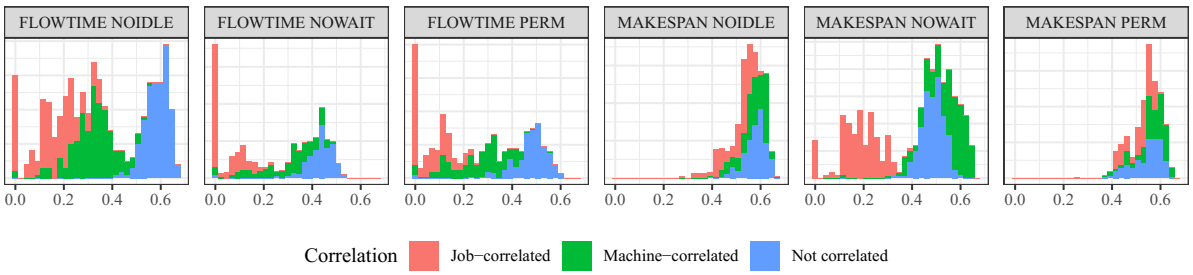
Figures 33(a) and 33(b) show the mean distance between local optima using precedence and shift distance, respectively. From the histograms, we observe that job-correlated instances have closer local optima, except for no-idle and permutation flowshop with makespan objectives, which present local optima with similar distances to each other.

FLA metrics derived from LONs and CLONs are shown in Figures 34 and 35, respectively. The LONs with a greater number of nodes are machine-correlated instances, as well as no-wait and permutation flowshop problems with makespan objective. In most of this LONs

Figure 33 – Histograms of the mean distance between local optima per objective and flowshop type.



(a) Mean job-precedence distance between local optima.



(b) Mean shift distance between local optima.

Source: The author.

sampling always reaches a new local optimum every time it applies the perturbation operator. The output degree is lowest on job-correlated instances, mostly from small LONs with several self-loops. Fitness-fitness correlation mostly does not depend on the instance features. For the other measures, job-correlated instances have high average self-loop weights and high weight disparity for flowtime objective.

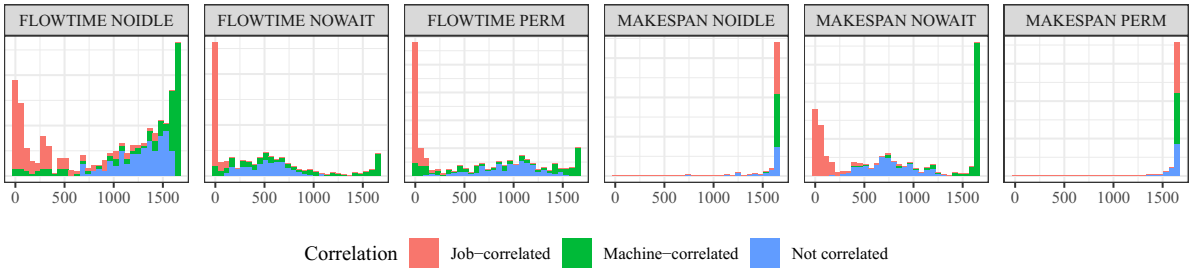
From the derived CLONs, Figure 35(a) shows the distribution of the number of nodes. The correlated instances have their LONs compacted into a few neutral nodes, as also shown by the average node size in Figure 35(b). The average output degree is smaller in non-correlated instances, mainly for flowshop objective (Figure 35(c)).

The sampling process is also tested using more complex initialization procedures and local searches like NEH and IG algorithm. For those, LONs are smaller and have a better distinction of correlation types for the mean average distance between local optima.

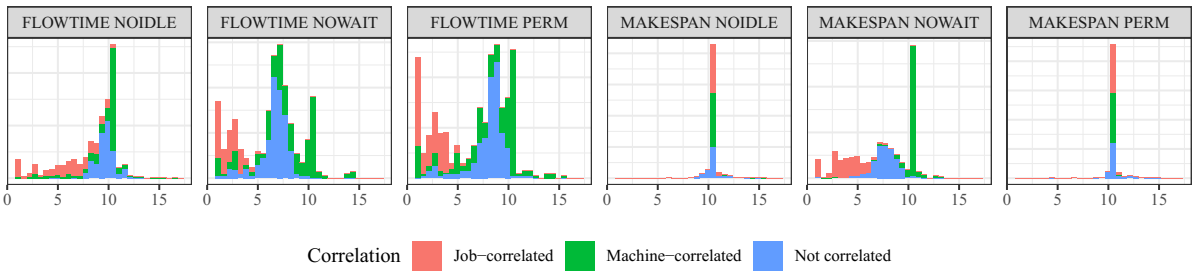
In order to observe the correlation between the different FLA metrics, we use principal component analysis (PCA) to find the best projection of all FLA metrics computed (including others not mentioned, like FDC with different metrics, other graph-based metrics for LONs). Several FLA metrics are correlated with each other, like random-walk autocorrelation with different delays, FDC, and mean distance between local optima with different distances.

Figure 36 shows the PCA graph of variables with some chosen FLA metrics to avoid

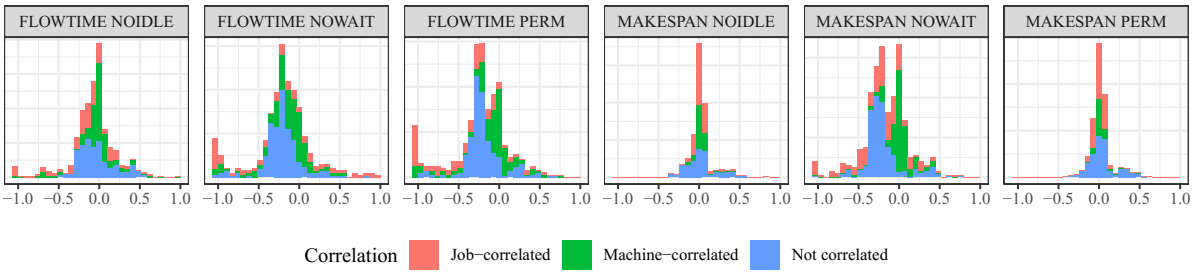
Figure 34 – Histograms of LON based metrics per objective and flowshop type.



(a) LON sizes (excluding instances with 10 jobs).



(b) LON average output degrees.



(c) LON fitness-fitness correlation.

Source: The author.

over-plotting. The number of jobs and machines are shown as supplementary variables. As noted above, the autocorrelation is related to the number of jobs. The number of machines does not seem to influence any FLA metric.

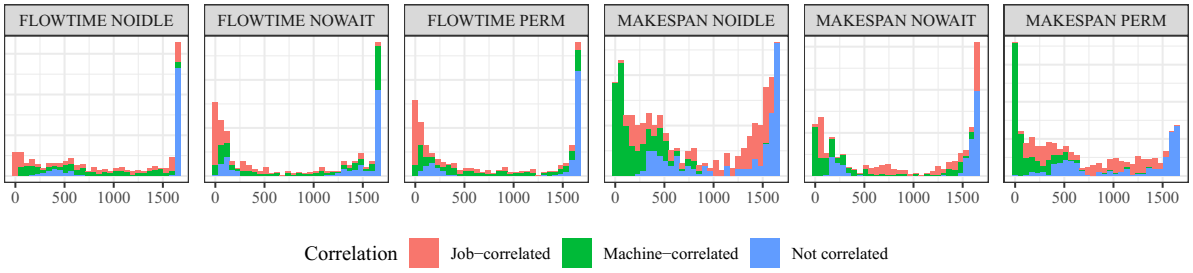
In order to better understand the importance of each FLA metric in predicting the problem hardness, the following ratio is computed for all instances:

$$ig\_perf = \frac{\text{IG performance with 100 * J function evaluations}}{\text{Random local optima fitness}}$$

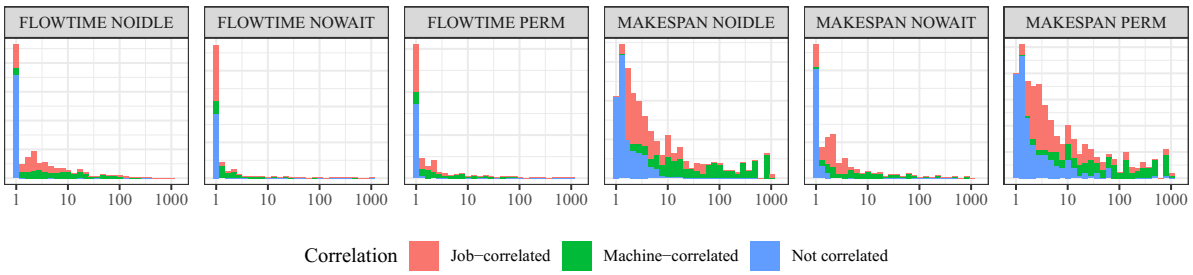
ig_perf is close to 1 for easy instances where a simple HC is comparable to a state-of-the-art IG. Otherwise, the instance is not trivial, and the IG run can significantly improve the final fitness. This metric is also projected on PCA of Figure 36, showing that IG outperforms HC most frequently on instances where:

- the number of jobs is high (subsequently the autocorrelation is high);

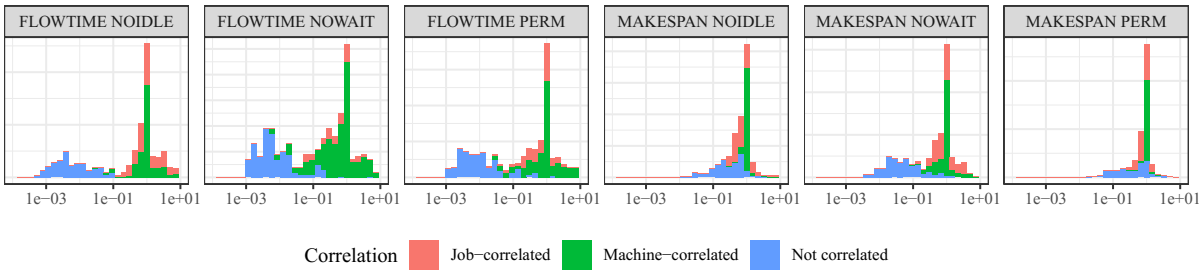
Figure 35 – Histograms of CLON based metrics per objective and flowshop type.



(a) CLON sizes (excluding instances with 10 jobs).



(b) CLON average node size.



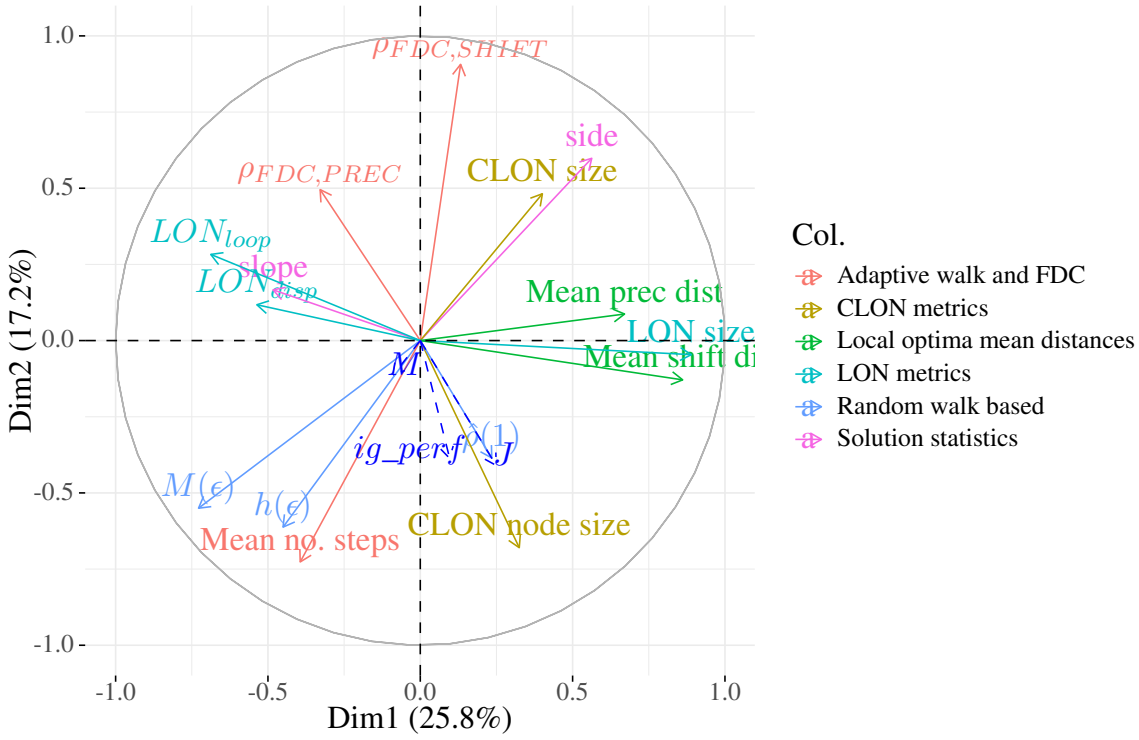
(c) CLON average output degrees.

Source: The author.

- the neutral regions are small and separate (high number of neutral groups or CLON nodes);
- the partial information and density basin is high;
- the average number of steps until finding a local optimum is high;
- The FDC is low.

The main conclusions from analyzing FLA metrics are that most of them can be used to indicate some instance feature. Several metrics are sensitive to the type of processing times correlation, as well as the objective, and type of flowshop problem. Some features can, and possibly some combination of them can also indicate instance hardness, as seen in the case of correlation with ig_perf.

Figure 36 – Principal component analysis for FLA metrics (supplementary variables are shown as dashed arrows).



Source: The author.