

Knowledge-based Design of Stochastic Local Search Algorithms in Combinatorial Optimization

Marie-Eléonore Kessaci

▶ **To cite this version:**

Marie-Eléonore Kessaci. Knowledge-based Design of Stochastic Local Search Algorithms in Combinatorial Optimization. Discrete Mathematics [cs.DM]. Université de Lille, 2019. tel-02430822

HAL Id: tel-02430822
https://hal.science/tel-02430822

Submitted on 11 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université de Lille
École Doctorale Sciences Pour l'Ingénieur Université Lille Nord-de-France
Centre de Recherche en Informatique, Signal et Automatique de Lille (UMR CNRS 9189)

Habilitation à Diriger des Recherches

Discipline : Informatique

Knowledge-based Design of Stochastic Local Search Algorithms in Combinatorial Optimization

Marie-Éléonore Kessaci

Date de soutenance : 14 Novembre 2019

Membres du jury :

- Géry Casiez, Université de Lille, France, *examinateur*
- Clarisse Dhaenens, Université de Lille, France, *invitée*
- Holger Hoos, University of Leiden, Pays-Bas, *invité*
- Lhassane Idoumghar, Université de Haute-Alsace, France, *examinateur*
- Laetitia Jourdan, Université de Lille, France, *garant*
- Gabriela Ochoa, University of Stirling, Royaume-Uni, *rapporteur*
- Frédéric Saubion, Université d'Angers, France, *rapporteur*
- Christine Solnon, INSA de Lyon, France, *rapporteur*
- Thomas Stützle, Université Libre de Bruxelles, Belgique, *examinateur*

Numéro d'ordre : xxx | Année : 2019

Université de Lille

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

cnrs

Acknowledgments

First of all, I would like to thank the members of my habilitation committee for their acceptance of the invitation, their attendance and questions during the defense. Géry Casiez for chairing the committee, Gabriela Ochoa, Frédéric Saubion and Christine Solnon for reviewing my habilitation thesis, Holger Hoos, Lhassane Idoumghar and Thomas Stützle for the attention payed to my work and, Clarisse Dhaenens and Laetitia Jourdan for their support.

Besides, I would like to take this opportunity to thank all the people who have played an important role since the end of my PhD. Thank you Thomas for the postdoc opportunity that you gave me. It made me grow up and gave me the chance to collaborate with great people - Manuel and Franco - that I would like to thank as well. Thank you Holger for the interesting discussions which led to our fruitful collaboration and looking forward for many others. Thank you also Myriam for your good mood, your friendliness and our successful collaboration.

Clarisse et Laetitia, je ne sais comment vous remercier à la hauteur de ce que je ressens. Merci pour tous vos conseils, merci pour votre bienveillance, merci pour votre soutien, merci pour votre confiance, merci pour les moyens donnés et merci pour votre amitié. Je prends toujours autant de plaisir à travailler avec vous depuis 11 ans et j'espère que nous continuerons encore longtemps.

Je remercie également mes collègues de Polytech, en particulier, les membres de l'équipe pédagogique GIS ainsi que notre super secrétaire pédagogique, Corinne. Mes remerciements vont aussi vers mes collègues du laboratoire CRIStAL, les membres actuels et passés de l'équipe ORKAD, en particulier Nada (merci pour les relectures) et mes doctorants actuels et passés : Aymeric, Lucien, Lucas, Weerapan et Laurent.

Pour finir mes remerciements vont à ma famille : mes parents, mes beaux-parents, mes frères et beau-frère. Votre soutien et votre confiance ont été essentiels. Enfin, merci Yacine, pour TOUT : tes conseils, ta patience, ton accompagnement, notre magnifique petite fleur et j'espère les autres bonheurs à venir.

Contents

Introduction 1

I Automatic Algorithm Configuration 11

1 AAC of Hybrid SLS 13
1.1 Generalized Local Search Structure . . . 14
1.2 Implementation . . . 15
1.2.1 A practical implementation of the GLS structure . . . 15
1.2.2 A grammar description of the GLS structure . . . 16
1.2.3 Automatic generation of hybrid LS metaheuristics . . . 17
1.3 Experimental Environment . . . 18
1.3.1 The PFSP-WT . . . 18
1.3.2 Local search components for the PFSP-WT . . . 18
1.3.3 Experimental Protocol . . . 19
1.4 Experimental Results . . . 20
1.4.1 Generated Hybrid SLS . . . 20
1.4.2 Comparison with the state-of-the-art algorithm . . . 22
1.5 Conclusion and Perspectives . . . 23

2 Multi-Objective SLS 27
2.1 Literature Review . . . 28
2.1.1 Extensions of Single-Objective SLS Algorithms . . . 28
2.1.2 SLS Techniques in Evolutionary Algorithms . . . 30
2.1.3 Pareto Local Search Algorithms . . . 31
2.1.4 Condensed Literature Summary . . . 33
2.1.5 Analysis and Discussion . . . 33
2.2 Identification of MO-SLS Strategies . . . 35
2.2.1 Set of Potential Pareto Optimal Solutions (Archive) . . . 35
2.2.2 Set of Current Solutions (Memory) . . . 36
2.2.3 Exploration Strategies . . . 36
2.2.4 Selection Strategies . . . 37
2.2.5 Termination Criteria . . . 37
2.2.6 Escaping Local Optima . . . 38
2.3 MO-SLS Structure . . . 38
2.3.1 Local Search Algorithm . . . 38
2.3.2 Local Search Exploration . . . 39
2.3.3 Iterated Local Search Algorithm . . . 40
2.3.4 Literature Instantiation . . . 40
2.4 Automatic Design using the Unified Structure . . . 41

2.4.1 Static MO-SLS Structure . . . 41
2.4.2 Adaptive MO-SLS Structure . . . 46
2.5 Conclusion and Perspectives . . . 46

3 AAC of Multi-Objective SLS 49
3.1 Multi-Objective AAC . . . 50
3.1.1 Definition . . . 50
3.1.2 MO-ParamILS . . . 50
3.2 Experimental Environment . . . 52
3.2.1 Unary Indicators . . . 52
3.2.2 MO-AAC vs. SO-AAC Approaches . . . 53
3.2.3 Configurators . . . 54
3.2.4 Permutation Problems . . . 54
3.3 Experiments 1: Interest of using a MO-AAC Approach . . . 57
3.3.1 Experimental Protocol . . . 57
3.3.2 Results and Analysis . . . 58
3.4 Experiments 2: Impacts of Objectives Correlation on AAC . . . 68
3.4.1 Experimental Protocol . . . 68
3.4.2 Results and Analysis . . . 69
3.5 Automatic Configuration of a Dynamic MO-SLS . . . 74
3.5.1 A Dynamic Algorithm Framework . . . 76
3.5.2 Automatic Configuration of our Framework . . . 76
3.5.3 Related Work . . . 77
3.5.4 Experimental Protocol . . . 77
3.5.5 Results and Analysis . . . 78
3.5.6 Evaluation of Dynamic MO-SLS . . . 78
3.5.7 Performance of the Dynamic *vs* Static MO-SLS . . . 80
3.6 Conclusion and Perspectives . . . 82

II Landscape-based Knowledge for Algorithm Design 85

4 Neutrality in Multi-Objective Fitness Landscapes 87
4.1 Background . . . 88
4.2 Indicator-based Definitions of Neutrality . . . 88
4.2.1 Neutrality based on Pareto Dominance . . . 88
4.2.2 Neutrality based on ε-indicator . . . 89
4.2.3 Neutrality based on HD-indicator . . . 93
4.3 Distribution of Neutral Neighbors in Permutation Problems . . . 95
4.3.1 Experimental Protocol . . . 96
4.3.2 Neighbors Distribution in Initial Partitions . . . 97
4.3.3 Neutral Neighbors Distribution according to the Bound . . . 97
4.4 Characterization of Promising Neutral Neighbors . . . 101
4.4.1 Definition of Promising Neutral Neighbors . . . 101
4.4.2 Methodology to Study the Distribution of the Promising Neutral Neighbors . . . 102

4.5 Distribution of Promising Neutral Neighbors in Permutation Problems . . . 104
4.5.1 Experimental Protocol . . . 104
4.5.2 Experimental results on FSP . . . 104
4.5.3 Experimental results on QAP . . . 106
4.5.4 Experimental results on TSP . . . 108
4.6 Conclusion and Perspectives . . . 109

5 Landscape-aware SLS **113**
5.1 Reduction of the Search Space . . . 114
5.1.1 The No-Wait Flowshop Scheduling Problem . . . 114
5.1.2 Super-jobs: Promising sequences of Consecutive Jobs . . . 115
5.1.3 Iterated Greedy with Learning . . . 118
5.1.4 Experiments . . . 119
5.1.5 IIG_{SJ}: an Iterative Version . . . 124
5.2 Reduction of the Neighborhood Size . . . 128
5.2.1 The Feature Selection Problem in classification . . . 128
5.2.2 The Feature Selection Problem with Learning Tabu Search . . . 131
5.2.3 Learning Tabu Search for Feature Selection . . . 133
5.2.4 Experiments . . . 134
5.3 Conclusion and Perspectives . . . 139

General Perspectives **143**

Appendix **146**

A Extended CV **147**

B Personal Bibliography after PhD **153**

Bibliographie **157**

Introduction

History & Research Activities

I did my PhD from 2008 to 2011 in the Laboratoire d'Informatique Fondamentale de Lille (LIFL) at Université Lille 1. My work focused on **combinatorial optimization** and **landscape analysis**. I mainly worked on neutral characteristics and proposed different ways to exploit this particularity into **stochastic local search** algorithms.

After my PhD, I obtained an ERCIM Alain Bensoussan fellowship to work with Thomas Stützle at Université Libre de Bruxelles (Belgium) on **automatic algorithm configuration** from September 2012 to August 2013. I collaborated with two postdoctoral researchers (Manuel López-Ibáñez and Franco Mascia) with whom we defined a generalized structure of stochastic local search that could automatically be instantiated using their grammar to describe algorithmic components. This common work was a real opportunity for me to learn from passionate people and to start building my research network since I continue to collaborate with Manuel on other topics.

In September 2013, I was hired as an Associate Professor in the LIFL – Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL) since 2015 – where I joined back my former research team. Although, this team is known for working on multi-objective combinatorial problems, I never worked on these problems. Naturally, I wanted to focus part of my research work on **multi-objective optimization**. With my novel knowledge on automatic algorithm configuration, we decided to work on a multi-objective approach to automatically configure multi-objective stochastic local search algorithms. This work was carried out with the collaboration of Holger Hoos from University of Leiden (The Netherlands) and is part of the PhD work of Aymeric Blot (2015-2018), I have co-supervised. Besides, I wanted to continue my PhD work on landscape analysis in order to extend the neutral characteristics to multi-objective optimization. In this way, we collaborated with Hernan Aguirre and Kiyoshi Tanaka from Shinshu University (Japan).

Alongside, I continued to work on single-objective problems. For example, I wanted to pursue my investigations on **landscape-based design** which was the thesis topic of another PhD student I co-supervised, Lucien Mousin (2015-2018). I met Myriam Delgado from the Federal University of Technology of Paraná (Brazil) in 2016. She was interested to work on **algorithm selection** using automatic algorithm configuration to find the optimal parameter setting of each considered algorithm and landscape analysis to extract features from each instance. This was a good opportunity to make these two fields meet. This work is currently carried out through the PhD of Lucas Pavelski that I co-supervised with Myriam.

Since my recruitment, I supervised undergraduate and Master's student internships and

some of them have continued to pursue a PhD under my supervision. A detailed list is given in Appendix in the extended CV. The contributions presented in this manuscript are derived from the articles and conference papers I wrote with my co-authors during these past eight years since I obtained my PhD. The following of this introduction will present the context of these recent works and give the organization of the manuscript.

Combinatorial Optimization

A combinatorial optimization problem (COP) consists in finding the optimal solution (or a set of optimal solutions) in a discrete (i.e. enumerable) space of feasible solutions of a given problem. In single-objective optimization, the notion of optimality is given by an objective function. A single-objective optimization problem involves minimizing, without loss of generality, an objective function $f : \Omega \rightarrow \mathbb{R}$ over a space of candidate solutions Ω, i.e., to determine

$$\arg\min_{s \in \Omega} \; f(s) \tag{1}$$

In multi-objective optimization, several criteria (or objective functions) characterizing the quality of solutions of a given problem are optimized simultaneously. A multi-objective optimization problem involves minimizing, without loss of generality, a vector of functions over a space of candidate solutions, i.e., to determine

$$\arg\min_{s \in \Omega} \; (f_1(s), f_2(s), ..., f_n(s)) \tag{2}$$

where n is the number of objectives ($n \geq 2$) and each function $f_i(s)$ has to be minimized. The concept of *Pareto dominance* is used to capture trade-offs between the criteria f_i: solution s_1 is said to dominate solution s_2 if, and only if, (i) s_1 is better than or equal to s_2 according to all criteria, and (ii) there is at least one criterion according to which s_1 is strictly better than s_2. A set S of solutions in which there are no $s_1, s_2 \in S$ such that s_1 dominates s_2 is called a *Pareto set* or a *Pareto front*. The goal when solving an instance of a multi-objective optimization problem is to determine the best such set, i.e., the set $S \subset \Omega$ such that there is no $s' \in \Omega$ that dominates any of the $s \in \Omega$.

Metaheuristics are widely used algorithms to solve large and complex multi-objective optimisation problems (Gendreau and Potvin 2010). In combinatorial optimization, the stochastic local search (SLS) algorithms (Hoos and Stützle 2004) are efficient metaheuristics used to solve $\mathcal{NP}$-hard problems and obtain good solutions in reasonable time for such problems. The *simple* and *hybrid* SLS algorithms manipulate only one single candidate solution of the given problem instance in each search step and move in the search space by iterating a neighborhood operator. We may cite the simulated annealing (Kirkpatrick et al. 1983), the tabu search (Glover et al. 1993), the variable neighborhood search (Mladenović and P. Hansen 1997) and the iterated local search (Lourenço et al. 2010). The SLS algorithms can also manage a population of candidate solutions which interact with each other. These SLS are often nature-inspired algorithms since they evolves in the search space following natural rules. We may cite evolutionary algorithms (Holland 1992) and ant colony optimization (Dorigo et al. 1996).

Most of the metaheuristics have been first designed for single-objective optimization but have been adapted to multi-objective optimization. However, performance assessment of multi-objective algorithm is not straightforward since Pareto sets of solutions are not easily comparable when all the solutions of a front do not dominate all the solutions of the other one. Binary indicators are then used, such as the epsilon indicator (ε) and the hypervolume difference indicator (HD) (Zitzler et al. 2003). The ε-indicator I_{ε^+} (resp. I_{ε^*}) gives the minimum distance (resp. value) by which a Pareto set approximation A needs to or can be translated (resp. divided) in each dimension in the objective space such that another Pareto set approximation B is weakly dominated. Formally, it is defined in a minimization context as follows:

$$I_{\varepsilon^+}(A,B) = min_{\varepsilon}\{\forall x_2 \in B, \exists x_1 \in A \mid f_i(x_1) - \varepsilon \leq f_i(x_2) \text{ for i } \in \{1,...,n\} \quad (3)$$

$$(\text{resp. } I_{\varepsilon^*}(A,B) = min_{\varepsilon}\{\forall x_2 \in B, \exists x_1 \in A \mid f_i(x_1)/\varepsilon \leq f_i(x_2)\})$$

The HD-indicator is based on the hypervolume concept and is defined for two sets A and B such as:

$$I_{HD}(A,B) = \begin{cases} I_{HD}(B) - I_{HD}(A) \\ \quad \text{if } \forall x_2 \in B, \exists x_1 \in A \mid x_1 \succ x_2 \\ I_{HD}(A+B) - I_{HD}(A) \quad \text{otherwise} \end{cases} \quad (4)$$

$I_{HD}(A)$ gives the hypervolume of the objective space dominated by A, and accordingly $I_{HD}(A,B)$ measures the volume of the space that is dominated by B but not by A with respect to a predefined reference point Z.

Both epsilon and hypervolume difference indicators have been proposed to compare two sets of solutions.

The works presented in this manuscript focus on combinatorial optimization problems and stochastic local search algorithms for both single- and multi-objective optimization.

Knowledge-based Design of Optimization Algorithms

Metaheuristics are generic and flexible algorithms able to adapt to any kind of optimization problems. Both genericity and flexibility are given by the available algorithmic strategies and their own parameters values. First, the generic design can be completed with problem-dependent mechanisms or heuristics. Second, while it is widely admitted that no single algorithm dominates all others on all problem instances, the metaheuristics have to be finely parameterized to perform well. Therefore learning process should be used to design metaheuristics adapted to the given problem instances.

In this manuscript, we will be interested in two different ways to address knowledge-based design of metaheuristics: automatic algorithm configuration and landscape-based knowledge algorithm design. I focused my research work on these two *independent* topics that, very recently, have started to meet each other.

Automatic Algorithm Configuration

Different research fields deal with automatic algorithm design. We may cite *algorithm configuration* (also called parameter tuning), *parameter control*, *algorithm selection* and *hyper-heuristics*. The first two expose only one algorithm while the last two consider various algorithms. Automatic algorithm configuration and algorithm selection are off-line processes while parameter control is an on-line process. The contributions on automatic algorithm design presented in this manuscript focus on algorithm configuration. Therefore, we will present automatic algorithm configuration in extended detail while we will only briefly present the other fields.

Algorithm configuration determines a parameter setting that optimizes the performance of a given algorithm for a given class of problem instances. In this context, we call the algorithm whose parameters are being optimized the *target algorithm* and the procedure that automatically configures the target algorithm a *configurator*. Automatic algorithm configuration (AAC) is a machine learning process, whose general concept is illustrated on Figure 1. It involves making a prediction of the optimal configuration of the target algorithm over a training dataset, usually relatively to a given running time or computational budget. The configurations resulting from this training are then re-evaluated on a disjoint test dataset to ensure the unbiasedness of the final prediction.

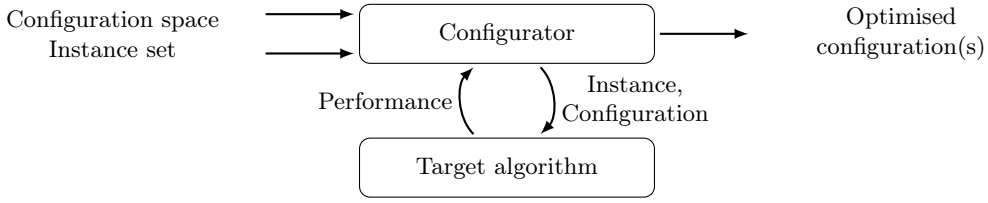


Figure 1: Automatic configuration of a given parameterized target algorithm for a given set of problem instances.

Given a configurable target algorithm A, a space Θ of configurations of A, a distribution of instances $\mathcal{D}$, a performance indicator $o : \Theta \times \mathcal{D} \to \mathbb{R}$, and a statistical population parameter E; the algorithm configuration problem consists in optimizing the aggregated performance of the target algorithm A across all instances $i \in \mathcal{D}$, as given in Equation 5 (in which A_θ denotes the algorithm obtained by associating the configuration θ to the target algorithm A).

$$\begin{cases} \text{optimize} & E[o(A_\theta, i),\ i \in \mathcal{D}] \\ \text{subject to} & \theta \in \Theta \end{cases} \tag{5}$$

Algorithm configuration supposes that the limit implied by Equation 5 exists and is finite. The most commonly used statistical parameter is the simple average of the performance of the target algorithm.
Among the configurators, we may cite the most commonly used: irace (López-Ibáñez et al. 2016) that is based on statistical racing, SMAC (Hutter et al. 2011), based on random forests, ParamILS (Hutter et al. 2009), based on iterated stochastic local search and, GGA+ (Ansótegui et al. 2015, 2009) based on a genetic algorithm and random forests.

Most of the configurators handle only one objective (e.g. the performance or the running time). However, SPRINT-race (Zhang et al. 2013) is a multi-objective configurator based on statistical racing where two metrics are simultaneously considered for model selection in machine learning. Morevoer, some multi-objective procedures have been also proposed and are based on evolutionary algorithms (Branke and Elomari 2012; Dréo 2009) where the performance and the runtime have to be optimize simultaneously.

Parameter control adapts the algorithmic components or parameter values of an algorithm during its execution. The underlying idea is that the behavior of the algorithm, and so its parameters, has to change during the run in order to better adapt to the local structure of the search space. Parameter control approaches use techniques such as multi-armed bandits (Fialho et al. 2009) or adaptive pursuit (Thierens 2005) to dynamically determine good parameter settings in response to observations made while trying to solve a given problem instance. However, the number of configurations that can be handled by such approaches is very limited (Desport et al. 2015; Tollo et al. 2015; Veerapen et al. 2012). The first taxonomy was proposed by Eiben et al. (1999). Karafotias et al. (2015) expanded upon the trends and the challenges of parameter control. B. Doerr and C. Doerr (2018) updated and completed the original taxonomy of Eiben et al. (1999) and presented a survey for discrete black-box optimization.

Algorithm selection investigates the relation between algorithm performance and problem instance features (Rice 1976). The principle is that, for a given set of problem instances, a set of complementary algorithms can be used to improve overall performance. This problem optimizes the performance of every instance of the set independently, then it is also called *per-instance* algorithm selection. In machine-learning, this process is known as meta-learning. The procedure that automatically selects the appropriate algorithm for a given instance is called a *selector*. Among others, we may cite SATzilla (Xu et al. 2008) and ISAC (Kadioglu et al. 2010). An extension of algorithm selection is the *per-instance algorithm scheduling* problem where a schedule of algorithms is output rather than one single algorithm. This process is preferred when instance features are not very informative (M. Lindauer et al. 2016). Despite the scarcity of multi-objective works, Horn et al. (2016) proposed a multi-objective approach where both performance and runtime are two important criteria of the optimization. Meanwhile, Kotthoff (2014) gave a comprehensive survey of algorithm selection for COPs which has recently been updated by Kerschke et al. (2019) and completed with works on numerical optimization problems.

A hyper-heuristic selects, combines, generates or adapts simpler heuristics to efficiently solve optimization problems. It is designed to handle classes of problems and be adapted to each single instance to solve. Therefore, it is often considered like on-line algorithm selection. Hyper-heuristics have been applied to a large variety of COPs (see the comprehensive survey of Burke et al. (2013)) and also in multi-objective optimization (Guizzo et al. 2017; Li et al. 2019).

During my postdoctoral research, I worked on automatic configuration of Stochastic Local Search (SLS) algorithms (see Chapter 1); and one of the objectives of my recruitment

at Université de Lille was to extend this work to the multi-objective optimization. We were convinced that configuring a multi-objective SLS is a multi-objective optimization problem. Chapter 3 presents our contribution in this field.

Landscape-based Knowledge for Algorithm Design

The concept of fitness landscape was first introduced in the literature by Wright (1932) in his work on the evolution of living beings. It consists in representing individuals in relation to their fitness and giving a geometric structure to the problem in order to understand the drifts of evolution. This concept was then transposed to many fields such as combinatorial or numerical optimization (Jones 1995; Manderick et al. 1991). In combinatorial optimization, and more precisely in single-objective optimization, fitness landscape is defined as a triplet $(\Omega, \mathcal{N}, f)$, where Ω represents the search space, $\mathcal{N} : \Omega \rightarrow 2^{\Omega}$ is the neighborhood relation that associates to each solution s of the search space Ω a set of solutions $\mathcal{N}(s)$, called neighbors and, $f : \Omega \rightarrow \mathbb{R}$ is an objective function that measures the quality of the solutions. This definition gives a geometrical structure to the search space which is based on the neighborhood relation and the fitness function. Indeed, the search space can be seen as a graph in which solution representations are nodes, neighborhood relation defines edges and fitness values are "heights" in the landscape (see Figure 2 (Left) in a maximization context).

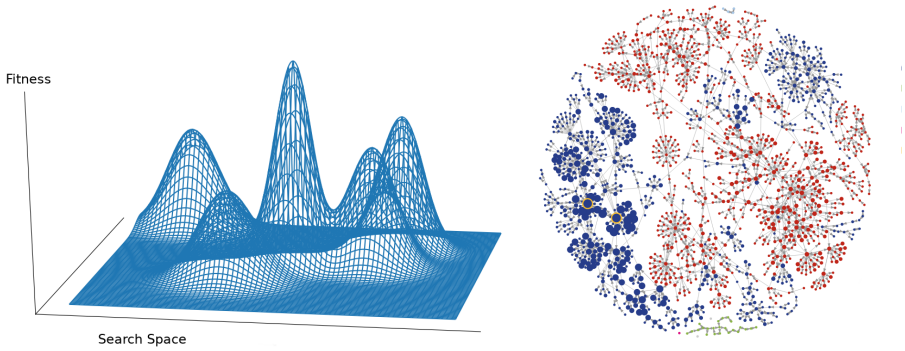


Figure 2: (Left) 3D Visualization of a maximization fitness landscape; (Right) Visualization of a LON (Ochoa and Veerapen 2016).

Different Fitness Landscape Analysis (FLA) measures and tools have been proposed to capture information about the structural properties of the problem instances. For example, the *autocorrelation* of the fitness landscape can be used to measure its ruggedness (Weinberger 1990). Other FLA measures based on information theory can be calculated from a random/adaptive walk, such as the *entropy* and the *density basin* (Vassilev et al. 2000). Another common measure is the *fitness-distance correlation* (Jones and Forrest 1995). *Neutrality* is a particular property of some landscapes that can be characterized with the neutral degree or the size of the plateaus (Marmion et al. 2013a, 2011b).

Ochoa et al. (2008) introduced the Local Optima Networks (LON), an oriented graph

where the nodes are local optima and the directed edges represents the possibility to escape from a local optima. It gives a better representation of the landscape (see Figure 2 (Right)). Recently, Vérel et al. (2018) proposed a methodology to build LONs for large search spaces where a sampling is mandatory. From these works, other metaphors appeared like the funnels (Herrmann et al. 2016) and the big valley hypothesis (Ochoa and Veerapen 2016). Due to the neighborhood relation, landscapes and even LONs are high-dimensional graphs that are difficult to represent. Recent works aims at giving some tools for better visualizing such landscapes (Ochoa et al. 2015; Veerapen and Ochoa 2018).

The behavior of stochastic local search algorithms in the search space depends on the neighborhood relation; the same applies for hybrid metaheuristics that integrate neighborhood-based mechanisms. Therefore, the landscape metaphor is used to help designers to understand the behavior of their algorithms. Kauffman (1993) introduced the NK-landscape family that gives artificial landscapes where the ruggedness can be tuned. Several works have studied the performance of algorithms on these specific landscapes like Merz (2004) for memetic algorithms, Daolio et al. (2012) for iterated local search algorithms and, Tari et al. (2017, 2018) for basic climbers. Permutation problems like the permutation flowshop scheduling problem (FSP) and the quadratic assignment problem (QAP) have also been investigated. Marmion et al. (2013b) studied FLA measures of two landscapes induced by different neighborhood relations and evaluated the efficiency of metaheurictics for both landscapes. Marmion et al. (2011a) studied the neutral property of the FSP with makespan minimization on the performance of an iterated local search, while Hernando et al. (2017) characterized the landscape of the FSP compared to the makespan or the sum of flowtimes minimization. Chicano et al. (2012) analyzed the LONs of the QAP and the performance of heuristic search algorithms. Basseur and Goëffon (2015) compare the behavior of classical basic local search techniques and their ability to reach high local optima for both NK-landscapes and permutation problems.

The works discussed below and most of the works on landscape analysis used FLA measures or tools to explain and understand the algorithm performance in an *a posteriori* analysis. However, an *a priori* or *on-line* landscape analysis can give some knowledge about the problem/instances to solve. This knowledge can be integrated in the design of an algorithm *before* the execution (Marmion et al. 2011a; Mousin et al. 2016, 2019) or directly *during* the execution. For example, Consoli et al. (2016) exploits fitness landscapes measures to dynamically select the operators of an evolutionary algorithm. Given the limited number of such works, designing algorithm with knowledge extracted from landscape analysis seems and is actually not straightforward and will be discussed in the conclusion of Chapter 5. Recently, Liefooghe et al. (2017b) used landscape characteristics in a racing-based configurator and improved its efficacy.

Contrary to single-objective COPs, very few works deal with multi-objective fitness landscapes. Knowles and Corne (2003) first proposed landscape features to characterize multi-objective QAP instances. Garrett and Dasgupta (2007) adapted the classical FLA measures to the multi-objective optimization. Garrett (2009) proposed a definition to plateaus being the solutions with the same rank with respect to the Pareto Local Optima. Similarly

to the analysis of LON, he studied the graph built from these plateaus. More recently, Vérel et al. (2013) gave a definition to multi-objective fitness landscapes and studied the impact of the correlations between the objectives in the structure of the decision space. Recent works analyzed the features of the landscape compared to the performance of multi-objective stochastic local search algorithms (Daolio et al. 2017; Liefooghe et al. 2017a). In order to better characterize multi-objective landscapes, we proposed an extension of the concept of neutrality to the multi-objective COP presented in Chapter 4.

Document Outline

This manuscript is divided into two parts: the first one deals with the automatic configuration of stochastic local search (SLS) algorithms and, the second one deals with the characterization of landscapes and the exploitation of features into stochastic local search algorithms. Both parts present works in single- and multi-objective optimization.

The first part, called *Automatic Algorithm Configuration*, is divided into three chapters:

In Chapter 1, we present a generalized structure that unifies classical SLS algorithms as well as powerful hybridizations. Using a grammar-based approach, we give a practical implementation to automatically generate efficient SLS. The performance of our approach is validated on the permutation flowshop scheduling problem.

In Chapter 2, we propose a comprehensive survey and a unification of SLS techniques in metaheuristics for multi-objective combinatorial optimisation. This generalized structure is able to instantiate all the prominant multi-objective SLS of the literature. We present a simpler multi-objective SLS template where numerous algorithmic components as well as control and feedbacks mechanisms can be integrated. It is the starting point of the works presented in the next chapter.

In Chapter 3, we present different contributions on multi-objective configuration of multi-objective SLS. First, the multi-objective AAC is defined as a multi-objective optimization problem. Experiments are conducted on three permutation problems (the flowshop scheduling problem, the traveling salesman problem and the quadratic assignment problem) and lead to validate our conviction that multi-objective SLS are better configured using a multi-objective approach. We also show that this statement remains true regardless of the correlation between the objectives. Second, we propose a novel approach to design dynamic SLS algorithms that modify their parameters during the execution following a schedule that is pre-defined with automatic algorithm configuration. The presented experiments are encouraging.

The second part, called *Landscape-based Knowledge for Algorithm Design*, is divided into two chapters:

In Chapter 4, we extend the concept of neutrality for multi-objective landscapes. The

proposed definitions are based either on the Pareto dominance or on two performance indicators namely the epsilon indicator and the hypervolume difference indicator. Moreover, a characterization of some neutral neighbors is proposed in order to identify those who would be the most promising to lead to new better solutions. The neutral property of three permutation problems is analyzed with respect to different correlations between the objectives.

In Chapter 5, we present two state-dependent SLS algorithms designed from an *a priori* analysis of the problem. First, a particular characteristic of good solutions is exploited into an iterated greedy algorithm in order to drastically reduce the size of the search space to solve a variant of the permutation flowshop scheduling problem. Then, in a context of expensive evaluation, feedbacks computed during the execution are exploited into a tabu search to reduce the size of the neighborhood that also extremely reduces the optimization runtime.

Part I

Automatic Algorithm Configuration

Chapter 1

Automatic Configuration of Hybrid Stochastic Local Search Algorithms

In this chapter, we propose a practical, unified structure that encompasses stochastic local search (SLS) algorithms and an automatic configuration approach able to generate powerful hybrid SLS algorithms. This work was carried out in collaboration with Thomas Stützle, Manuel López-Ibáñez and Franco Mascia during my postdoctoral contract at ULB (Université Libre de Bruxelles) in Belgium from September 2012 to August 2013.

Successful algorithms for hard combinatorial problems are often the result of an effective engineering of metaheuristics or of an appropriate combination of ideas originating from various such methods. However, despite the plethora of possibilities, algorithm designers often consider, only a few available methods when tackling a new problem. In our work, we proposed a semi-automatic system that, with little human effort, is able to generate powerful hybrid SLS algorithms (López-Ibáñez et al. 2013, 2014; Marmion et al. 2013c; Mascia et al. 2014).

This chapter is organized as follows:

- Section 1.1 presents our generalized local search (GLS) structure able to instantiate the classical SLS algorithms and hybrid SLS algorithms.
- Section 1.2 presents a practical implementation to easily generate efficient hybrid SLS for a given problem. Our approach is based on the grammar description of the GLS structure and the use of automatic algorithm configuration.
- Section 1.3 gives all the necessary ingredients to replicate the experiments.
- Section 1.4 analyzes the results obtained with our approach for designing a hybrid SLS for a variant of the permutation flowshop scheduling problem.

1.1 Generalized Local Search Structure

Many stochastic local search (SLS) algorithms manipulate a single solution at each of the search steps (Hoos and Stützle 2004). They may internally keep a memory of multiple solutions, such as the best solution found so far, but there is the concept of the *current solution*, whose neighborhood is being explored. Examples of such SLS methods (also called metaheuristics) include classical iterative best- and first-improvement algorithms (Papadimitriou and Steiglitz 1982), iterated local search (ILS) (Lourenço et al. 2010), simulated annealing (SA) (Kirkpatrick et al. 1983), variable neighborhood search (VNS) (Mladenović and P. Hansen 1997), random iterative improvement (RII) and probabilistic iterative improvement (PII) (Hoos and Stützle 2004), and iterated greedy (IG) (Ruiz and Stützle 2007), among others.

ILS Algorithm

1: $s_0 :=$ Initialization()
2: $s^* :=$ ILS(s_0)
3: **return** s^*

Function ILS(s_0)
Require: perturbation, ls, acceptanceCriterion, stop
1: $s^* :=$ ls(s_0)
2: **repeat**
3: $s' :=$ perturbation(s^*)
4: $s'' :=$ ls(s')
5: $s^* :=$ acceptanceCriterion(s'', s^*)
6: **until** termination criterion (stop) is satisfied
7: **return** s^*

Figure 1.1: The iterated local search (ILS) algorithm

We propose a generalized local search (GLS) structure modeled after iterated local search (ILS) (Lourenço et al. 2010). ILS, as shown in Fig. 1.1, starts from an initial solution s_0, applies an improvement method (usually referred as local search, ls), and then three steps are repeated until the termination criterion is met: the current solution is perturbed to generate a new one, local search is applied to the new solution, and the acceptance criterion (in the simplest case of acceptance criteria) accepts the new solution or stays with the current one. ILS contains the most important elements of any hybrid LS algorithm, which are a *perturbation* operator, a subsidiary *local search*, and an *acceptance criterion*. The *perturbation* is a transformation of the input solution. In ILS, this is typically a small random transformation of the solution but it may also be a random re-initialization. A perturbation may be one simple move in a neighborhood space, but it may also be composed of k applications of a simple move, and k may be even vary during the search either based on feedback of the search process or according to a pre-defined schedule. The *local search* can range from a simple iterative improvement over short runs of an SA algorithm to a full-fledged ILS. It could also be that no local search algorithm is used at all.

The *acceptance criterion* determines which solution will replace the current solution. The most basic acceptance criterion (*improveAccept*) accepts only solutions that are better (strictly or not) than the best solution found so far. Other acceptance criteria allow worse solutions to be accepted in order to increase the exploration of the search space. For instance, the *probAccept* criterion accepts a worsening solution with a probability $p \in [0, 1]$.

Table 1.1: Classical metaheuristics formulated as instances of the GLS template.

Name	Reference	Perturbation	Local Search	Acceptance Criterion
ILS	Lourenço et al. 2010	*any*	*any*	*any*
SA	Kirkpatrick et al. 1983	one move	none	Metropolis
PII	Hoos and Stützle 2004	one move	none	Metropolis (fixed temp.)
RII	Hoos and Stützle 2004	one move	none	Probabilistic
VNS	Mladenović and P. Hansen 1997	variable move	iterative improvement	Better
IG	Ruiz and Stützle 2007	destruct-construct	*any*	*any*

For example, if $p = 1$, every new solution is accepted (*alwaysAccept*). A *thresholdAccept* criterion accepts a worsening solution if the relative deviation between the best and the current solution is below a threshold. In simulated annealing, a worse solution is accepted according to the Metropolis criterion (*metropolisAccept*). This criterion uses a cooling schedule that starts from an initial temperature, the temperature is decreased according to the cooling schedule until the algorithm stops after reaching a final temperature. Often, the temperature is decreased periodically after a number of iterations (*span*).
By considering different alternatives for each of these components, we can replicate many of the SLS methods proposed in the literature. For instance, simulated annealing (SA) can be replicated by defining the perturbation operator as a simple move operator in a neighborhood, using no subsidiary local search, and using the Metropolis acceptance criterion. With these components, the scheme given for ILS above will actually replicate a classical SA algorithm.
Another example is variable neighborhood search (VNS) (Mladenović and P. Hansen 1997). VNS executes an iterative improvement method at each iteration, and varies the strength of the perturbation depending on whether the resulting solution improves the best so far. This is equivalent to an ILS with a specialized *variable move* operator, iterative improvement as local search, and an *improveAccept* acceptance criterion. Other classical SLS algorithms can be modeled after ILS in a similar manner, as shown in Table 1.1.
In addition to replicating these classical SLS algorithms, the GLS structure proposed here can also reproduce more complex combinations of SLS algorithms. For example, an ILS algorithm can use a different ILS algorithm (with different perturbation and/or acceptance criterion) as a subsidiary local search, which, in turn may use SA as its own subsidiary local search. We called *recursion* the possibility of an ILS to embed another ILS. The level of recursion is the number of embedded ILSs. This ability of combining simple components to generate hybrid local search algorithms allows designing powerful algorithms. However, it raises the question of how to find high-performing algorithms for a particular problem, among all the possible combinations.

1.2 Implementation

1.2.1 A practical implementation of the GLS structure

In this section, we describe how to implement the GLS structure proposed above in order to generate practical algorithms for a given problem. Our method consists of three parts. First, we use a generative grammar to describe the design space defined by the GLS struc-

ture. Second, we use a re-usable framework of source code components as the underlying implementation of the grammar. This implementation includes both problem-independent code, which can be re-used in any problem, and problem-specific components, which must be developed for each problem. Finally, we use an automatic algorithm configuration tool to search the configuration space and generate high-performing instantiations of the grammar, given a set of training instances representative of the problem.

1.2.2 A grammar description of the GLS structure

A practical implementation of the GLS structure should contain many components that interact. Implementing such a GLS structure as a unique monolithic algorithm is a complex task. Moreover, the fact that a local search can be embedded within another in arbitrary ways complicates such implementation. The alternative that we proposed was to implement only the individual components, with clearly defined interfaces, and directly generate specific algorithms by combining these components. This is a typical problem in genetic programming, where grammars are often used to represent the design space of an algorithm (McKay et al. 2010).

A grammar is a set of derivation rules that describes how the symbols in a language can be combined to produce valid sentences. In our case, the valid sentences are local search algorithms encoded in `C++`, but for clarity we will describe the algorithms in pseudo-code. Fig. 1.2 shows the grammar that describes the GLS structure proposed in the previous section. Each line is a production rule of the form `<non-terminal> ::=`*expression* that describes how the non-terminal on the left-hand side can be replaced by the expression on the right-hand side. Expressions may contain terminal and/or non-terminals. Alternative expressions are separated with the symbol "`|`". The non-terminal symbol `<algorithm>` defines the starting point for instantiating an algorithm from the grammar.

The first three rules in the grammar describe the main structure of the GLS structure proposed earlier (see Fig. 1.1). The next three rules describe the basic components of our GLS structure, that is, the perturbation operator (`<perturb>`), local search (`<ls>`), and acceptance criterion (`<accept>`). Since the rule `<ls>` can expand to `<ils>` which contains again `<ls>`, a local search can be embedded within another local search (recursion). The other rules describe the alternatives available for the various components. Our grammar explicitly contains classical local search algorithms, but defined in terms of ILS, as detailed in Table 1.1. Moreover, the grammar also allows problem-specific components (`<pbs_...>`), which can be implemented for each problem tackled.

The possibility of adding problem-specific components is an advantage of our proposed method. Such components are critical for the success of SLS algorithms. For example, in this way problem specific construction and destruction mechanisms can be incorporated and be used in the destruction/construction phase (`<deconst-construct_perturb>`) of an IG algorithm. Hence, our grammar must account for such components. A practical implementation of our method also requires to define other problem-specific components in order to describe the representation of the problem, neighborhood operators, the objective function and how to read an instance of the problem. For simplicity, we do not include these in our exposition, but they are implemented in a similar fashion.

Finally, each ILS in the proposed grammar has its own termination criterion (`<stop>`),

```
     <algorithm> ::= <initialization> <ils>
<initialization> ::= random | <pbs_initialization>
           <ils> ::= ILS(<perturb>, <ls>, <accept>, <stop>)

<perturb> ::= none | <initialization> | <pbs_perturb>
     <ls> ::= <ils> | <descent> | <sa> | <rii> | <pii> | <vns> | <ig> | <pbs_ls>
 <accept> ::= alwaysAccept | improvingAccept  <comparator>
            | prob(<value_prob_accept>) | probRandom | <metropolis>
            | threshold(<value_threshold_accept>) | <pbs_accept>

<descent> ::= bestDescent(<comparator>, <stop>)
            | firstImprDescent(<comparator>, <stop>)
    <sa>  ::= ILS(<pbs_move>, no_ls,  <metropolis>, <stop>)
    <rii> ::= ILS(<pbs_move>, no_ls, probRandom, <stop>)
    <pii> ::= ILS(<pbs_move>, no_ls, prob(<value_prob_accept>), <stop>)
    <vns> ::= ILS(<pbs_variable_move>, firstImprDescent(improvingStrictly),
                  improvingAccept(improvingStrictly), <stop>)
     <ig> ::= ILS(<deconst-construct_perturb>, <ls>, <accept>, <stop>)

<comparator> ::= improvingStrictly | improving
<value_prob_accept> ::= [0, 1]
<value_threshold_accept> ::= [0, 1]
<metropolis> ::= metropolisAccept(<init_temperature>, <final_temperature>,
                                  <decreasing_temperature_ratio>, <span>)
 <init_temperature> ::= {1, 2,..., 10000}
<final_temperature> ::= {1, 2,..., 100}
<decreasing_temperature_ratio> ::= [0, 1]
<span> ::= {1, 2,..., 10000}
```

Figure 1.2: A simplified view of the grammar for the GLS structure.

which is typically a maximum computation time. If there is more than one level of ILS algorithms, the total computation time must be divided among them, such that the inner level does not consume all available time. We adopt here a simple scheme. The top-level ILS stops once the total time is consumed. Each subsequent level stops after consuming a ratio of the time allocated to its parent ILS. This ratio is controlled by a parameter $t_{\text{ls}} \in \{0.1, 0.2, \ldots, 1\}$ for each level of ILS.
These ratios have to be tuned in order to generate an efficient hybrid SLS algorithm.
In practice, an instantiation of the grammar produces an algorithm that is mapped to source code implementing the individual components. In our case, the implementation of the components is done using Paradiseo-MO (Humeau et al. 2013), an open-source `C++` framework whose purpose is to facilitate the design of metaheuristics by providing a library of reusable components. The idea is that an algorithm designer can re-use the available algorithm components or implement her own components, and freely combine these components to design new algorithms.
Our proposal goes a step beyond this idea, since in our proposed method the algorithm designer can focus on implementing problem-specific components, while the grammar takes care of describing the possible algorithm designs given the available components. The next section describes how to automatically find a high-performing SLS algorithm for a given problem, among all the possible algorithm designs that can be generated from the grammar.

1.2.3 Automatic generation of hybrid LS metaheuristics

Given a particular problem, our goal is to find the highest-performing instantiation of the grammar given above. As mentioned above, techniques such as genetic programming (McKay et al. 2010) and grammatical evolution (Burke et al. 2012) are often used

for this task. Before our work, Mascia et al. (2013) showed how to instantiate IG algorithms from a grammar by means of a parametric representation. The use of a parametric representation has certain advantages and enables the use of state-of-the-art automatic configuration tools for offline parameter tuning. In that work, they showed that a parametric representation produced better IG algorithms than the representation used by grammatical evolution. In our joint work, we explore the much larger space of SLS algorithms defined by the proposed GLS structure.
We follow the method described in the previous work (Mascia et al. 2013) to generate a parametric description of the grammar. This requires defining a maximum limit to the number of ILS levels in the final algorithm, that is, a maximum number of applications of the rule `<ls>` in the grammar. This limit has an influence on the number of parameters required to describe the grammar. In the next section, we explore the effect of this limit on the results.
From the parameter description and given a set of training instances representative of the problem, we apply an automatic configuration tool to search the space of possible algorithm designs. Here, we use irace (López-Ibáñez et al. 2011). Nonetheless, any automatic configuration tool that handles large numbers of categorical, numerical and conditional parameters with complex constraints would be appropriate.
Each parameter configuration tested by irace is an instantiation of the grammar, which is mapped to `C++` code and compiled into an executable. This executable is then run on various training instances by irace in order to determine its performance. The irace procedure stops after exhausting a given budget of algorithm runs, and it returns the SLS algorithm configuration that it identified as the best performing one during the tuning.

1.3 Experimental Environment

1.3.1 The PFSP-WT

We tested our proposed method on the permutation flowshop scheduling problem with weighted tardiness (PFSP-WT). The permutation flowshop scheduling problem (PFSP) encompasses a variety of problems that are typical of industrial production environments. The common goal of various PFSPs is to schedule N jobs on M machines with the condition that all jobs must be processed in the same order and jobs are not allowed to pass each other. Each job i requires, on each machine j, a fixed, non-negative processing time p_{ij}.
In the PFSP-WT, we are asked to determine a schedule that minimizes the total weighted tardiness. Each job i has a due date d_i, which denotes the desired completion time of the job on the last machine, and a priority weight w_i, which denotes its importance. The *tardiness* of a job i is defined as $T_i = \max\{C_i - d_i, 0\}$, where C_i is the completion time of job i on the last machine, and the *total weighted tardiness* is given by $\sum_{i=1}^{n} w_i \cdot T_i$. This problem is $\mathcal{NP}$-hard even for a single machine (Du and Leung 1990).

1.3.2 Local search components for the PFSP-WT

To the best of our knowledge, a state-of-the-art algorithm for the PFSP-WT was proposed by Dubois-Lacoste et al. (Dubois-Lacoste et al. 2009). This algorithm, henceforth called soa-IG, is an iterated greedy algorithm that works as follows. An initial solution

```
<pbs_initialization> ::= NEH | NEH-WSLACK
<pbs_perturb> ::= <deconst-construct_perturb>
                | <perturb_move>(<k>)
                | var_<perturb_move>(<k>)
<perturb_move> ::= insert | swap | exchange
<k> ::=  {1,2,...,10}
<deconst-construct_perturb> ::= soa_ig_perturb(<d>)
<d> ::= {1,2,...,10}
<pbs_ls> ::= soa_ig_ls

<pbs_variable_move> ::= var_<pbs_move>(<k>)
<pbs_move> ::= insert
<pbs_accept> ::= soa_ig_accept(<Tc>)
<Tc> ::= [0,1]
```

Figure 1.3: The extended grammar for the PFSP-WT.

is constructed using a modified version of the well-known NEH algorithm (Nawaz et al. 1983) called NEH-WSLACK where the WSLACK heuristic provides the initial order for the NEH algorithm, and the jobs are inserted in the solution in the order that minimizes the partial objective function, i.e., computed using the jobs present in the partially constructed solution. The local search in soa-IG is a first-improvement descent using a swap neighborhood and with a maximum number of swaps, fixed to $2 \cdot (N-1)$, where N is the number of jobs. The perturbation operator consists in removing d jobs randomly from the solution. These jobs are re-inserted one by one to minimize the partial objective function. Finally, in the acceptance criterion, a new solution that is worse than the current one is accepted with a probability given by $\exp(100 \cdot (f(\pi) - f(\pi'))/(f(\pi) \cdot T_c))$, where T_c is a user-defined parameter, $f(\pi)$ is the objective value of the current solution and $f(\pi')$ is the objective value of the new one. Dubois-Lacoste et al. (2009) suggest the settings $d = 5$ and $T_c = 1.2$.

We added the aforementioned components to the grammar of our GLS structure as additional problem-specific components (Fig. 1.3). In particular, we add two initialization methods, NEH with and without the WSLACK heuristic (`NEH` and `NEH-WSLACK`). In addition to the random destruction-construction perturbation used by soa-IG, we added further problem-specific perturbations based on classical neighborhood move operators (insert, exchange and swap) and a strength parameter k that controls the number of random moves applied per perturbation. The value of k may be fixed or vary during the run (`var_`) as in VNS. The problem-specific local search used by SOA is added to the grammar (`soa_ls`). Moreover, the `pbs(_variable)_move` used in the grammar (see Fig 1.2) are set to the insert move. Note that, the descents also use the insert move to define the neighborhood. Finally, we add the acceptance criterion of soa-IG as an additional acceptance criterion.

1.3.3 Experimental Protocol

We assess the potential of the proposed method by generating three hybrid SLS algorithms for the PFSP-WT, and comparing them with soa-IG. In particular, we generate three algorithms (ALS1, ALS2, and ALS3) for tackling the PFSP-WT from our GLS structure, by allowing different levels of recursion.

The procedure for generating these three algorithms is as follows. We consider the grammar presented in Fig. 1.2 and the PFSP-WT-specific extensions discussed above (Fig. 1.3). For each level of recursion, we automatically generate a parameter description. Indeed,

the recursion leads to an increasing number of parameters. With one level of recursion, i.e., a single ILS, the grammar is represented by 80 parameters. Of these 80 parameters, 27 are categorical and represent possible algorithmic choices, 25 are integer-valued, and 28 are real-valued. With two or three levels of recursion, the number of parameters increases to 127 and 174, respectively.
The parameter description is given to irace together with a number of training instances. As training instances, we generated 10 random PFSP-WT instances for each number of jobs in $\{50, 60, 70, 80, 90, 100\}$ and with 20 machines, following the procedure described by Minella et al. (Minella et al. 2008). Within irace, a specific algorithm, i.e., a specific parameter configuration, is evaluated by running it on a training instance with a time limit of 30 CPU-seconds. A single run of irace stops after exhausting a given budget of evaluations. Since the number of parameters is different according to the level of recursion, we used different budgets for the different runs of irace; concretely, 30 000 evaluations for generating ALS1, 40 000 for generating ALS2, and 50 000 for generating ALS3.
The three algorithms ALS1, ALS2 and ALS3 generated by irace are then run on a set of test instances of size `50x20` and `100x20`, different from the set of training instances. Also soa-IG is run on the same instances. To avoid differences due to implementation details, we have instantiated soa-IG as one specific SLS algorithm through our grammar, taking care that the algorithm is implemented correctly in this way. These test instances were generated by Minella et al. (Minella et al. 2008) from well-known PFSP instances (Taillard 1993). Each run is repeated 30 times with different random seeds.

1.4 Experimental Results

1.4.1 Generated Hybrid SLS

The three algorithms (ALS1, ALS2 and ALS3) generated by irace are shown in Fig. 1.4. The first one (ALS1) is an IG algorithm within a classical ILS. It uses the NEH-WSLACK initialization, then executes a classical ILS with a k-`insert` move as perturbation, IG as the subsidiary local search, and an `improving` acceptance criterion. The IG has a time limit of $0.8 \cdot maxTime$, and it is represented by an ILS with the construction/deconstruction operator of soa-IG as the perturbation, a first-improvement descent as the subsidiary local search, and the IG acceptance criterion. The first-improvement descent has a time limit of $0.5 \cdot 0.8 \cdot maxTime$. (Note that the first-improvement descent will actually terminate much before its maximum time limit upon finding a local optimum; in fact, the time limits mentioned here and in the following do actually not restrict the computation times of iterative improvement algorithms.)

ALS2 is a VNS algorithm included in an ILS that is itself included in an ILS. ALS2 uses the NEH initialization, then executes a classical ILS without perturbation, an ILS as the subsidiary local search, and an `improving` acceptance criterion. The subsidiary ILS has a time limit of $0.8 \cdot maxTime$, again no perturbation, a VNS as the subsidiary local search, and a `Metropolis` acceptance criterion. The VNS has a time limit of $0.4 \cdot 0.8 \cdot maxTime$, and it is represented as an ILS with a *variable* `insert` move perturbation, a first-improvement descent as the subsidiary local search, and the `improvingStrictly` acceptance criterion.

The first-improvement descent has a time limit of $0.4 \cdot 0.4 \cdot 0.8 \cdot maxTime$.

```
ALS1 Algorithm: ILS(IG)
  s_0 := NEH-WSLACK()
  s* := ILS(perturb_move_insert(k = 6),
            ILS(soa_ig_perturb(d = 9),
                firstImprDescent(strict,
                                 t_ls = 0.5),
                soa_ig_accept(T_c = 0.8956),
                t_ls = 0.8)
            improvingAccept,
            t_ls = maxTime)
  return s*

ALS2 Algorithm: ILS(ILS(VNS)) :
  s_0 := NEH()
  s* := ILS(perturb_none,
            ILS(perturb_none,
                ILS(variable_move_insert(k = 1),
                    firstImprDescent(strict,
                                     t_ls = 0.4),
                    improvingStrictlyAccept,
                    t_ls = 0.4),
                metropolisAccept(1548, 56, 0.7447, 7401),
                t_ls = 0.8),
            improvingAccept,
            t_ls = maxTime)
  return s*

ALS3 Algorithm: ILS(ILS(VNS)) :
  s_0 := NEH-WSLACK()
  s* := ILS(perturb_move_exchange(k = 7)),
            ILS(soa_ig_perturb(d = 5),
                ILS(variable_move_insert(k = 3),
                    firstImprDescent(strict, t_ls = 0.3),
                    improvingStrictlyAccept, t_ls = 0.4),
                metropolisAccept(4969, 48, 0.8356, 8954), t_ls = 0.8),
            alwaysAccept, t_ls = maxTime)
  return s*
```

Figure 1.4: Hybrid LS algorithms automatically generated for PFSP-WT.

ALS3 is also a VNS algorithm included in an ILS that is itself included in an ILS. Although three levels of recursion were allowed when generating ALS3, this algorithm only has two levels as ALS2. ALS3 uses the NEH-WSLACK initialization, then executes a classical ILS with a k-`exchange` move as perturbation, an ILS as the subsidiary local search, and an acceptance criterion that always accepts a new solution. The subsidiary ILS has a time limit of $0.8 \cdot maxTime$, and uses the construction/deconstruction operator of soa-IG as the perturbation, a VNS as the subsidiary local search, and a `Metropolis` acceptance criterion. The VNS has a time limit of $0.4 \cdot 0.8 \cdot maxTime$ and it is represented as an ILS with a *variable* `insert` move perturbation, a first-improvement descent as the subsidiary local search, and the `improvingStrictly` acceptance criterion. The first-improvement descent has a time limit of $0.3 \cdot 0.4 \cdot 0.8 \cdot maxTime$ seconds.

1.4.2 Comparison with the state-of-the-art algorithm

To assess the performance of the three automatically generated algorithms, we run them 30 times on the test instances and compare them with soa-IG. Fig. 1.5 and 1.6 show the solution cost reached by each algorithm on each instance. Table 1.2 gives the best and mean solution. The behavior of the algorithms is slightly different depending on the instance size. The performance of the automatically generated SLS algorithms on the `50x20` instances matches the quality obtained by soa-IG in most instances, and they are noticeably better on a few. On the `100x20` instances, the automatic SLS algorithms clearly outperform soa-IG.

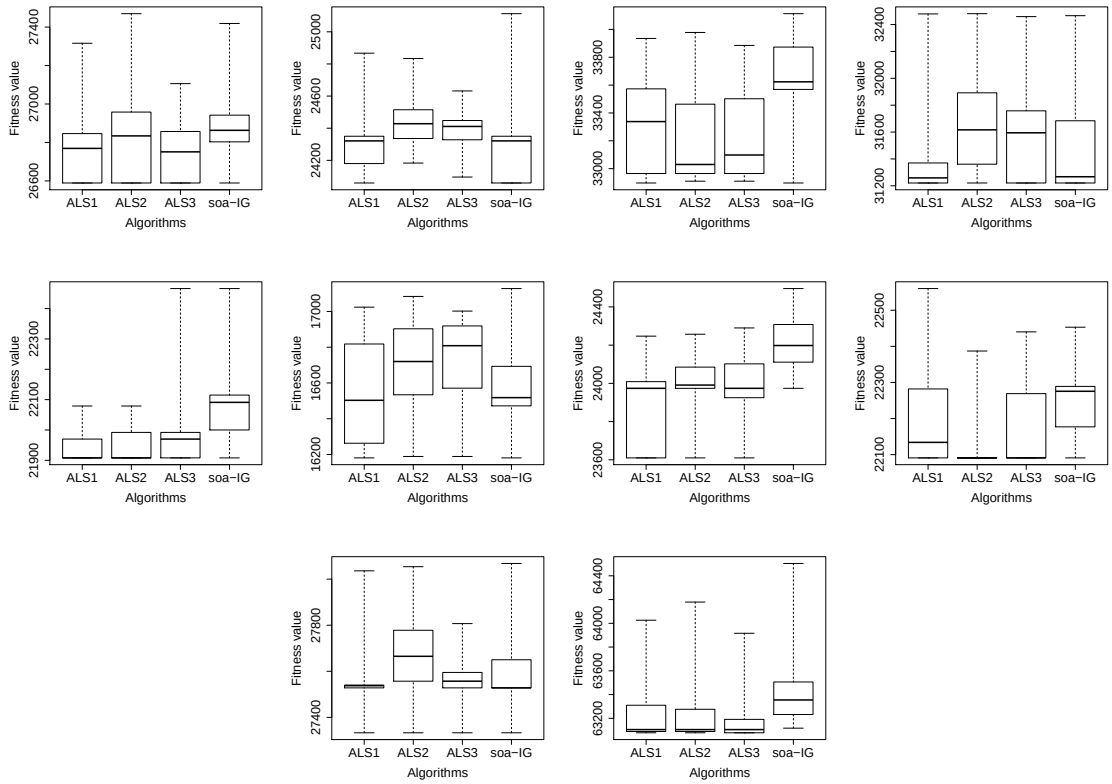


Figure 1.5: Solution costs obtained by the three automatic SLS algorithms (ALS1, ALS2 and ALS3) and soa-IG on the texttt50x20 instances.

In order to assess the performance over each set of instances, we perform a statistical analysis based on the Friedman test for analyzing non-parametric unreplicated complete block designs, and its associated post-hoc test for multiple comparisons (Conover 1999). First, we pair the runs performed on the same instance using the same random seed. This is the blocking factor, and the different algorithms are the treatment factor. Algorithms are ranked within each block, lower solution cost corresponds to lower rank. If the Friedman test rejects the hypothesis that the different algorithms obtain the same mean rank, then we calculate the difference (ΔR) between the sum of ranks of each algorithm and the best ranked one (with the lowest sum of ranks). We also calculate the minimum difference

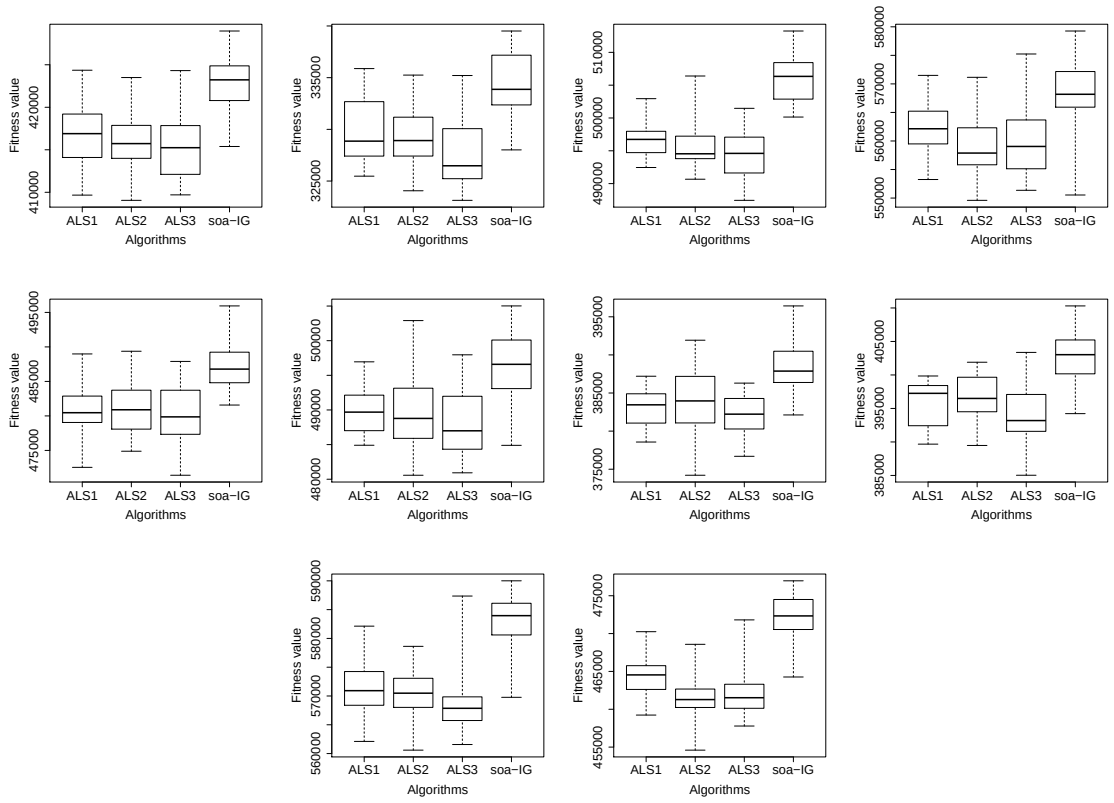


Figure 1.6: Solution costs obtained by the three automatic SLS algorithms (ALS1, ALS2 and ALS3) and soa-IG on the `100x20` instances.

between the sum of ranks of two algorithms that is statistically significant (ΔR_α), given a significance level of $\alpha = 0.05$. Table 1.3 gives the results of this analysis, applied separately to the two sets of instances of size `50x20` and `100x20`. We indicate in bold type the best strategy (the one having the lowest sum of ranks) and those that are not significantly different from the best one. In both cases, the best ranked algorithm is significantly better than the rest. However, the best algorithm is different in each case. Notably, soa-IG is consistently ranked as the worst by a large margin, especially on the `100x20` instances. These results are consistent with the observations above. Therefore, our conclusion is that the current state of the art can be matched and outperformed by the automatically generated algorithms.

1.5 Conclusion and Perspectives

Summary In this chapter, we have shown that the process of designing hybrid stochastic local search (SLS) algorithms can become mostly automatic. In particular, we have proposed a unified and practical generalized local search (GLS) structure. We have shown that the GLS structure unifies the formulation of various simple SLS methods and their possible combinations (hybridizations) into a single structure. In fact, the best algorithms

Table 1.2: Solution costs obtained by the three automatic SLS algorithms and soa-IG on the test instances.

Instances		ALS1		ALS2		ALS3		soa-IG	
		Best	Avg	Best	Avg	Best	Avg	Best	Avg
50x20	ta051	26589	26806.2	26589	26824.9	26589	**26756.1**	26589	26899.6
	ta052	24059	**24273.2**	24183	24443.2	24096	24390.8	24059	24333.5
	ta053	32897	33307.8	32910	**33183.7**	32910	33206.8	32897	33634.6
	ta054	31221	**31470.2**	31221	31663.3	31221	31572.7	31221	31488.9
	ta055	21908	**21936.2**	21908	21948.3	21908	21975.9	21908	22094.7
	ta056	16181	**16516.4**	16189	16711.6	16189	16740.7	16181	16556.7
	ta057	23610	**23869**	23610	23990.4	23610	23953.9	23974	24211.2
	ta058	22091	22207.7	22091	**22131.9**	22091	22166.8	22091	22262.1
	ta059	27333	**27521.3**	27333	27685.1	27333	27573.1	27333	27577.9
	ta060	63078	63286.3	63078	63235.9	63078	**63179.9**	63117	63456
100x20	ta081	409667	416932.6	409052	415941.5	409697	415306.3	415388	422625
	ta082	325472	329803.8	324060	329161.3	323133	**327466.6**	328014	334437.1
	ta083	492455	496922.6	490669	495669.7	487450	**494569.8**	500142	505772
	ta084	553249	562380.8	549600	**558824.5**	551359	559419.9	550536	568600.5
	ta085	472546	480861	474883	481147.7	471402	**479941.3**	481576	487291.6
	ta086	484905	490357.9	480575	489379	480926	**488144.7**	484892	496511.1
	ta087	378567	382931.6	374208	384024.5	376694	**382277.9**	382122	388511.8
	ta088	389673	395809.4	389475	396729.7	385029	**394056.2**	394226	402836.5
	ta089	562109	571495.2	560593	570489.5	561570	**568465.7**	569769	582829.9
	ta090	459232	464206.4	454597	**461262.9**	457784	462177.3	464264	471961.3

Table 1.3: Statistical analysis based on the Friedman-test. The second column gives the minimum difference in the sum of ranks that is statistically significant (ΔR_α), given a significance level of $\alpha = 0.05$. For each instance set, algorithms are ordered according to the rank obtained. The numbers in parenthesis are the difference of ranks relative to the best algorithm. The algorithm that is significantly better than the other ones is indicated in bold face.

Instances	ΔR_α	Algorithms (ΔR)
50x20	57.92	**ALS1**, ALS3 (75), ALS2 (115.5), soa-IG (221.5)
100x20	47.04	**ALS3**, ALS2 (100), ALS1 (143), soa-IG (573)

generated when applied to the PFSP-WT are complex hybrids that combine ILS with IG, VNS and even a different ILS. Our proposal is also practical, in the sense that it generates algorithms that are as efficient as if they were hand-crafted by a competent programmer. Two properties of our proposal are key for obtaining such efficiency. First, instead of a complex algorithmic framework with many parameters, our system generates specific algorithms from a grammar description of the GLS structure. These specific algorithms, which contain only a small fraction of all the algorithmic components available in the grammar, are generated directly as C++ code and compiled. Second, our grammar description allows algorithm designers to include problem-specific components, which are often crucial for obtaining high-performing SLS algorithms. The system takes care of combining, testing and selecting (or discarding) these problem-specific components among all the available algorithmic components. The evaluation of this proposal has been performed on the permutation flowshop scheduling problem with weighted tardiness (PFSP-WT). Our

experimental results showed that the three automatically generated SLS algorithms are able to outperform the state-of-the-art algorithm on well-known PFSP-WT instances from the literature.

Extension of the GLS structure Additional SLS algorithms, for example, greedy randomized adaptive search procedure (GRASP) (Feo and Resende 1995) and Tabu Search (Glover et al. 1993) were integrated later for solving other hard combinatorial problems. We may cite the traveling salesman problem with time windows (López-Ibáñez and Blum 2010) and the unconstrained binary quadratic programming problem (Kochenberger et al. 2004), a non-linear combinatorial optimization problem that unifies a wide range of important problems including graph coloring, set partitioning, and maximum cut problems. Our approach also proved its ability to configure an efficient hybrid SLS. This work is continued by Thomas Stützle with PhD students where better implementations of both the components and the whole approach, are also studied (Franzin and Stützle 2019; Pagnozzi and Stützle 2019).

Extension to the Multi-objective Optimization This postdoctoral work allowed me to learn the field of automatic algorithm configuration (AAC). When I came back to Université de Lille, I wanted to extend this work to multi-objective optimization which was in tune with the ORKAD team expertise. The contributions presented in the rest of this part, result from this initial work. Chapter 2 presents a unified structure of multi-objective SLS algorithms. Chapter 3 gives a definition to multi-objective AAC and presents the results obtained on different configurations of multi-objective SLS.

Chapter 2

Multi-Objective Stochastic Local Search Algorithms

In this chapter, we consider multi-objective stochastic local search (MO-SLS) algorithms. Although SLS algorithms sometimes have been directly used as single-objective procedures of multi-objective algorithms, many multi-objective local search algorithms have also been specifically designed.

During the PhD of Aymeric Blot, we focused on these multi-objective local search techniques. We proposed a new basic local search structure (Blot et al. 2018b) that unifies most, if not all, multi-objective local search algorithms of the literature, and discussed a number of strategies that can be integrated inside this structure. This MO-SLS structure has been used in different way in our works on automatic design (Blot et al. 2018a, 2017a, 2018c, 2019, 2017c; Pageau et al. 2019). It showed its practicality in tuning MO-SLS or in integrating control mechanisms into MO-SLS.

This chapter is organized as follows:

- Section 2.1 presents a survey of the use of local search techniques in multi-objective algorithms. We distinguished two types of multi-objective local search algorithms: the direct extensions of well-known single-objective local search algorithms, and the others that have been designed together with evolutionary algorithms. Nevertheless, both types of local search algorithms share a common underlying structure.
- Section 2.2 details the basic components of our unification representing the sets of solutions and the strategies. Several values for each components are given.
- Section 2.3 describes our unification of the multi-objective local search structure and gives the instantiation of the most prominent multi-objective local search algorithms.
- Section 2.4 shows the ability of our structure to be tuned or to integrate control mechanisms. A static and an adaptive MO-SLS structure are proposed and discussed.

2.1 Literature Review

Historically, local search algorithms have been initially designed to solve single-objective combinatorial optimization problems and, therefore, are themselves single-objective algorithms (see Chapter 1). Multi-objective stochastic local search algorithms are used on the same combinatorial problems, e.g., multi-objective permutation problems (Basseur and Burke 2007; Dubois-Lacoste et al. 2015; Jaszkiewicz 2002; Liefooghe et al. 2012), and bioinformatics problems (Abbasi et al. 2015). The majority of the literature works focuses on bi-objective and tri-objective problems, while very fewer works tackle more than three objectives simultaneously. This is due to the nature of the induced search space; indeed, in these *many-objective* problems (Ishibuchi et al. 2008) solutions are much more often incomparable to each others, thus majorly hindering the neighborhood exploration of MO-SLS algorithms.
The development of MO-SLS algorithms has occurred simultaneously from two different directions. On the one hand, they were directly extended from well-known and established single-objective algorithms (e.g., (Czyzak and Jaszkiewicz 1996; M. P. Hansen 1997; Serafini 1994; Ulungu et al. 1995)). On the other hand, they were either integrated into evolutionary algorithms as inner components or used as post-processing algorithms (e.g., (Ishibuchi and Murata 1996; Knowles and Corne 1999; Talbi et al. 2001)). Nowadays, the prominent MO-SLS algorithms in the literature have grown into the Pareto Local Search (PLS) algorithms, which are derived from the second type of MO-SLS algorithms. In the following, we detail chronologically the development of these two algorithmic families before summarizing their common characteristics.

2.1.1 Extensions of Single-Objective SLS Algorithms

Since local search algorithms have been originally designed for single-objective optimization, they are single-trajectory algorithms, meaning that they follow a single solution (i.e., the *current* solution). Unsurprisingly, the first MO-SLS algorithms were extensions of these single-objective local search algorithms.
Simulated Annealing (SA) (Kirkpatrick et al. 1983) is a local search procedure that optimises a single solution, using a decreasing parameter, the temperature, to slowly converge to the global optimal solution. Serafini (1994), Fortemps et al. (1994) and Ulungu et al. (1995, 1999) have independently proposed the same algorithm, *Multi-Objective Simulated Annealing* (MOSA). Like in the original single-objective algorithm, a single *current* solution is considered and moved through the search space, while a subsidiary set is used to store the potential Pareto optimal solutions. The current solution is updated by evaluating a single random neighbor and potentially accepting it with regard to rules based on probabilities, which are themselves based on whether the neighbor dominates, is dominated by or is incomparable to the current solution, and on weighted projections of the fitness function. Czyzak and Jaszkiewicz (1996, 1998) proposed the *Pareto Simulated Annealing* (PSA), in which, rather than a single current solution, a set of current solutions is used to converge into multiple optima at the same time. The diversity of having multiple current solutions is also used to guide the parallel searches in diverse directions. Engrand (1998) and then Suppapitnarm and Parks (1999) proposed another MOSA variants, in which the current solution is periodically replaced by one of the archived solutions (SMOSA). Other

variants also include the *Pareto Archived Simulated Annealing* (PASA) by Suresh and Mohanasundaram (2004), the *Archived Multi-Objective Simulated Annealing* (AMOSA) by Bandyopadhyay et al. (2008) and those based on both Pareto Dominance (PDMOSA) and Weights (WMOSA) proposed in literature reviews by Suman (2003) and Suman and Kumar (2006).

Tabu Search (TS) (Glover et al. 1993) is a stochastic local search algorithm that uses an auxiliary set of solutions, the *tabu list*, to guide the search and escape local optima by preventing a *backward* move on the search space by banning the acceptance of neighbors too similar to recent considered solutions. In a TS local search, when a solution is explored, each of the solution's neighbors is evaluated and the best non-tabu one is selected to replace the current solution, even if it has a worse quality. The first multi-objective algorithm based on TS is the *Multi-Objective Tabu Search* (MOTS) proposed by M. P. Hansen (1997). It uses a set of *current* solutions and independently explores their neighborhoods using an aggregation of the multiple objectives. After a given number of iterations, a *drift* strategy is applied: the set of solutions is updated by replacing one of the solutions by another one, both uniformly selected at random, to explore the whole front and not merely to focus on one part of the objective space. Other TS algorithms have been proposed, such as the one by Baykasoglu et al. (1999), which is based on a local search with an intensification memory to restart from when no more improving move is available, or the two algorithms proposed by Jaeggi et al. (2004, 2008), based on the Hooke and Jeeves move (MOTS) and path-relinking (PRMOTS), respectively. Multi-objective variants of scatter search using TS and path-relinking have also been proposed (Beausoleil 2001; Molina et al. 2005).

Greedy Randomised Adaptive Search Procedure (GRASP) for Maximum Independent Set is a procedure originally presented by Feo et al. (1994) for single-objective problems. It has been extended by Vianna and Arroyo (2004) for multi-objective optimization problems (GRASP-MULTI). Both algorithms are multi-start metaheuristics that alternate two phases, the first being the construction of an initial solution, and the second being the iterative improvement of that solution through a local search procedure; the best overall solution (or set of solutions, for the multi-objective version) is then returned. The local search procedure simply replaces the current solution by any improving neighbor until there is no more. In the case of the multi-objective version, the different local search iterations use different aggregation weights and a list of all potential Pareto optimal solutions is automatically kept up to date. An extensive survey of multi-objective GRASP can be found in (Martí et al. 2015), in which the authors propose a multi-objective GRASP with path-relinking.

Lastly, Variable neighborhood Search (VNS) (Mladenović and P. Hansen 1997) is a local search algorithm that solves the *local* optima problem by simply considering multiple other neighborhoods. Indeed, a PLO regarding a given neighborhood may have dominating neighbors regarding other neighborhoods. Geiger (2008) proposed a Multi-Objective Variable Neighborhood Search (MOVNS) based on PLS-2 and the VNS methodology. Like in PLS, an archive is used to store all potential Pareto optimal solutions. In every iteration, both a solution and a neighborhood, if they have not been explored yet, are selected uniformly at random, and then the solution is entirely explored and the Pareto set is updated using all the neighbors. Arroyo et al. (2011) proposed an interesting alternative to the MOVNS algorithm by adding a *shaking* mechanism: instead of generating the neigh-

borhood of a solution of the Pareto set, their algorithm generates the neighborhood of a neighbor of the solution of the Pareto set.

2.1.2 SLS Techniques in Evolutionary Algorithms

In addition to the single-objective SLS algorithms, the development of MO-SLS algorithms also occurred at the same time jointly with the multi-objective evolutionary algorithms.
Evolutionary Algorithms (EAs) constitute a class of metaheuristics based on the iterative improvement of a set of solutions (namely, the *population*), which is often used to tackle multi-objective optimization problems. EAs usually iterate both crossover and mutation techniques to improve the population. There are two types of hybridization of multi-objective EAs with local search mechanisms. The first type integrates the SLS inside the EA, either complementing or replacing the mutation, by using a SLS on every solution of the population at each iteration. The local search is then generally a single-objective algorithm, based on an aggregation of the different objectives. The second type uses a MO-SLS as a post-processing procedure at the end of the EA.
Ishibuchi and Murata (1996) first proposed the *Multi-Objective Genetic Local Search* (MOGLS), which hybridizes a genetic algorithm with a single-objective SLS by performing a SLS on every solution generated at every iteration. During the SLS, at most k neighbors of the current solution are produced and any improving neighbor is accepted. The crossover and the mutation strategies generate new solutions where the SLS is applied using a new aggregation, i.e., the weights are randomly chosen. A *cellular* variant, called C-MOGLS, was proposed by Murata et al. (2000), which divides solutions into cells associated with weight vectors to guide the selection and local search procedures of the MOGLS. Jaszkiewicz (2002) proposed another Genetic Local Search (GLS) where the main differences are the way the solutions are selected for recombination and the SLS aggregation, which uses a weight vector selected at random from a set of possible weight vectors.
Knowles and Corne (1999, 2000a) proposed the *Pareto Archived Evolutionary Strategy* (PAES), an EA without crossover that only relies on local search techniques. PAES was presented as "the simplest non-trivial approach to a multi-objective local search procedure", and three versions were introduced. All versions maintain a single *current* solution to be explored, while the population takes the role of a Pareto set. In the simple (1+1)-PAES, the current solution is explored by generating a single neighbor. The neighbor replaces the current solution either if it dominates the latter or if it is in a less crowded region of the population. The population itself is updated in such a way that any dominated solution is discarded and the solutions of less crowded spaces replace the solutions of more crowded spaces. The (1+λ)-PAES variant generates λ neighbors of the current solution at every iteration, which are all considered for updating the population and replacing the current solution. The (μ+λ)-PAES variant replaces the current solution with a list of μ current solutions, one of which is explored, selected using a binary tournament. Knowles and Corne (2000b) also proposed the memetic-PAES (M-PAES), a variant periodically employing a crossover.
Talbi et al. (2001) also proposed a genetic algorithm hybridized with a MO-SLS procedure. The execution of the algorithm is divided into two separate steps, beginning with the execution of a genetic algorithm. Once the GA finishes, every solution of the final

population is then explored by generating and archiving every possible non-dominated neighbor, a procedure that is then iterated until the end of the algorithm.
Similarly, Moslehi and Mahnam (2011) proposed a hybridization of a *Multi-Objective Particle Swarm Optimisation* algorithm with a single-objective SLS (MOPSO+LS).

2.1.3 Pareto Local Search Algorithms

Following the steps of algorithms such as PAES (Knowles and Corne 1999) and PLS-2 (Talbi et al. 2001), which are based on Pareto dominance and use the population of the EA as a Pareto set, many more algorithms solely based on SLS techniques have been designed that are not simple extensions of known single-objective SLS algorithms. These simple extensions can still be seen as either single-trajectory algorithms or *multiple*-trajectory algorithms, as multiple solutions are simultaneously iteratively improved. On the contrary, the notion of trajectory is more blurred in the following algorithms, which are based more on improving iteratively the full archive (originally the evolutionary population) than on focusing on single solutions.
Paquete et al. (2004) and Angel et al. (2004) simultaneously proposed the first standalone SLS algorithms: the *Pareto Local Search* (PLS) and the *Bi-criteria Local Search* (BLS). Both algorithms are very similar and are both known as PLS (or PLS-1 in comparison to the PLS-2 algorithm, the SLS algorithm of the EA proposed by (Talbi et al. 2001) that is sometimes referred to under this name). Unlike in the previous EAs, in PLS algorithms, a *population* is called an *archive* and always consists of a Pareto set. At every iteration of the PLS algorithm, a single solution not yet considered is taken from the archive to explore its neighborhood and all of its neighbors are used to update the archive.
Aguirre and Tanaka (2005) proposed the *multiple multi-objective random bit climbers* (moRBC), which also follows a SLS scheme. At each iteration, all the possible moves of the neighborhood are generated. They are all successively applied to the current solution, which can be immediately replaced when a dominating neighbor is found. These authors proposed multiple versions of moRBC wherein incomparable neighbors may be accepted and a separate archive may be used for crowding or restarting purposes.
Using the idea of employing a separate standalone procedure to generate the initial solutions of the SLS, Paquete and Stützle (2003) proposed the Two-Phase Local Search procedure (TPLS) for bi-objective optimization problems. First, an initial solution is generated (originally, using a SLS), considering the first objective only. Then, a SLS is performed, starting from the resulting solution, but using an aggregation of the objectives slightly more oriented towards the second objective. This step is then repeated until the final local search considers the second objective only. Many variants of the TPLS procedure have been proposed, among which is the 2-Phase Pareto Local Search (2PPLS) procedure by Lust and Teghem (2010), which hybridises the first step by constructing *potentially extreme supported efficient solutions* as the initial set of a PLS algorithm and an adaptive version of TPLS, likewise hybridised with a PLS algorithm, by Dubois-Lacoste et al. (2011).
Instead of using the Pareto dominance or an aggregation-based comparison, the Indicator-Based Multi-Objective Local Search (IBMO-SLS) (Basseur and Burke 2007; Basseur et al. 2012) accepts neighbors that are better than any solution of the population, by using a

binary multi-objective indicator, such as the hypervolume indicator (Zitzler and Thiele 1999). The population size of IBMO-SLS is fixed, the worse solution being replaced as soon as a new neighbor is accepted. The authors also proposed an iterative version of IBMO-SLS, in which the new initial Pareto set is obtained by applying random noise to a given number of solutions of the Pareto set. If the Pareto set is not big enough, additional solutions randomly generated are considered.

Drugan and Thierens (2012) proposed a multi-restart version of PLS with the Iterated PLS (IPLS). IPLS follows the PLS-2 algorithm, but associates with every solution a Boolean flag that is turned off after the solution neighborhood is explored. When all solutions are flagged, the search first restarts from a new solution that is randomly generated. After a given number of PLS runs, instead of considering a new solution, IPLS uniformly selects at random a solution from the archive, applies a mutation and restarts from the resulting solution.

Two separate generalisations of the PLS algorithms have since been independently proposed: the Dominance-based Multi-objective Local Search (DMLS) (Liefooghe et al. 2012) and the Stochastic Pareto Local Search (SPLS) (Drugan and Thierens 2012). The DMLS generalisation uses an archive of solutions and includes multiple strategies related to the selection of solutions to explore and to the exploration of the neighborhood. $DMLS(\alpha \cdot \beta)$ denotes that the DMLS uses the selection strategy α (with $\alpha \in \{1, \star\}$ for the selection of a single random solution and all solutions, respectively) and the exploration strategy β (with $\beta \in \{1, 1_{\not\prec}, 1_{\succ}, \star\}$ for the acceptance of a single neighbor at random, a single non-dominated neighbor at random, a single dominating neighbor at random and all neighbors, respectively). The SPLS generalisation also uses an archive of solutions from which at each iteration a solution is selected uniformly to be explored. Similarly, multiple exploration strategies are discussed. Furthermore, like in IPLS, a Boolean flag is associated with each solution to avoid exploring it multiple times and to enable faster termination and restarts when the exploration is not performed exhaustively. Indeed, the aforementioned authors also proposed a more generic process to restart PLS algorithms, together with a hybrid genetic PLS algorithm. Moalic et al. (2013) also proposed the Fast Local Search (FLS), which behaves like SPLS in that the exploration of a solution neighborhood stops as soon as a neighbor not dominated by the archive is found. Tricoire (2012) also proposed the *multi-directional local search* (MDLS), loosely based on PLS, in which at every iteration a solution from the archive is taken at starting point of a subsidiary local search, before merging the resulting archives by filtering dominating solutions.

The anytime behaviour of PLS algorithms has been investigated by Dubois-Lacoste et al. (2012, 2015), who proposed variants that optimize not just the quality of the final archive only, but also the quality of intermediate archives. They proposed the *Optimistic HyperVolume Improvement* (OHVI), an alternative mechanism for selecting the solution of the archive whose neighborhood will be explored, and, more importantly, they showed that changing the exploration strategy during the search could improve the performances of the PLS algorithm.

Finally, Inja et al. (2014) proposed the *Queued Pareto Local Search* (QPLS), another restart scheme using a queue to avoid premature convergence. Starting from the initial solutions, QPLS recursively explores every solution of the queue by using dominating neighbors to finally obtain a single final solution. If this final solution is not dominated

by the archive, it is merged and k incomparable neighbors are added to the queue. The authors also proposed the *Genetic Queued Pareto Local Search* (GQPLS), which hybridizes genetic algorithm techniques to update the queue.

2.1.4 Condensed Literature Summary

All of the MO-SLS algorithms outlined above share a common structure, in which a Pareto set of solution is iteratively improved by considering either a solution or a set of solutions as *current*, which is then explored to merge some or all of their neighboring solutions to the Pareto set. Table 2.1 summarizes the main SLS algorithms in the literature, according to the five following local search attributes.

Current solutions A single current solution or a current set of multiple solutions is used by the local search.

Archive The local search keeps track of a separate current set, or the current solutions can be directly selected from the archive.

Neighborhood Exploration A single neighbor, the full neighborhood or only a subset of the neighborhood (if a stopping criterion is used) is evaluated.

Acceptance criterion Incomparable and dominated neighbor may be accepted and returned after the neighborhood exploration, either as the stopping criterion of the exploration or in addition to the final neighbor.

Quality The comparison of the quality of two neighbors is done by considering either an aggregation or the Pareto dominance.

Reference During the neighborhood exploration, neighbors are compared either to the current solution or to other solutions such as the full Pareto set.

In Table 2.1, an "X" means that the algorithm possesses the corresponding characteristic, possibly depending of the context during the resolution (e.g., SA algorithm accepting dominated solutions by means of the temperature), whereas a "C" means that the characteristic is only present in some particular variant of the algorithm (e.g., the DMLS structure is able to instantiate many different MO-SLS algorithms).

2.1.5 Analysis and Discussion

Table 2.1 shows a trend between the two algorithmic families of the MO-SLS algorithms, where extensions of single-objective SLS algorithms generally separate the archive and the current solutions and use aggregations, and the family of the PLS algorithms, which generally directly select the current solutions from the archive and use Pareto dominance. One of the apparent weakness of MO-SLS algorithms relates to the possible number of solutions included in the archive and thus the size and shape of the optimal Pareto front. Indeed, if a MO-SLS algorithm does not use any mechanism to bound the size of its archive, exploration of too many solutions (and furthermore exhaustive neighborhood explorations) can become prohibitively computationally expensive slowing the convergence

Table 2.1: Condensed Literature Summary

"X": the algorithm possesses the given characteristic
"C": the algorithm may be configured to possess the given characteristic

	Current		Archive		neighbors			Acceptance Criterion		Quality		Reference	
Algorithm	Single current solution	Multiple current solutions	Separate archive and current solution	Current solution(s) from the archive	Single neighbor explored	Partial neighborhood exploration	Full neighborhood exploration	Accept if incomparable	Accept if dominated	Aggregation	Pareto dominance	Compare to current solution	Compare to Pareto set
MOSA	X		X		X			X	X	X		X	
PSA		X	X		X			X	X	X		X	
MOTS		X	X				X	X		X		X	X
MOVNS	X			X			X	X			X		X
MOGLS	X			X	X			X		X			X
PAES	X	C	X		C	C		X		X		X	
PLS-2		X		X			X	X			X		X
PLS-1	X			X			X	X			X		X
moRBC	X		X	X		X		C			X	X	
IBMO-SLS		X		X		X		X	X				X
DMLS	C	C		X	C	C	C	C			X	C	C
SPLS	X			X	C	C	C	C			X	C	C
FLS	X			X	X			X			X		X

MOSA (Fortemps et al. 1994; Serafini 1994); PSA (Czyzak and Jaszkiewicz 1996); MOTS (M. P. Hansen 1997); MOVNS (Geiger 2008); MOGLS (Ishibuchi and Murata 1996); PAES (Knowles and Corne 1999, 2000a); PLS-2 (Talbi et al. 2001); PLS-1 (Angel et al. 2004; Paquete et al. 2004); moRBC (Aguirre and Tanaka 2005); IBMO-SLS (Basseur and Burke 2007); DMLS (Liefooghe et al. 2012); SPLS (Drugan and Thierens 2012); FLS (Moalic et al. 2013)

of the algorithm to a halting point (Liefooghe et al. 2012). MO-SLS are similarly much weakened when using too large neighborhood, especially when explorations are performed exhaustively. Another current weakness of MO-SLS algorithms is that there is usually no explicit handling of the intensification/diversification trade-off. If some works focus on preserving diversity at the cost of some convergence speed (Blot et al. 2015), in most of the MO-SLS algorithms only intensification is rewarded and diversification is delegated as a side-effect of the archiving process. Furthermore, some variants of MO-SLS algorithms may require long computational time to reach high-quality approximations of the Pareto fronts and result on poor solutions if stopped early. Anytime mechanisms for MO-SLS algorithms have been proposed to deal with this particular limitation (Dubois-Lacoste et al. 2015).
The two DMLS and SPLS generalizations can be configured to instantiate a large range of Pareto local search strategies, but are not compatible with many extensions of single-objective strategies (and do not claim to be). The first fundamental limitation is that these generalizations do not use an explicit set of current solutions that is conveyed through the iterations of the local search, but instead select new current solutions from the archive every iteration. This also implies that the current solutions are always non-dominated. They can, through the use of an activation/deactivation scheme, emulate to some extent some trajectory-based strategies by keeping track of the selection of the previous iteration, but without the flexibility of keeping a separate set of current solutions, which allows, for example, to easily perform explorations outside their current archive (e.g., to explore dominated neighbors or when the algorithm allows some deterioration of the current solutions). The second main limitation is that the use of an archive as the main set of solutions leads to the use of the Pareto dominance (or a weakened version) for quality comparison, which leaves out the use of scalar-based comparisons in the exploration procedure.
To overcome these limitations, to allow more flexibility and to incorporate more diverse strategies, we propose a new MO-SLS generalization, which is detailed in Sections 2.2 and 2.3. Its main characteristics are the use of two explicit sets of solutions (namely, the set of current solutions and the archive), the separation of acceptance and stopping criteria in the exploration strategy, the possibility of using a simple set and not a Pareto set for the set of current solutions, the possibility of using scalar-based acceptance criteria and, finally, the use of an explicit reference during neighborhood comparisons.

2.2 Identification of MO-SLS Strategies

In this section, we describe different sets and strategies of the MO-SLS algorithms through examples from the literature review of the previous section. They are the basic components of our unification of MO-SLS that is presented in Section 2.3.

2.2.1 Set of Potential Pareto Optimal Solutions (Archive)

The *archive* is the Pareto set at the core of all MO-SLS algorithms. It holds potential Pareto optimal solutions, i.e., solutions not yet dominated by any other found solutions. This is the set of solutions finally returned by the procedure.
Depending on the problem considered, the size of the archive can become very large. Unless

this size is kept unbounded, a mechanism such as a diversity criterion (e.g., crowding, relaxed dominance, etc.) or a basic filtering mechanism may be used to remove the less important potential Pareto optimal solutions once a given size is reached (Liefooghe et al. 2012).

2.2.2 Set of Current Solutions (Memory)

In addition to the archive, the *current set*, a second set of solutions, is used to keep all the solutions whose neighborhood may be explored. These solutions are taken either from the archive or from previous iterations and may possibly be dominated by some solutions of the archive. To avoid using the same term (i.e., *current*) for both the current set and the current solutions it contains, we propose to call this set *memory*.
We identified three categories of strategies concerning the usage of the memory. First, as a direct extension of the single-objective local search algorithms, the memory can contain a single current solution (e.g., MOSA algorithm (Ulungu et al. 1999)). Iteration after iteration, the current solution is explored, potentially replaced by one of its neighbors, while the archive is automatically updated. If the current solution appears to be a Pareto Local Optimum, a restart can then be performed from one of the other potential Pareto optimal solutions. However, considering a single current solution means focusing on a single trajectory in the search space, whereas the multi-objective setting requires optimizing the whole Pareto front. Thus, the second category of strategies includes algorithms that keep a set of multiple *current* solutions and explores it sequentially, with the direct consequence of an improved diversity since each of the separate trajectories can then focus on the subset of the Pareto front (e.g., PSA (Czyzak and Jaszkiewicz 1998), MOTS algorithms (M. P. Hansen 1997), etc.). Finally, the third category includes algorithms that do not keep track of the trajectory, but rather directly select and explore solutions from the archive (e.g., PAES (Knowles and Corne 1999), PLS (Paquete et al. 2004), DMLS algorithms (Liefooghe et al. 2012), etc.).
Note that, like in the archive, the size of the memory may become very large, and, therefore, the same bounding mechanisms may be used. However, as such mechanisms were proposed for algorithms in which the memory and the archive were joined, it may be advantageous to bound only the memory and keep the archive unbounded.
We may envision a new exploration strategy where multiple solutions could be explored at the same time by combining their neighborhoods. In that case, without loss of generality, the *current* object would be itself a set of solutions and the *memory* would be a set of sets of solutions.

2.2.3 Exploration Strategies

The exploration of the current solution consists in the construction of its neighborhood, i.e., the generation of its neighbors.
Like in the single-objective case, two types of exploration strategy are distinguished: the *best improvement strategy* and the *first improvement strategy*. The *best* strategies compare every neighbor to the current solution or to the reference so that only the best non-dominated neighbors are accepted. On the contrary, the *first* strategies generate neighbors one by one and stop when a given stopping criterion is reached. Of course, the latter

strategies are not limited to stopping after a single accepted neighbor. In both the *best* and the *first* strategies, the exploration procedure generates some neighbors, accepting some of them, and then returns the set of *accepted* neighbors. For each of these neighbors, three questions arise: (i) Should it be included into the archive? (ii) Should it replace the current solution? (iii) Should the exploration continue or stop in regard to its quality?
The quality of a neighbor can be a function of either the current solution or a part or the totality of the archive. In multi-objective optimization, the objective space is divided into dominating solutions, incomparable solutions and dominated solutions compared to a *current* solution (these region of the objective space are defined in Chapter 4). The dominated solutions of the *current* solution are generally ignored, whereas exploration strategies usually consider solutions in the two other area. Considering incomparable solutions potentially enables to make better-informed decisions, however the main drawback is the added cost (e.g., computational time) of an overall more expensive exploration procedure.
An alternative to using the Pareto dominance criterion is to aggregate the objectives, to obtain a scalar value subsequently used to either rank neighbors or compute probabilities. The weights of the aggregation can be either globally set, associated with the current solution or updated automatically in regard to the state of the archive.
The archive (the set of potential Pareto optimal solutions) can be updated directly either during the exploration of a current solution or after the exploration of all current solutions has been performed. In the direct update, the explorations of the remaining current solutions may be impacted, i.e., the reference set is modified on the fly.
Similarly, the memory (the set of current solutions) can be updated during the exploration to replace the current explored solutions (e.g., in trajectory-based local search algorithms (Czyzak and Jaszkiewicz 1996; Fortemps et al. 1994; M. P. Hansen 1997; Serafini 1994)) or to include promising new neighbors directly (Blot et al. 2015).
If the memory contains multiple solutions, they are all explored before the search continues unless an early stopping criterion is met. Note that, if multiple solutions are explored and either the memory or the archive is updated during the exploration, the order in which the solutions of the memory are explored can strongly impact the performance.

2.2.4 Selection Strategies

After the exploration step has been completed, the solutions of the memory will have been explored and the archive will have been updated with the accepted neighbors. The memory has to be updated for the next iteration. Generally, the solutions are taken from the archive (e.g., randomly, with regard to a crowding or sharing property (Deb 2001), to an individual contribution (Dubois-Lacoste et al. 2012) or to the order of insertion in the archive (Blot et al. 2017a)). However, in trajectory-based local search algorithms, the memory is unchanged since it has been updated during the previous exploration step.

2.2.5 Termination Criteria

The local search has a natural termination criterion, which is reached when the memory becomes empty, meaning that no more solution is to be explored. Such an event generally means that every solution of the archive is a Pareto local optimum. This situation

also arises when the algorithm intentionally removes partially explored solutions from the memory, for example, to force a quick convergence or ensure diversification. Other commonly used termination criteria include the whole computational time; the total number of iterations, explorations or evaluations; and the number of successive iterations without improvement.

2.2.6 Escaping Local Optima

In single-objective optimization, SLS algorithms are generally trapped in local optima. However, various mechanisms (e.g., SA, TS, etc.) can be used to converge further towards a global optimum. Likewise, the basic instantiations of the procedures detailed in this chapter will generally be trapped in sets of Pareto local optimum. Likewise, the same various mechanisms can be and have been adapted for MO-SLS procedures to converge further towards the set of Pareto optima.
First, a temperature can be used to compute the probabilities of accepting neighbors of lesser quality (Czyzak and Jaszkiewicz 1996; Fortemps et al. 1994; Serafini 1994). This temperature can be either a global parameter of the local search or a specific temperature that can be associated with each and every solution of the memory when the local search follows a set of solutions of fixed size. The Tabu paradigm can also be used to drive the search out of the PLO (M. P. Hansen 1997). Similarly, a global tabu list or a set of tabu list can be used for each followed solution. Finally, it is possible to use an iterated SLS scheme (Drugan and Thierens 2012) to stop the SLS early, before reaching a true set of PLO. In this case, a convergence condition is defined as, for example, a threshold in the convergence rate or a stagnation criterion. The search can then restart either from the new solutions selected uniformly in the search space or from the solutions in the close neighborhood of the current or the best solutions, using a *kick*. In the single-objective case, a kick consists in taking a solution, either the current one or the best one, and performing a given number of random moves over the search space. In the multi-objective case, some solutions are selected (either a single one, a fixed number or a ratio of solutions, or all of them) from the memory or the archive; a single-objective kick is performed on each of them, and the resulting solutions are included in a new Pareto set, and then the algorithm restarts from it.

2.3 MO-SLS Structure

From the basic components presented in Section 2.2, we define a unified structure of MO-SLS algorithms that can instantiate the algorithms in the literature (see Section 2.1) and, we hope future designs.

2.3.1 Local Search Algorithm

Procedure 1 (LS) describes the main loop of the local search. This procedure takes an initial current set and an archive as input and returns the updated archive. It consists in iterating three steps (the names in parentheses are the names of the sub-procedures described as they appear in Procedure 1).

Procedure 1: LS(`memory`, `archive`)

Input: `memory`, a set of solutions to generate neighborhoods
Input: `archive`, a Pareto set of solutions
Output: the updated `archive` set

```
repeat
    all_accepted ← ∅;
    repeat
        let current ∈ memory;
        ref ← REFERENCE(current, memory, archive, all_accepted);
        accepted ← EXPLORE(current, ref, archive);
        memory ← UPDATE(memory, current, accepted);
        all_accepted ← all_accepted ∪ accepted;
    until iteration stopping condition is met
       or every current ∈ memory has been considered;
    archive ← COMBINE(archive, all_accepted);
    memory ← SELECT(memory, archive, all_accepted);
until local search stopping condition is met
   or memory = ∅;
return archive;
```

1. First, the solutions of the memory are explored one by one: for each, a reference is chosen to compare the neighbors with (REFERENCE), then some or all of the neighbors are accepted as candidates (EXPLORE), and, finally, the memory may be updated with the neighbors (UPDATE).

2. When all the current solutions have been explored, or when an early stopping condition is met, all accepted neighbors are used to update the archive (COMBINE). Note that it is possible to update the archive during the exploration, in which case the COMBINE procedure can still be used to bound its size.

3. Finally, the memory is set up with the new solutions to explore.

These three steps are iterated until the memory is empty or as soon as a given stopping condition is met.

2.3.2 Local Search Exploration

The exploration mechanism (EXPLORE) is described in Procedure 2. This procedure handles how neighboring solutions are generated and accepted, and how the reference set is updated. It takes as input a solution to explore, which is used to generate the neighborhood; a reference set to compare the neighbors with; and the archive of the SLS. It returns a set of accepted neighbors of the input solution and possibly modifies the archive as a side effect.
The neighbors of the current solution are generated one by one, and for each new neighbor, the set of accepted neighbors is updated (ACCEPT). To implement some local search

Procedure 2: EXPLORE(`current`, `ref`, `archive`)

Input: `current`, a solution to generate the neighborhood
Input: `ref`, a set of solutions to compare neighbors with
Input: `archive`, a Pareto set of solutions
Output: `accepted`, the set of accepted solutions
Side effect: modifies the `archive` set

```
accepted ← ∅;
repeat
    let neighbor ∈ 𝒩(current);
    accepted ← ACCEPT(accepted, neighbor, ref);
    current, ref, archive ← UPDATE(ref, accepted, current, archive,
     neighbor);
until exploration stopping condition is met
   or every neighbor ∈ 𝒩(current) has been considered;
return accepted;
```

algorithms from the literature, it is possible to immediately update the current solution, the reference set and the archive (UPDATE). neighbors are generated until every possible neighbor of the current solution has been generated or as soon as a given stopping condition is met.

2.3.3 Iterated Local Search Algorithm

The local search of Procedure 1 (LS) can eventually stop because either the archive contains only PLO or an early stopping condition has been met. One of the possible mechanisms to iterate the local search (LS) and continue the search is described in Procedure 3 (ITER). It follows the Iterated Local Search (ILS) scheme (Drugan and Thierens 2010, 2012; Lourenço et al. 2010) where the final archive given by the local search is slightly modified and given again as input to the local search procedure.
First, the local search is performed once, which sets up `archive`*, the Pareto set that contains the overall best non-dominated solutions across local search iterations. Then, until the global stopping condition is met, new initial memory and archive are generated (PERTURB), subsequent local search are performed and the two archives are combined to update `archive`* (COMBINE).

2.3.4 Literature Instantiation

Following the unification presented in Section 2.1, Tables 2.2 and 2.3 detail how the main literature algorithms are instantiated in Procedure 1 and Procedure 2, respectively, of our structure. In Table 2.2, k designates a constant of the algorithm set beforehand, and in Table 2.3, the "$*$" symbol means that the memory size is variable.
Table 2.2 shows that many of the MO-SLS algorithms in the literature use the current solution as a reference. However, recent studies increasingly encourage the use of the archive as a reference since it leads to improved results (Blot et al. 2017a,c). The recombination

Procedure 3: ITER(archive)

Input: archive, a Pareto set of solutions
Output: the updated archive* set

```
archive ← LS(archive);
archive* ← archive;
repeat
    memory, archive ← PERTURB(archive, archive*);
    archive ← LS(memory, archive);
    archive* ← COMBINE(archive, archive*);
until global stopping condition is met;
return archive*;
```

column highlights that the recombination only makes sense when the exploration step returns a new Pareto archive; for trajectory-based local search algorithms, such a step is directly performed during the exploration, when a neighbor replaces the current solution in the memory. Not mentioned here is the possible bounding of the archive size, which is also performed on some problems after Pareto filtering (e.g., (Liefooghe et al. 2012)). The selection column mainly differentiates between trajectory-based algorithms, for which such a step is likewise irrelevant, and algorithms that do not use a memory mechanism but recreate the set of new solutions every iteration.
Lastly, Table 2.3 shows that, if the first MO-SLS algorithms predominantly accepted improving neighbors, newer MO-SLS algorithms have shown that considering incomparable neighbors leads to improved results.

2.4 Automatic Design using the Unified Structure

The main objective of this literature review on MO-SLS during the PhD of Aymeric Blot was to get a unified structure of MO-SLS in order to tune or control many components and values. This MO-SLS unified structure was used as the core ingredients of tuning and control experiments carried out during the thesis. Two types of MO-SLS was therefore defined: a static MO-SLS to be tuned and an adaptive MO-SLS integrating control mechanisms.

2.4.1 Static MO-SLS Structure

The static MO-SLS algorithm is based on an iterated local search (see Procedure 3) and the DMLS algorithm of Liefooghe et al. (2012). We proposed a static MO-SLS used in several works on automatic configuration published in international conferences (Blot et al. 2018a, 2017a,c; Pageau et al. 2019) and a journal (Blot et al. 2019). The main contributions are presented in Chapter 3.
Algorithm 4 gives the pseudo-code of our static MO-SLS. The inner MO-SLS is close to the DMLS algorithm and iterates three successive steps: the selection step (procedure SELECT) in which solutions of the *current set* are selected and recorded in the *memory*,

Table 2.2: Condensed Literature Instantiation (LS Procedure)

imp.: improving (implies underlying scalarisation)
dom.: dominating
ndom.: non-dominated (either dominating or incomparable)

Algorithm	REFERENCE	UPDATE	COMBINE	SELECT	Miscellaneous
MOSA	current solution	replace if *imp.*	irrelevant	do nothing	
PSA	current solution	replace if *imp.*	irrelevant	do nothing	
MOTS	irrelevant	always replace	irrelevant	drift if long enough	
MOVNS	irrelevant	remove current if fully explored; add *ndom.* neighbors; filter	Pareto dominance	all solutions	
MOGLS	current solution	replace if *imp.*	Pareto dominance	do nothing	stop after a given number of explorations without improvement
$(1+1)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	
$(1+\lambda)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	
$(\mu+\lambda)$-PAES	current solution	replace if *dom.* or less crowded	Pareto dominance	do nothing	reference chosen via binary tournament
PLS-2	archive	do nothing	Pareto dominance	all solutions	
PLS-1	irrelevant	replace if *dom.*	Pareto dominance	single unexplored	
moRBC	current solution	replace if *imp.*	irrelevant	irrelevant	restart on local optima
IBMO-SLS	memory	do nothing	irrelevant	all solutions	
DMLS $(1 \cdot\ )$	current solution	do nothing	Pareto dominance	single unexplored	
DMLS $(\star \cdot\ )$	current solution	do nothing	Pareto dominance	all solutions	
SPLS	current solution	do nothing	Pareto dominance	single non-flagged	
FLS	current solution	always replace; filter	Pareto dominance	do nothing	

MOSA (Fortemps et al. 1994; Serafini 1994); PSA (Czyzak and Jaszkiewicz 1996); MOTS (M. P. Hansen 1997); MOVNS (Geiger 2008); MOGLS (Ishibuchi and Murata 1996); PAES (Knowles and Corne 1999, 2000a); PLS-2 (Talbi et al. 2001); PLS-1 (Angel et al. 2004; Paquete et al. 2004); moRBC (Aguirre and Tanaka 2005); IBMO-SLS (Basseur and Burke 2007); DMLS (Liefooghe et al. 2012); SPLS (Drugan and Thierens 2012); FLS (Moalic et al. 2013)

Table 2.3: Condensed Literature Instantiation (EXPLORE Procedure)

imp.: improving (implies underlying scalarisation)
dom.: dominating
ndom.: non-dominated (either dominating or incomparable)

Memory size: 1: single solution; k: constant; $*$: variable; other: other constant

Algorithm	Memory size	ACCEPT	UPDATE	Miscellaneous
MOSA	1	if *imp.*	update both	
PSA	$*$	if *imp.*	update both	
MOTS	$*$	best non-tabu neighbor	update both	
MOVNS	1	if *ndom.*	irrelevant	use an unexplored neighborhood
MOGLS	$*$	first *imp.*	do nothing	stop after first *imp.* neighbor or k neighbors
$(1+1)$-PAES	1	if *dom.* or less crowded	do nothing	stop after 1 neighbor
$(1+\lambda)$-PAES	1	if *dom.* or less crowded	do nothing	stop after λ neighbors
$(\mu+\lambda)$-PAES	μ	if *dom.* or less crowded	do nothing	stop after λ neighbors
PLS-2	$*$	if *ndom.*	do nothing	
PLS-1	1	if *ndom.*	do nothing	
moRBC$(1+1)$	1	if *dom.*	replace `ref` if accepted	neighbors generated using the current reference
moRBC$(1+1)^*$	1	if *ndom.*	replace `ref` if accepted	neighbors generated using the current reference
moRBC$(1+1)^A$	1	if *dom.* or less crowded *ndom.*	replace `ref` if accepted	neighbors generated using the current reference
IBMO-SLS	$*$	if not worst w.r.t. the indicator	remove worst from `ref`	
DMLS $(\ \cdot 1)$	k	if *dom.* or *ndom.*	do nothing	stop after 1 neighbor
DMLS $(\ \cdot 1_{\not\prec})$	k	if *dom.* or *ndom.*	stop after 1 *ndom.*	
DMLS $(\ \cdot 1_{\succ})$	k	if *dom.* or *ndom.*	do nothing	stop after 1 *dom.*
DMLS $(\ \cdot \star)$	k	if *dom.* or *ndom.*	do nothing	
SPLS	1	if *dom.* or *ndom.*	do nothing	
FLS	$*$	if *ndom.*	do nothing	stop after first *ndom.* neighbor

MOSA (Fortemps et al. 1994; Serafini 1994); PSA (Czyzak and Jaszkiewicz 1996); MOTS (M. P. Hansen 1997); MOVNS (Geiger 2008); MOGLS (Ishibuchi and Murata 1996); PAES (Knowles and Corne 1999, 2000a); PLS-2 (Talbi et al. 2001); PLS-1 (Angel et al. 2004; Paquete et al. 2004); moRBC (Aguirre and Tanaka 2005); IBMO-SLS (Basseur and Burke 2007); DMLS (Liefooghe et al. 2012); SPLS (Drugan and Thierens 2012); FLS (Moalic et al. 2013)

Algorithm 4: Static Multi-Objective Iterated Stochastic Local Search

Input: `archive`, a Pareto set of solutions
Output: the updated `archive` set

```
current ← archive;
/* MO-Iterated-SLS */
until termination criterion is met do
    /* Perturbation (except in loop 1) */
    current ← PERTURB ( archive );
    /* Inner MO-SLS */
    until inner termination criterion is met do
        /* Selection */
        memory ← SELECT(current);
        /* Exploration */
        candidates ← ∅;
        for solution ∈ memory do
            ref ← REFERENCE ( solution, current );
            accepted ← EXPLORE( solution, ref );
            candidates ← candidates ∪ accepted;
        /* Archive */
        current ← BOUND( pareto ( current ∪ candidates ));
    archive ← pareto ( archive ∪ current );
return archive;
```

the exploration step (procedure EXPLORE) in which the neighborhood of every selected solution is explored according to the reference (procedure REFERENCE) being either the *solution* or the *current set* , and the archive step (procedure BOUND) in which the resulting neighbors are merged into the current set of solutions in order to keep only the Pareto solutions. The outer MO-SLS simply performs a perturbation on the current *archive set*, then the inner MO-SLS procedure from the perturbed set of solutions, and finally, merges the resulting *current set* with the *archive*, and iterates until the global termination criterion is met.
This instantiation differs from the unification by the set of strategies we choose to focus on. In particular, it only supports dominance-based strategies, while strategies based on aggregation are not taken into account. Other differences include a selection step *before* the exploration step, at the beginning of the main loop, rather than at its end, a complete exploration of every solution selected, and an explicit combination mechanisms that first remove dominated solutions before bounding the size of the archive. Finally, as in the DMLS algorithm, the *memory* is not updated during the exploration, as it is discarded at the end of every iteration.
Table 2.4 lists 10 strategies/parameters and their possible values considered in the experiments presented in Chapter 3. There are five categorical parameters that enable different strategies of selection, exploration, archiving and perturbation, and five integer parameters that enable a better tuning of each corresponding strategies.

Table 2.4: Configuration space of the static multi-objective iterated SLS.

Phase	Parameter	Parameter values
Selection	`select-strat`	{`all`, `rand`, `newest`, `oldest`}
Selection	`select-size`	$\mathbb{N}^+$
Exploration	`explor-strat`	{`all`, `all-imp`, `imp`, `imp-ndom`, `ndom`}
Exploration	`explor-ref`	{`sol`, `arch`}
Exploration	`explor-size`	$\mathbb{N}^+$
Archive	`bound-strat`	{`unbounded`, `rand`, `replace`}
Archive	`bound-size`	$\mathbb{N}^+$
Perturbation	`perturb-strat`	{`kick`, `kick-all`, `restart`}
Perturbation	`perturb-size`	$\mathbb{N}^+$
Perturbation	`perturb-strength`	$\mathbb{N}^+$

In the following, we briefly recall and complete the description of every parameters of the configuration space of the static multi-objective iterated SLS.

`select-strat:` the selection strategy: with the value `all`, every solution of the archive will be explored; otherwise, with either of the values `rand`, `newest`, or `oldest`, only some solutions, respectively uniformly chosen at random from the archive, or chosen within the latest or oldest solutions included in the archive will be explored.

`select-size:` the (strictly positive) number of solution selected from the archive.

`explor-strat:` the exploration strategy: with the values `all` or `all-imp`, every neighbor is evaluated and the non-dominated and dominating neighbors, respectively, are returned; with the values `imp`, `imp-ndom`, or `ndom`, the neighbors are iteratively evaluated until a sufficient number of dominating neighbors, dominating and non-dominated neighbors, respectively, are found and returned. Finally, with the value `imp-ndom`, the non-dominated neighbors does not contribute to the *number* of neighbors selected (`explor-size`) but are also returned.

`explor-ref:` the reference of the exploration, either the current explored solution (with the value `sol`), or the current archive (with the value `arch`).

`explor-size:` the number of neighbors selected in the `imp`, `imp-ndom`, and `ndom` exploration strategies.

`bound-strat:` the bounding strategy after Pareto dominance. With the value `unbounded`, it returns the current archive. With the value `rand`, solutions chosen uniformly at random are discarded from the archive as long as the size of the archive is too large; with the value `replace`, it uses the DMLS strategy of removing newly accepted solutions if they did not replaced at least one solution of the archive.

`bound-size:` the maximum number of solutions in the archive above which the bounding strategy will apply.

`perturb-strat:` the perturbation strategy: with the value `kick`, some solutions of the archive will be selected uniformly at random and then iteratively replaced by one of their neighbors; with the value `kick-all`, the `kick` strategy applies for every solution of the archive; with the value `restart`, some solution of the search space, selected uniformly at random, will be considered instead.

`perturb-size:` the number of solutions considered in the `kick` and `restart` strategies.

`perturb-strength:` the number of successive kicks in the `kick` and `kick-all` strategies.

In the next Chapter, we propose a multi-objective methodology to automatically configure algorithms, and in particular, we use our static MO-SLS algorithm as a case study.

2.4.2 Adaptive MO-SLS Structure

Adaptive algorithms integrates control and feedback mechanisms in order to modify strategy components or numerical parameters during the execution (B. Doerr and C. Doerr 2018; Karafotias et al. 2015). They can, in principle, achieve robust performance over a broad range of problem instances since the integrated mechanisms and procedures are parameter and problem-independent. However, these algorithms generally consider only one or two parameters/strategies to adapt.
In the work (Blot et al. 2018c), we proposed an adaptive MO-SLS based on the static MO-SLS presented before and add the necessary structure to control the exploration strategy. We decided to control the exploration strategy only since our studies on automatic configuration show it is one of the strategy component that most impacts the performance. Therefore, we fixed the selection procedure to `select_1_rand` (i.e. `select-strat = rand` and `select-size = 1`), the archiving procedure to `bounded_pareto` (i.e. `bound-strat=rand` and `bound-size=100`), and the perturbation procedure `kick_1_3` (i.e. `perturb-strat=kick`, `perturb-size=1` and `perturb-strength=3`). We simplified the exploration procedure and we fixed `explor-ref=arch` and `explor-size=1`. Algorithm 5 describes the resulting adaptive MOLS algorithm.

2.5 Conclusion and Perspectives

Summary In this Chapter, we proposed a structure that unifies the multi-objective stochastic local search algorithms of the literature. This structure can instantiate extensions of single-objective SLS, SLS techniques integrated in evolutionary algorithm as well as real Pareto SLS. A simplified version of our structure has been automatically configured and the experiments are presented in Chapter 3. This version has also been extended with a learning mechanism in order to make it adaptive.

A Framework to Build Metaheuristics ParadisEO (Cahon et al. 2004) is a C++ white-box object-oriented framework dedicated to the reusable design of metaheuristics. It was updated to enable the design of both single and multi-objective SLS algorithms (Humeau et al. 2013; Liefooghe et al. 2011). The major advantage of ParadisEO

Algorithm 5: Adaptive Multi-Objective Iterated Local Search

Input: `archive`, a Pareto set of solutions
Output: the updated `archive` set

```
current ← archive;
/* Initialize all rewards */
INIT_REWARDS ( );
until termination criterion is met do
    /* Select exploration strategy */
    exploration ← CONTROL ( );
    /* Perturbation */
    current ← kick_1_3 ( archive );
    /* Apply the MO-SLS algorithm */
    current ← MO-SLS(current);
    /* Merge resulting archive and update rewards */
    tmp ← pareto ( archive ∪ current );
    UPDATE_REWARDS (exploration, current, tmp);
    archive ← tmp;
return archive;
```

is the dissociation between the problem-dependent and problem-independent design. However, the core of ParadisEO was first designed for single-objective evolutionary algorithms, and it became too constraining for the design of configurable or adaptive metaheuristics. jMetal is a Java framework for multi-objective optimization with metaheuristics (Durillo and Nebro 2011; Nebro et al. 2015) used in many optimization problems. Recently, a new version has been published to facilitate the resolution of dynamic optimization problems (Barba-González et al. 2018). Again, this framework is not adapted for the design of configurable or adaptive metaheuristics. Therefore, during his PhD, Aymeric Blot designed the AMH framework (Adaptive MetaHeuristics) (Blot et al. 2017b). This framework is dedicated to the design of configurable and adaptive metaheuristics. It is based on the execution flow of an optimization algorithm that enables flexibility of implementation. All the different structures of MO-SLS used in his thesis have been implemented into AMH. However, the current implementation of AMH is difficult to reuse in its state, since the implementation remained at the prototype stage. A short-term objective is to propose a framework with the flexibility of AMH and the problem-dependent/independent dissociation of ParadisEO.

Chapter 3

Automatic Configuration of Multi-Objective Stochastic Local Search Algorithms

In this chapter, we extend our work, on the automatic algorithm configuration (AAC) of single-objective stochastic local search algorithms presented in Chapter 1, to the multi-objective optimization. Indeed, the proposed unification of multi-objective stochastic local search (MO-SLS) algorithms (see Chapter 2) can be configured using many strategies or numerical parameters. The contributions presented in this chapter are derived from the PhD of Aymeric Blot and the collaboration with Holger Hoos (University of Leiden, The Netherlands).

MO-SLS are multi-objective algorithms that simultaneously optimize at least two objective criteria. The performance of such algorithms is evaluated through one or more indicator-based measures that give different information on the resulting Pareto front. Therefore, we consider that configuring a multi-objective algorithm is a multi-objective problem. In our work, we demonstrated that better results can be obtained by using a native, multi-objective algorithm configuration procedure (Blot et al. 2018a, 2017a, 2019, 2017c). Moreover, we used this procedure to automatically configure a dynamic algorithm that switches between configurations during the run (Pageau et al. 2019).

This chapter is organized as follows:

- Section 3.1 defines the multi-objective automatic algorithm configuration and then, presents MO-ParamILS, a multi-objective extension of the single-objective algorithm configuration framework ParamILS.
- Section 3.2 gives the experimental environment shared by the experiments presented.
- Section 3.3 presents extensive evidence that multi-objective AAC is preferable over single-objective AAC for the performance optimization of MO-SLS algorithms when there are several performance indicators of interest.
- Section 3.4 presents a study on the impact of correlation between optimization objectives on both the efficacy of different AAC approaches and the optimized algorithms obtained from these automated approaches.
- Section 3.5 presents a dynamic algorithm framework in which strategies and parameters are modified during the run according to a static schedule defined with AAC.

3.1 Multi-Objective AAC

3.1.1 Definition

Automatic algorithm configuration (AAC) has traditionally been defined as a single-objective optimization problem, with either running time or solution quality as the optimization objective. Multi-objective automatic algorithm configuration (MO-AAC) naturally arises when several criteria have to be optimized simultaneously and independently. Formally, a multi-objective configuration problem consists in a direct extension of the single-objective configuration problem (see Introduction chapter) where the original performance indicator is a vector of performance indicators.
Given a configurable target algorithm A, a space Θ of configurations of A, a distribution of instances $\mathcal{D}$, and a statistical population parameter E; we denote by A_θ the algorithm obtained by associating the configuration θ to the target algorithm A, Equation 5 (given in the Introduction chapter) becomes Equation 3.1 when the original performance indicator $o : \Theta \times \mathcal{D} \to \mathbb{R}$ is replaced by $o : \Theta \times \mathcal{D} \to \mathbb{R}^n$ a vector of n performance indicators $O(A, i) = (o_1(A, i), o_2(A, i), \ldots, o_n(A, i))$.

$$\begin{cases} \text{optimise} & E[O(A_\theta, i),\ i \in \mathcal{D}] \\ \text{subject to} & \theta \in \Theta \end{cases} \tag{3.1}$$

Here, the supposition is also made that the limit exists and is finite. Therefore, every component of the performance vector can be optimized independently and the MO-AAC becomes a standard multi-objective optimization problem (Equation).

$$\begin{cases} \text{optimise} & (E[o_1(A_\theta, i),\ i \in \mathcal{D}], \ldots, E[o_n(A_\theta, i),\ i \in \mathcal{D}]) \\ \text{subject to} & \theta \in \Theta \end{cases} \tag{3.2}$$

3.1.2 MO-ParamILS

3.1.2.1 Description

In the Introduction chapter, we claimed while there are numerous state-of-the-art AAC procedures in the literature, including irace (López-Ibáñez et al. 2016), based on statistical racing, SMAC (Hutter et al. 2011), based on random forests, and ParamILS (Hutter et al. 2009), based on iterated stochastic local search. When Aymeric Blot started his PhD, only few MO-AAC procedures exist, including SPRINT-race (Zhang et al. 2015) based on statistical racing and procedures based on evolutionary algorithms (Branke and Elomari 2012; Dréo 2009). Therefore, we proposed MO-ParamILS (Blot et al. 2016), a multi-objective extension of ParamILS.
The core algorithm of MO-ParamILS is given by Algorithm 6. Like its predecessor ParamILS, it is based on an iterated stochastic local search (Hoos and Stützle 2004; Lourenço et al. 2010), in which the incumbents (i.e., the best solutions so far) are iteratively improved by mean of both local search and perturbation mechanisms. Three parameters are exposed: the number r of initial random configurations, a restart probability p_{restart}, and the number s of random search steps performed in each perturbation phase. The `update()` function performs target algorithms runs and ensures that the

different configurations can be compared, while the `archive()` function simply discards dominated configurations.

Algorithm 6: Multi-objective ParamILS

Exposed parameters: r, p_{restart} and s
Input: Initial archive of configurations
Output: The archive of incumbents, i.e., the overall best configurations found

```
/* Initialisation */
current_arch ← initial archive;
for i ← 1...r do
    tmp ← random configuration;
    update(tmp, current_arch);
    current_arch ← archive(current_arch, tmp);

/* Iterated local search */
until termination criterion is met do
    /* Perturbation (unless for the first iteration) */
    if first iteration then
        tmp ← current_arch;
    else
        with probability p_restart then // Restart
            /* incumbents are not forgotten */
            current_arch ← { random configuration };
            tmp ← current_arch;
        otherwise // Random walk
            config ← current_config;
            for i ← 1...s do
                config ← random neighbor of tmp;
            tmp ← { config };
    /* Local search */
    tmp ← local_search(tmp);
    foreach config ∈ tmp do
        update(config, current_arch);
        current_arch ← archive(current_arch, config);
return the archive of incumbent;
```

As well as for ParamILS, MO-ParamILS starts by considering r random configurations, in order to compare the initial (usually default) configuration to a few others to make sure of its relevance. Then, it applies a stochastic local search procedure, which is based on the *one-exchange neighborhood*, i.e., modifying a single parameter value at a time. A tabu mechanism is also used to ensure that the configurator is never stuck. Between iterations, there is a p_{restart} chance to restart the search from a new configuration, uniformly chosen at random from the search space. Otherwise, a perturbation of s random steps is performed.

The main difference between ParamILS and MO-ParamILS is that the former focuses on optimizing a single configuration with regard to a single performance indicator, while the later optimizes an *archive* of configurations, i.e., the set of the current best configurations with regard to the multiple performance indicators.

3.1.2.2 The MO-AAC Protocol of MO-ParamILS

We proposed a MO-AAC protocol in three phases used in MO-ParamILS:

Training: In the training phase, MO-ParamILS is independently run multiple times on a given training set of instances; each of these runs produces a Pareto set of configurations. Multiple runs of MO-ParamILS are used, because individual runs can suffer from stagnation and to make effective use of parallel computing resources. However, as different MO-ParamILS runs usually use different subsets of the given training set, the configurations obtained from them cannot be fairly compared to each other.

Validation: To fairly compare configurations obtained from different MO-ParamILS runs and to reduce the number of configurations ultimately evaluated in the test phase, every configuration from the training phase is evaluated on the same, fixed subset of training instances. Based on the performance measurements thus obtained, Pareto-dominated configurations are removed.

Test: The Pareto set of configurations obtained from the validation phase is evaluated again, on a set of test instances that does not contain any of the instances used for training or validation. Again, Pareto-dominated configurations are removed.

3.2 Experimental Environment

In the two following sections, we will present the results of two campaigns of experiments where our MO-AAC approach is compared to classical single-objective AAC (SO-AAC) approaches. These experiments have been conducted on permutation problems described in this section. The target algorithm is the static multi-objective stochastic local search presented in Chapter 2, Section 2.4.

3.2.1 Unary Indicators

Multi-objective algorithms provide a Pareto set of solutions. In order to assess the quality of such Pareto sets, different indicators have been proposed (Knowles and Corne 2002; Okabe et al. 2003; Zitzler and Thiele 1999). These usually characterize the final Pareto set produced by a multi-objective algorithm in terms of convergence, distribution or cardinality. Since no single quality indicator captures all of these properties, it is recommended to consider multiple indicators, preferably complementing each other, in order to assess the performance of multi-objective algorithms (Zitzler et al. 2003).
In our experiments, we use a combination of two indicators: the classical hypervolume (Zitzler and Thiele 1999) and a complementary spread measure. These have been chosen in light of their common usage, their complementarity, and the additional requirements for unary indicators that do not require reference sets arising in the context of the automatic

algorithm configuration process at the core of our study.

Hypervolume (*HV*) is by far the most broadly used performance indicator in the literature on multi-objective optimization (Riquelme et al. 2015). Assuming normalized objective values in $[0, 1]$, the unary hypervolume measures the volume between the Pareto set of solutions and the point $(1, 1)$. *HV* is primarily a convergence indicator, but also captures information about the diversity of the front of solutions.
We use, as a complementary indicator, a variant of spread to capture the distributional properties of the Pareto set. Given a Pareto set S, ordered regarding the first criterion, we define

$$\Delta' := \frac{\sum_{i=1}^{|S|-1} |d_i - \bar{d}|}{(|S| - 1) \cdot \bar{d}},$$

where $\bar{d}$ denotes the average over the Euclidean distances d_i for $i \in [1, |S| - 1]$ between adjacent solutions on the ordered set S. This last indicator is to be minimized; it takes small values for large Pareto sets with evenly distributed solutions, and values close to or greater than 1 for Pareto sets with few or unevenly distributed solutions.
This slightly differs from the widely used spread indicator (Deb et al. 2002) in that it does not use extreme positions (after normalization, the points $(1, 0)$ and $(0, 1)$) and only considers the distribution inside the Pareto set. Obviously, this indicator cannot be used alone to assess a Pareto set, but it complements the information captured by the hypervolume indicator. Using these two unary indicators, we can assess the performance of multi-objective algorithms in terms of the quality, diversity and distribution of the final Pareto sets obtained. In our experiments, in order to facilitate analysis, we will consider the minimizing variant of hypervolume, calculated as $1-HV$, so that performance of a multi-objective algorithm can be optimized by minimizing both indicators. Then, good hypervolume values mean high *HV* (i.e., low values of $1-HV$).

3.2.2 MO-AAC vs. SO-AAC Approaches

In our experiments, we compare two SO-AAC approaches and one MO-AAC approach optimizing the performance of a multi-objective algorithm. Specifically, we consider three distinct AAC approaches (Blot et al. 2017c):

***HV*,** a SO-AAC approach that optimizes the hypervolume indicator only.

***HV*+Δ′,** a SO-AAC approach that optimizes a weighted sum of hypervolume (with a 0.75 coefficient) and Δ' spread (with a 0.25 coefficient).

***HV*||Δ′,** a MO-AAC approach that simultaneously considers hypervolume and Δ' spread.

The former approach is usually used in the literature (e.g., Bezerra et al. (2016)) while the latter two approaches are motivated by the belief that the performance assessment of multi-objective algorithms benefits from the use of multiple performance indicators. By comparing *HV* to the two other configuration approaches, we aim to assess this belief in the context of automatic configuration of MO-SLS algorithms. Furthermore, by comparing *HV*+Δ′ and *HV*||Δ′, we intend to assess the benefits of MO-ACC compared to SO-ACC

with aggregated performance metrics like it is usually done in the literature. Since the Δ' indicator is a complementary measure to the hypervolume, the aggregation coefficients (0.75 and 0.25) were thus set to mainly focus on convergence but keeping diversity in the solutions.

3.2.3 Configurators

We use the configurators ParamILS (Hutter et al. 2009) and MO-ParamILS (Blot et al. 2016), both shared a similar conception and are based on stochastic local search algorithms. Specifically, we use the FocusILS variants of both configurators, since these usually give the best performance. Since the target algorithm of our works is a multi-objective stochastic local search algorithm, we use the protocol presented before in Section 3.1.2.2. Using this protocol, we compare the three AAC approaches. In the case of the *HV* approach, ParamILS is used during the training phase to only optimize the hypervolume of the target algorithm, since the configuration scenario is single-objective. However, during the subsequent validation and test phases, the configurations resulting from the single-objective training are assessed using both hypervolume and Δ' spread separately, in a Pareto way. Ultimately, it is expected that this approach only finds good hypervolume values and disregards the Δ' spread value. Similarly, the *HV+*Δ' approach uses ParamILS during the training phase to optimize an aggregation of hypervolume and Δ' spread, while the assessment of the validation and test phases are then performed in a Pareto way. These two SO-AAC approaches constitute the baseline against which our MO-AAC approach is compared to. While they focus on specific directions of the multi-objective space and use an optimisation criterion less complete than the final evaluation, they represent the performance that our multi-objective approach will have to at least match.

3.2.4 Permutation Problems

3.2.4.1 The Bi-objective Permutation Flowshop Problem

In the permutation flow-shop scheduling problem (PFSP), N jobs $\{J_1, \ldots, J_N\}$ have to be scheduled on M machines $\{M_1, \ldots, M_M\}$. Each job J_i is processed sequentially on each of the machines, with fixed processing times $\{p_{i,1}, \ldots, p_{i,M}\}$, and machines are critical resources that can only process one job at a time. The sequencing of jobs is identical on every machine, so that a solution may be represented by a permutation π of size N.

In Experiments 1, we consider a bi-objective PFSP (PFSP) minimizing two classical objectives being the makespan $C_{\max}$ (Equation 3.3), i.e. the total completion time and, the sum of flowtimes SFT (Equation 3.4), where C_i represents the completion time of job i on machine M_M. Note that $C_{\max}$ and SFT are highly dependent to each other as the makespan is included into the sum of flowtimes.

$$f_1(\pi) = C_{\max} = \max_{i \in \{1,\ldots,N\}} \{C_i\} = C_{\pi_N} \tag{3.3}$$

$$f_3(\pi) = SFT = \sum_{i=1}^{N} C_i \tag{3.4}$$

The most widely studied PFSP instances are those introduced by Taillard (1993), with numbers of jobs, $N \in \{20, 50, 100, 200, 500\}$ and numbers of machines, $M \in \{5, 10, 20\}$. There are 10 instances for every valid (N, M) combination in Taillard benchmark set. We considered three types of PFSP instance sets, characterized by their number of jobs, $N \in \{50, 100, 200\}$ and a fixed number of machines set to $M = 20$. The higher the number of jobs, the more challenging the instances tend to be. For the exhaustive analysis and the test phase of the three AAC approaches, we use the 10 Taillard instances for each of those N values. For the training phase of the AAC approaches, we used a different, completely disjoint, set of instances, composed by newly generated *Taillard-like* instances. We generated 30 of these instances for each $N \in \{50, 100, 200\}$, using the original generation procedure.

In Experiments 2, we need to control the correlation between the objectives. Therefore, we considered a PFSP minimizing two makespan objectives computed from separate matrices $(P^k)_{k=\{1,2\}}$ of processing times, with controlled correlation between P^1 and P^2, such that $p^k_{i,j}$ is the processing time of job i on machine j. The two objectives f_1 and f_2 are computed from the respective matrix. We generated our own instances following the idea of uniform random generation underlying the commonly used Taillard instances (Taillard 1993). The processing times of matrix P^1 are generated following the uniform distribution $\mathcal{U}([1; 99])$. In the uncorrelated version of the problem, matrix P^2 is generated independently of matrix P^1, following the same distribution $\mathcal{U}([1; 99])$. For the two $\rho-$correlated versions, the coverage method is used to generate matrix P^2 from matrix P^1. For each $p^2_{i,j}$ value of matrix P^2, a real number α is drawn uniformly at random from $[0; 1]$. Then, $p^2_{i,j} = p^1_{i,j}$ if $\alpha < \rho$; otherwise, $p^2_{i,j}$ is sampled from $\mathcal{U}([1; 99])$. PFSP instances with medium and high correlation were obtained for $\rho = 0.6$ and $\rho = 0.9$, respectively.

Classical PFSP neighborhoods include the exchange neighborhood, where the positions of two jobs are exchanged, and the insertion neighborhood, where one job is reinserted at another position in the permutation. In both experiments, we consider a hybrid neighborhood defined as the union of the exchange and insertion neighborhoods, which is known to lead to better performance than considering each neighborhood independently (Dubois-Lacoste et al. 2015).

3.2.4.2 The Bi-objective Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is one of the most widely studied combinatorial optimization problems. It can be defined by a complete weighted graph G whose nodes represent cities, while edges corresponds to direct paths between cities. In the symmetric TSP, the graph is undirected, and edge weights correspond to distances between cities. Given a TSP instance G, the goal is to determine a tour passing through every city exactly once, such that the total distance traveled is minimized, i.e., a minimum-weight Hamiltonian cycle in G. This cycle corresponds to a permutation of the cities. In our experiments, we consider the bi-objective symmetric TSP (TSP), in which each instance is defined by a graph G, as in the standard TSP, but each edge between two nodes i and j has two weights, $d^1_{i,j}$ and $d^2_{i,j}$. The two objectives, f_1 and f_2, are defined as the total distance cov-

ered by a given tour according to each of the two distance matrices D^1 and D^2, respectively.

A benchmark set of Euclidean instances (available online[1]) has been widely used in the literature to assess the performance of TSP algorithms. These instances were constructed by combining two independently generated distance matrices computed using Euclidean distance between cities randomly placed on a two-dimension grid. The TSP benchmark set contains six instances each for 100, 300 and 500 cities, meaning 15 different pairwise independent combinations of two instances per benchmark size. In the test phase of Expriments 1, we used these 15 instances. For the training phase of the AAC approaches, we used a different, completely disjoint, set of newly generated instances containing 30 instances for each number of cities, obtained using the original generator from the DIMACS challenge.

Once again, we need to control the objectives correlation. First, the coordinates of the N cities are uniformly sampled from $[0; 3163]^2$, and the distance values in matrix D^1 are computed as Euclidean distances between these points. In the uncorrelated version, matrix D^2 is independently generated using the same protocol. For the two $\rho-$correlated versions, the original coordinates of each city are moved based on a normal distribution $\mathcal{N}(0, \rho)$, and then, the Euclidean distances between the new coordinates form the entries of D^2. The lower ρ, the higher the correlation between D^1 and D^2. TSP instances were obtained with medium and high correlation using $\rho = 600$ and $\rho = 150$, respectively.

In both experiments, we consider the 2-opt neighborhood, where two tours are neighbors if, and only if, one can be obtained from the other by removing two non-adjacent edges reconnecting the resulting tour fragments by two other edges.

3.2.4.3 The Bi-objective Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) involves assigning a set of N facilities to a set of N given locations, minimising a cost function that depends on the distance between locations and the flow required between the facilities assigned to these locations. A solution is then a permutation π where π_i is the facility of location i. This well-known single-objective problem has been extended to multiple objectives by Knowles and Corne (Knowles and Corne 2003). In our works, we considered a bi-objective QAP with distance matrix D, where $d_{i,j}$ is the distance between locations i and location j, and two flow-matrices $(W^k)_{k=\{1,2\}}$, such that, $w^k_{i,j}$ is the required flow between facility i and facility j according to flow-matrix W^k. The objectives are computed following Equation 3.5.

$$f_k(\pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} w^k_{i,j} d_{\pi_i,\pi_j} \tag{3.5}$$

This problem is tackled in Experiments 2 only. As for the PFSP and TSP, we follow the same protocol for generating problem instances. First, we compute D as a symmetric matrix of size $N \times N$ using the Manhattan distance between N locations uniformly generated within the interval $[0; 99]$.

[1] `https://eden.dei.uc.pt/~paquete/tsp/#Exp2`

The values of the flow-matrix W^1 are sampled from $\mathcal{U}([0;99])$. In the uncorrelated version, W^2 is also sampled from $\mathcal{U}([0;99])$, independently of W^1. For the two $\rho-$correlated versions, we use the coverage method to generate matrix W^2 from matrix W^1. For each $w^2_{i,j}$ value of matrix W^2, a real number β is sampled uniformly at random from $[0;1]$. Then, $w^2_{i,j} = w^1_{i,j}$ if $\beta < \rho$; otherwise, $w^2_{i,j}$ is sampled from $\mathcal{U}([0;99])$. As for the PFSP, we obtain QAP instances with medium and high correlation for $\rho = 0.6$ and $\rho = 0.9$, respectively.

In this work, we consider the commonly used exchange operator for the QAP, under which a neighbor of a solution is obtained by swapping two facilities between their two locations.

3.3 Experiments 1: Interest of using a MO-AAC Approach

3.3.1 Experimental Protocol

3.3.1.1 Configuration Spaces

Our MO-SLS algorithm iterates over four distinct phases: *selection*, *exploration*, *archive* and *perturbation*. All four phases can be instantiated in many different ways via parameters and their permissible values presented in Chapter 2, Section 2.4. In these experiments, we propose two configurations spaces. The first one leads to 300 configurations only, which enables an exhaustive analysis of the configuration space as well as the comparison with the three different AAC approaches. Table 3.1 gives an overview of the subset of parameters chosen. This *small* configuration space has been defined based on preliminary experiments, in which we informally identified parameters and parameter values most likely to result in good performance of our target algorithm. The small size of this search space permits us to perform validation and testing in an exhaustive way, where the performance of each of the 300 configurations is assessed on the entire training and test instance sets, respectively. This exhaustive analysis of the configuration space enables the comparison of the configurations resulting from the training phase to ones that may otherwise be never considered. Our goal with this analysis is to demonstrate that our AAC approaches can effectively configure a MO-SLS algorithm for solving bi-objective permutation problems. We extend the permissible values of some parameters and obtained 10 920 configurations presented in Table 3.2. Hence, the three AAC approaches will compete over a larger configuration space.

3.3.1.2 AAC Protocol

Table 3.3 summarizes the details of our AAC protocol for both configuration spaces. The main differences concern the training phase. For the small (large) configuration space, ParamILS starts by evaluating a single (10) random configuration, and can execute 100 (1000) MO-SLS runs before stopping, where each selected configuration cannot be run more than 10 (100) times. Due to the reduced size of the small configuration space, only 10 independent runs of ParamILS are performed, compared to 20 runs for the large space. In the validation phase, the configurations resulting from the training phase are evaluated on all training instances, running every configuration once on each instance. In the test

Table 3.1: Small configuration space (300 configurations)

Phase	Parameter	Parameter values
Selection	`select-strat`	{`all`, `rand`, `oldest`}
Selection	`select-size`	{1, 10}
Exploration	`explor-strat`	{`imp`, `imp_ndom`, `ndom`}
Exploration	`explor-ref`	{`sol`, `arch`}
Exploration	`explor-size`	{1, 10}
Archive	`bound-size`	{1000}
Perturbation	`perturb-strat`	{`kick`, `kick_all`, `restart`}
Perturbation	`perturb-size`	{10}
Perturbation	`perturb-strength`	{3, 10}

Table 3.2: Large configuration space (10 920 configurations)

Phase	Parameter	Parameter values
Selection	`select-strat`	{`all`, `rand`, `newest`, `oldest` }
Selection	`select-size`	{1, 3, 10}
Exploration	`explor-strat`	{`all`, `all_imp`, `imp`,`imp_ndom`, `ndom`}
Exploration	`explor-ref`	{`sol`, `arch`}
Exploration	`explor-size`	{1, 3, 10}
Archive	`bound-size`	{20, 50, 100, 1000}
Perturbation	`perturb-strat`	{`kick`, `kick_all`, `restart`
Perturbation	`perturb-size`	{1, 5, 10}
Perturbation	`perturb-strength`	{3, 5, 10}

phase, each of the configurations in the Pareto set obtained from the validation phase is run 10 times on every test instance. For both validation and test phases, the performance of each configuration is assessed based on the average hypervolume and spread values over the runs. Obviously, for the small configuration space, our exhaustive analysis ensures that the performance of all configurations are known for all training and test instances, and we will directly use these results in the validation and test phases to avoid recomputing the performance of configurations selected in the training phase.

3.3.2 Results and Analysis

3.3.2.1 Small Configuration Space

The results from this analysis for training and test instance sets are shown in Figures 3.1 and 3.2. Generally, the shapes of the Pareto sets in objective space are similar between validation and test results, indicating that our AAC approaches do not suffer from overfitting. Therefore, we will focus our discussion on the test results seen in Figure 3.2.

Table 3.3: AAC Protocol

Phase	Small configuration space	Large configuration space
Training	No default configuration	No default configuration
	1 random configuration	10 random configurations
	10 ParamILS runs	20 ParamILS runs
	100 MO-SLS runs budget	1000 MO-SLS runs budget
	max 10 MO-SLS run per config.	max 100 MO-SLS run per config.
Validation	1 run per instance	1 run per instance
Test	10 runs per instance	10 runs per instance

Table 3.4 details how many unique (in parentheses: non-unique) configurations were found by each AAC approach, and how many survived the validation and test phases. (we recall that Pareto-dominated configurations are pruned in those phases).

Table 3.4: Number of configurations after training, validation and testing

Scenario	Approach	Small space			Large space		
		Configs	Pareto	Final	Configs	Pareto	Final
PFSP 50	HV	10	2	2	20	2	2
	$HV{+}\Delta'$	10	4	2	10	2	2
	$HV{\|\|}\Delta'$	32 (38)	9	7	145	14	11
PFSP 100	HV	10	4	3	19 (20)	1	1
	$HV{+}\Delta'$	8 (10)	3	3	20	4	2
	$HV{\|\|}\Delta'$	36 (42)	12	6	171 (172)	27	19
PFSP 200	HV	10	3	3	20	4	3
	$HV{+}\Delta'$	9 (10)	5	3	16 (20)	2	2
	$HV{\|\|}\Delta'$	29 (39)	11	8	111 (117)	14	9
TSP 100	HV	6 (10)	1	1	15 (20)	2	1
	$HV{+}\Delta'$	6 (10)	2	2	15 (20)	6	4
	$HV{\|\|}\Delta'$	16 (26)	3	3	62 (73)	11	5
TSP 300	HV	9 (10)	2	2	13 (20)	4	2
	$HV{+}\Delta'$	9 (10)	5	5	12 (20)	5	2
	$HV{\|\|}\Delta'$	33 (41)	8	6	107 (130)	18	12
TSP 500	HV	6 (10)	5	4	16 (20)	4	4
	$HV{+}\Delta'$	8 (10)	5	4	14 (20)	3	2
	$HV{\|\|}\Delta'$	36 (40)	12	11	135 (145)	25	22

Figure 3.3 shows the parameter distribution of the 300 configurations on test instances according to our three selection mechanisms (crosses : $+ \times \star$, polygons: $\square\Delta\diamond$, and circles: $\circ\oplus\otimes$) and our three exploration strategies (orange: $+\square\circ$, green: $\times\Delta\oplus$, and violet: $\star\diamond\otimes$). Finally, Tables 3.5 and 3.6 list the Pareto-optimal configurations within the small con-

figuration space. A "$*$" symbol indicates that the value of the respective parameter does not impact the performance of the configured MO-SLS when the other parameter values are held fixed at the values shown. Conversely, when a specific parameter is shown, any deviation from it will reduce performance.

First, we will discuss the results for the PFSP. None among the 300 possible configurations simultaneously achieves good hypervolume and spread values (see Figure 3.2); the Pareto front is distinctly non-convex. While for the smallest scenario with 50 jobs, most of all configurations achieve good hypervolume values (i.e., low $1 - HV$), such configurations get rarer as the number of jobs increases. This result was expected, since it is known that larger PFSP instances are harder for multi-objective stochastic local search. Examining these results in more detail, we observe that the `imp` exploration strategy always obtains rather bad hypervolume values (see Figure 3.3). For 50 jobs, this strategy leads to better spread values; however, this tends to be no longer true for larger instances. For the three instance sizes, the `imp-ndom` and `ndom` strategies appear to give better performance in terms of hypervolume.

All three approaches find very good, even near-optimal configurations – in particular, $HV||\Delta'$, which achieves spreads over the entire Pareto-front. The 10 configurator runs of HV and $HV+\Delta'$ produce close to 10 unique configurations each (see Table 3.4), and all of these show good hypervolume values. However, after validation and testing, for both AAC approaches, few configurations remain, and those tend to have good hypervolume but average spread. On the other hand, our MO-AAC approach, $HV||\Delta'$, produces many more configurations after the training, validation and test phases. Compared to the other approaches, $HV||\Delta'$ clearly achieves better coverage of the optimal Pareto set of configurations (Figure 3.2). Note that all three approaches use the same time budget for configuration, the number of final solutions being strongly dependent of the kind (single-objective or multi-objective) of AAC used for training.

Regarding the nature of the configurations, we observe a trend across the three instance sizes (Tables 3.5): The best hypervolume is always reached with the `oldest` selection strategy, the `ndom` exploration strategy and the `arch` exploration reference set choice. Slightly worse hypervolume, but better spread is achieved using the `imp_ndom` exploration strategy. Finally, the best spread values are obtained from configurations using the `imp` exploration strategy, although this comes at the cost of rather bad hypervolume. In almost every case, the perturbation strategy did not significantly impact the performance of the non-dominated configurations.

Our results on the TSP differ markedly from those on the PFSP. Firstly, we observe that the shape of the Pareto-optimal front of configurations varies with instance size: While it is convex for 100 cities with some degree of correlation between hypervolume and spread, for larger instances, the correlation between the two performance indicators decreases, and the front becomes non-convex. In contrast to the PFSP, where the two objectives were correlated, for our TSP benchmark sets, the objectives are completely independent; therefore, the final archives are much bigger, as there exist a richer space of trade-off solutions.

The impact on spread is evident from Figure 3.2; values above 1 correspond to two tightly clustered sets of solutions separated by a large gap that the respective configuration of MO-SLS failed to cover, and spread values of 0 correspond to final sets containing only

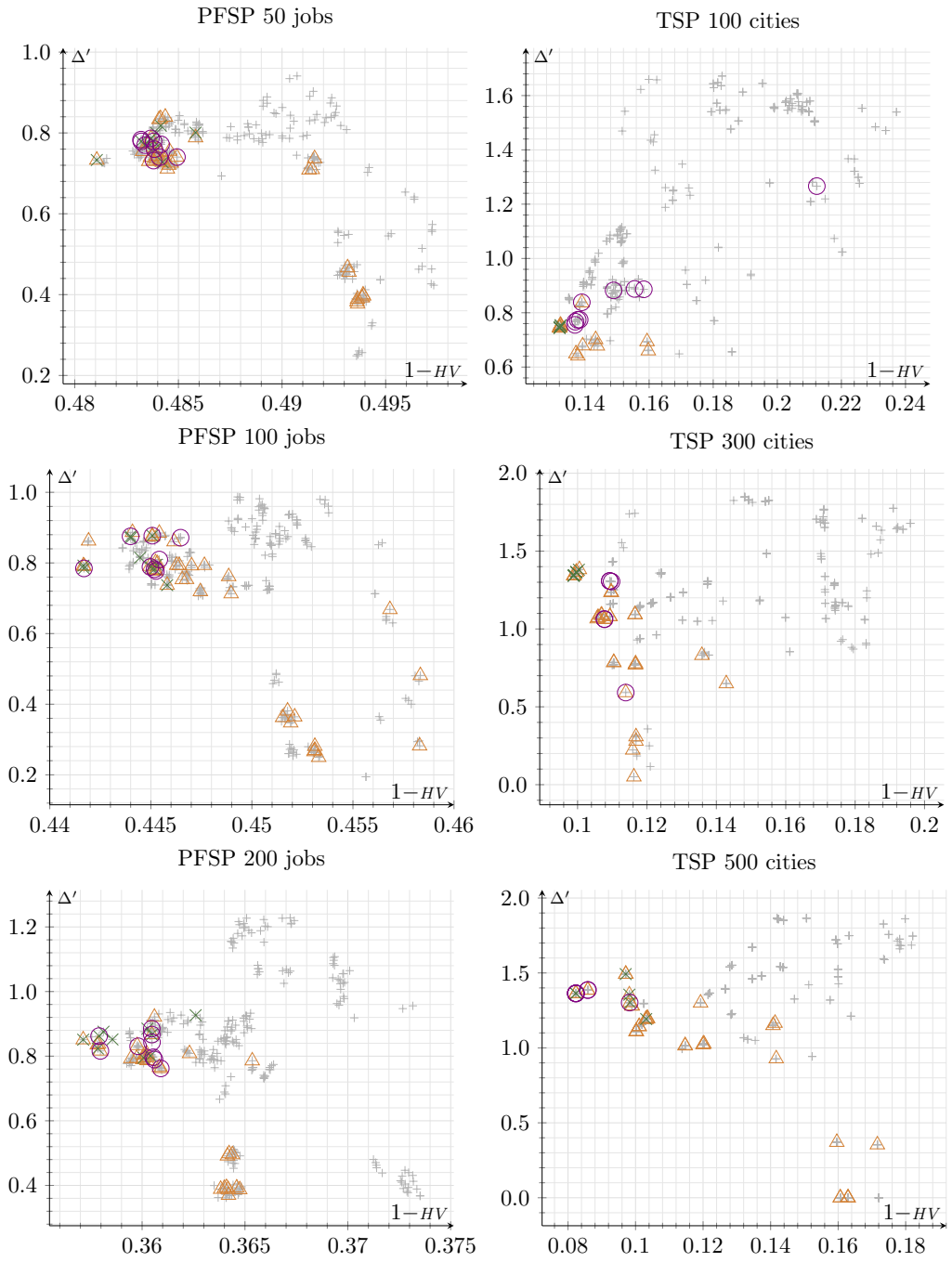


Figure 3.1: Exhaustive analysis on training instances (left: PFSP; right: TSP)
x: HV approach, o: HV+Δ' approach, △: $HV||\Delta'$ approach, +: exhaustive analysis

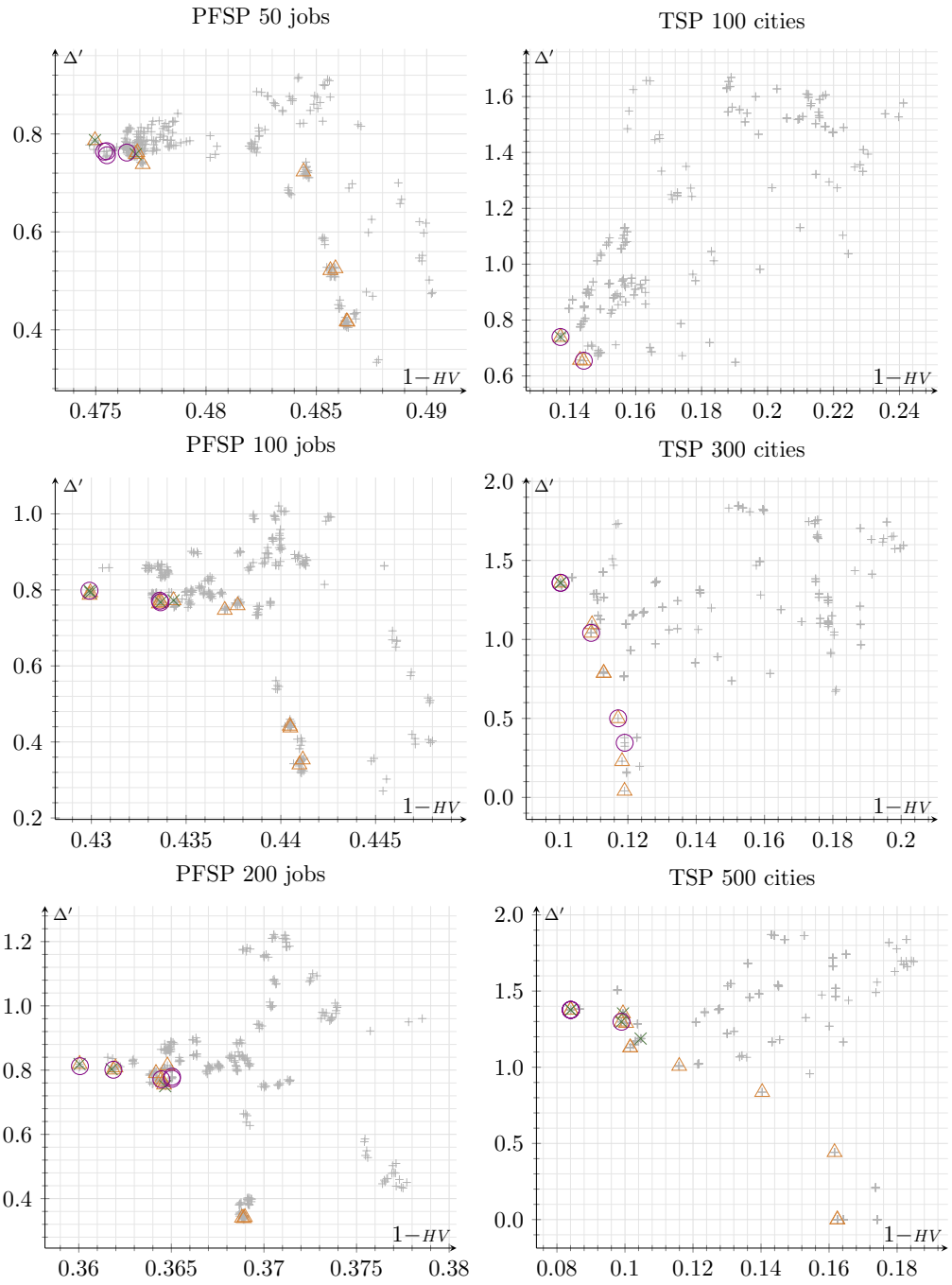


Figure 3.2: Exhaustive analysis on test instances (left: PFSP; right: TSP)
x: HV approach, o: HV+Δ' approach, △: $HV||\Delta'$ approach, +: exhaustive analysis

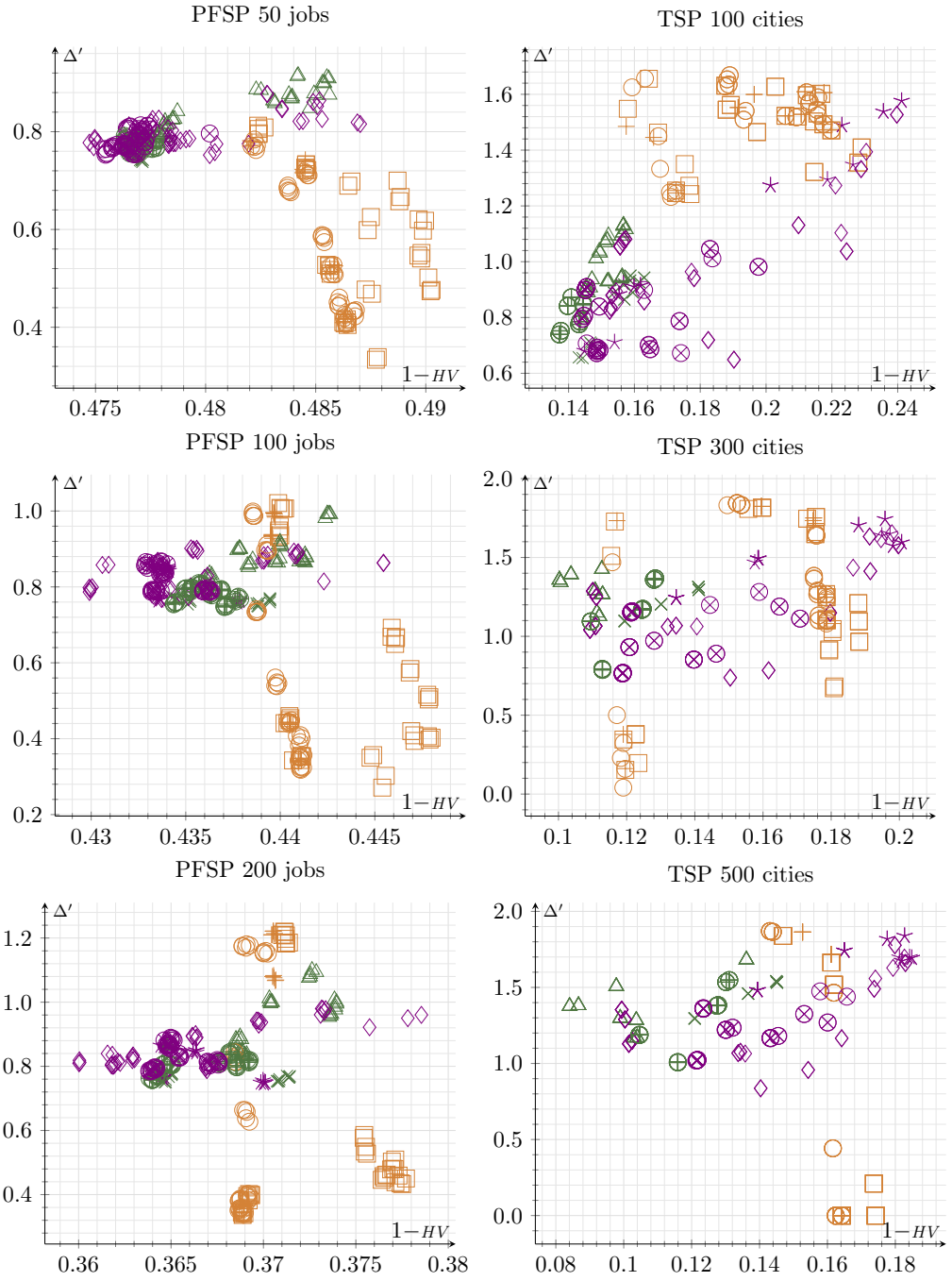


Figure 3.3: Exhaustive analysis parameter distribution on test instances (left: PFSP; right: TSP); Selection strategy: +×⋆: `all` (crosses), □△◇: `oldest` (polygons), ○⊕⊗: `rand` (circles); Exploration strategy: +□○: `imp` (orange), ×△⊕: `imp_ndom` (green), ⋆◇⊗: `ndom` (violet)

Table 3.5: Pareto-optimal configurations (small space)

50-job PFSP

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.4747	0.7775	oldest	10	ndom	arch	1	1000	$*$	10	$*$
0.4754	0.7640	all		ndom	arch	1	1000	$*$	10	$*$
0.4770	0.7420	all		imp_ndom	sol	10	1000	$*$	10	$*$
0.4837	0.6798	rand	1	imp	arch	10	1000	$*$	10	$*$
0.4853	0.5856	rand	1	imp	sol	10	1000	$*$	10	$*$
0.4855	0.5277	$*$	10	imp	arch	1	1000	$*$	10	$*$
0.4860	0.4433	rand	1	imp	arch	1	1000	$*$	10	$*$
0.4862	0.4093	$*$		imp	sol	1	1000	$*$	10	$*$
0.4877	0.3336	oldest	1	imp	sol	1	1000	kick	$*$	10

100-job PFSP

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.4299	0.7865	oldest	10	ndom	arch	1	1000	kick	10	3
0.4299	0.7979	oldest	10	ndom	arch	1	1000	kick_all		$*$
0.4332	0.7802	oldest	1	ndom	arch	1	1000	kick	10	$*$
0.4336	0.7640	all		ndom	arch	1	1000	$*$	10	$*$
0.4344	0.7541	rand	10	imp_ndom	arch	1	1000	$*$	10	$*$
0.4351	0.7540	all		imp_ndom	sol	1	1000	$*$	10	$*$
0.4370	0.7470	rand	10	imp_ndom	arch	10	1000	$*$	10	$*$
0.4387	0.7338	rand	1	imp	arch	10	1000	$*$	10	$*$
0.4397	0.5396	rand	1	imp	sol	10	1000	$*$	10	$*$
0.4402	0.4409	$*$	10	imp	arch	1	1000	$*$	10	$*$
0.4407	0.3428	oldest	10	imp	sol	1	1000	$*$	10	$*$
0.4410	0.3201	rand	1	imp	sol	1	1000	$*$	10	$*$
0.4410	0.3371	all		imp	sol	1	1000	$*$	10	$*$
0.4454	0.2711	oldest	1	imp	sol	1	1000	kick	10	$*$

200-job PFSP

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.3600	0.8093	oldest	1	ndom	arch	1	1000	restart/kick	10	$*$
0.3618	0.8027	oldest	10	ndom	arch	1	1000	$*$	10	$*$
0.3638	0.7628	rand	1	imp_ndom	arch	1	1000	$*$	10	$*$
0.3645	0.7534	all		imp_ndom	arch	1	1000	$*$	10	$*$
0.3686	0.3511	rand	1	imp	sol	1	1000	$*$	10	$*$
0.3687	0.3456	$*$	10	imp	sol	1	1000	$*$	10	$*$

Table 3.6: Pareto-optimal configurations (small space)

100-city TSP instances

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.1372	0.7389	rand	10	imp_ndom	sol	10	1000	$*$	10	$*$
0.1431	0.6572	all		imp_ndom	sol	10	1000	restart		
0.1443	0.6544	all		imp_ndom	arch	10	1000	restart		
0.1902	0.6488	oldest	1	ndom	sol	1	1000	kick	10	3

300-city TSP instances

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.1003	1.3582	oldest	10	imp_ndom	sol	1	1000	$*$	10	$*$
0.1006	1.3417	oldest	10	imp_ndom	sol	10	1000	$*$	10	$*$
0.1092	1.0409	oldest	10	ndom	sol	10	1000	$*$	10	$*$
0.1128	0.7933	rand	10	imp_ndom	arch	1	1000	$*$	10	$*$
0.1129	0.7880	rand	1	imp_ndom	arch	1	1000	$*$	10	$*$
0.1171	0.5003	rand	1	imp	sol	10	1000	restart		
0.1183	0.2288	rand	1	imp	sol	1	1000	restart		
0.1190	0.0409	rand	1	imp	arch	1	1000	restart		

500-city TSP instances

1-HV	Δ'	Selection		Exploration			Archive	Perturbation		
0.0841	1.3767	oldest	10	imp_ndom	sol	1	1000	$*$	10	$*$
0.0989	1.2983	oldest	1	imp_ndom	arch	10	1000	$*$	10	$*$
0.1003	1.2897	oldest	10	ndom	arch	10	1000	$*$	10	$*$
0.1015	1.1290	oldest	10	ndom	sol	10	1000	$*$	10	$*$
0.1159	1.0080	rand	$*$	imp_ndom	arch	1	1000	$*$	10	$*$
0.1403	0.8468	oldest	10	ndom	arch	1	1000	kick	10	$*$
0.1616	0.4420	rand	1	imp	sol	10	1000	$*$	10	$*$
0.1624	0.0000	rand	1	imp	$*$	1	1000	$*$	10	$*$

two solutions, which are produced when the `imp` exploration strategy fails to sufficiently diversify.
Our HV configuration approach produced few configurations, achieving near-optimal hypervolume. $HV{+}\Delta'$ produced weak training results on the 100-city instances, but worked well on the 300-city instances, because of the shape of the Pareto-optimal front. As for the PFSP, $HV||\Delta'$ found many more configurations and achieved far better coverage of the Pareto front. In our test instances from the literature, all three AAC approaches produced optimal configurations for 100-city instances, $HV{+}\Delta'$ and $HV||\Delta'$ still did on 300-city instances, and only $HV||\Delta'$ managed to find most of the optimal configurations on the 500-city instance (see Figure 3.2).
Analysing the MO-SLS configurations in more detail, those that achieve the best hypervolume values always use the `imp_ndom` exploration strategy with the `sol` reference set. While for 300- and 500-city instances, the `oldest` selection strategy is preferred, for 100 cities, the more common `rand` selection strategy performs better. Similarly to the PFSP, the choice of perturbation mechanism does not significantly impact the performance of optimal configurations.

For both problems, within the small configuration space, all three AAC approaches are able to find configurations very close to the true Pareto-front. The two SO-AAC approaches strongly favors the hypervolume indicator, while the MO-AAC approach is able to accurately cover the full range of Pareto-optimal configurations.

3.3.2.2 Large Configuration Space

Figure 3.4 shows the final configurations produced by all three AAC approaches for our six benchmarks (two problems, three instance sizes). We also show the configurations of the smaller set of configurations that we exhaustively evaluated, in order to show that these final configurations map very closely those found within the small space, which suggests that the small space indeed captures the high-performance configurations from the much larger space and, more importantly, demonstrates that our AAC approaches effectively finds such configurations. In the following, we will focus on the performance of our three AAC approaches.
Both SO-AAC approaches, HV and $HV{+}\Delta'$, produced few non-dominated configurations in their final testing phase – typically between 2 and 4 on each instance set (see Table 3.4). As one might expect, HV always finds a final configuration with near-optimal hypervolume. The results for $HV{+}\Delta'$ are similar to those for HV for the PFSP, but markedly different on our TSP benchmarks. For 100-city TSP instances, $HV{+}\Delta'$ covers the Pareto front, while for 300 cities, it finds the two extreme configurations, due to accidentally well-adapted weights used for aggregating hypervolume and spread. However, due to the non-convex shape of the front, no trade-off configurations are found between these extremes. For 500-city instances, $HV{+}\Delta'$ only finds configurations with near-optimal hypervolume, similar to what we observed for the PFSP.
On the other hand, our MO-AAC approach, $HV||\Delta'$, consistently provides many more non-dominated configurations, except for the small 100-city TSP instance set, where the Pareto front is completely covered by all three approaches. In all cases, the sets of configurations found by $HV||\Delta'$ are very well distributed over the entire front of optimal configurations.

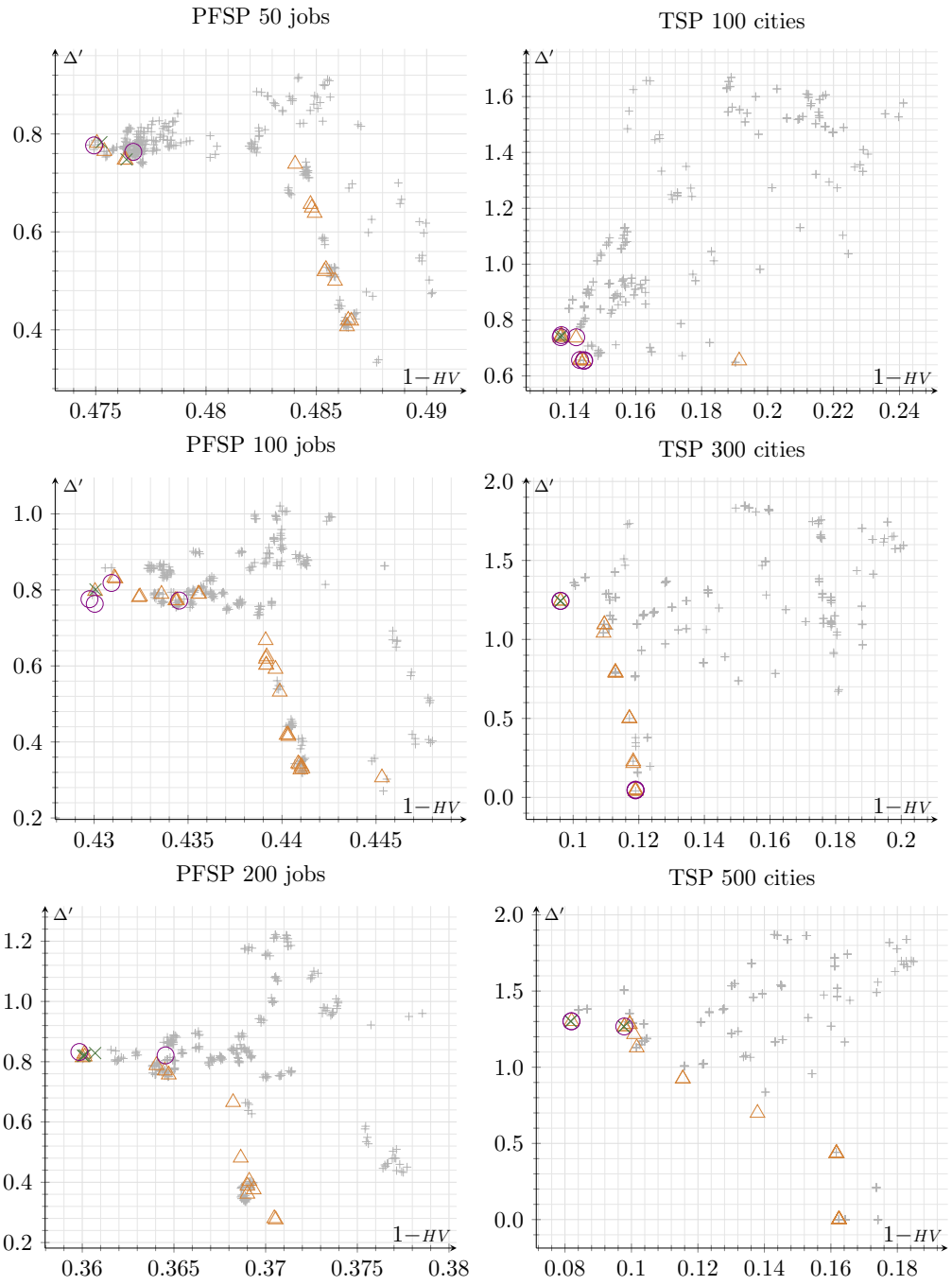


Figure 3.4: Large-scale analysis on test instances (left: PFSP; right: TSP)
x: HV approach, o: HV+Δ' approach, △: $HV||\Delta'$ approach, +: exhaustive analysis

Although $HV{+}\Delta'$ sometimes finds better configurations (e.g., on the 100- and 200-jobs PFSP scenarios), $HV{|}|\Delta'$ always produces configurations with similar performance.

Overall, our MO-AAC approach, $HV{|}|\Delta'$, produces substantially better results than the two SO-AAC approaches, HV and $HV{+}\Delta'$. HV finds excellent sets of configurations with respect to hypervolume, but only provides very few of those and consequently fails to achieve good spread. $HV{+}\Delta'$ sometimes provides better results and, under favorable circumstances, can cover the entire set of Pareto-optimal configurations; however, especially for more challenging scenarios, its performance is similar to that of HV. The main drawback of this approach is the requirement of a costly preliminary step for calibrating the weights used for aggregating the two optimization objectives. Finally, $HV{|}|\Delta'$, our *native* MO-AAC approach, always efficiently covers the entire Pareto-front of configurations, while still finding sets of configurations with excellent hypervolume, as produced by the two SO-AAC approaches.

3.4 Experiments 2: Impacts of Objectives Correlation on AAC

3.4.1 Experimental Protocol

3.4.1.1 Permutation Problems Benchmarks

For each of the three bi-objective permutation problems, we considered six benchmark sets, using two instances sizes and three degrees of correlation between the given objectives (*high*, *medium* and *no* correlation). For the PFSP, we consider a set of medium-size instances with 50 jobs and 20 machines, as well as a set of instances with 100 jobs and 20 machines. For the TSP, we consider a set of 50-city instances and a set with 100-city instances. Similarly, for the QAP, we consider two sets of 50- and 100-facility instances. For each of the 18 resulting benchmark sets, we generated 30 bi-objective instances for use as training and validation sets during automated configuration, and a second set of 10 additional instances for subsequent performance testing.

3.4.1.2 Configuration Space

We use parameters and their permissible values that lead to 10 920 configurations already presented in the previous section (see Table 3.2).

3.4.1.3 AAC Protocol

Experiments are conducted for a given scenario following three phases using ParamILS or MO-ParamILS configurators. First, during the *training* phase, the configurator is run 20 times with a budget of 1000 target algorithm runs, using a different and random ordering of the training instances. The search starts from 10 configurations uniformly sampled on the configuration space. A threshold was set to limit, for a given configuration, the number of runs to 100, in order to force diversification. Indeed, preliminary experiments show that the threshold may be reached overall 5 times for both HV and $HV{+}\Delta'$ approaches. Then,

both approaches using ParamILS followed the recommendation of using adaptive and aggressive capping (Hutter et al. 2009). During the *validation* phase, each configuration returned by the configurator is run 1 time on each of the 30 training instances, totaling 30 runs per configuration. Pareto-dominated configurations are then filtered. Finally, during the *test* phase, each non-dominated configuration is run 10 times on each of the 10 testing instances, totaling 100 runs, and Pareto-dominated configurations are filtered to obtain the final set of optimal configurations, presented and discussed in the next Section.

3.4.2 Results and Analysis

3.4.2.1 Optimised Configuration

PFSP Tables 3.7 and 3.8 show the final non-dominated configurations found by the three AAC approaches HV, $HV{+}\Delta'$ and $HV||\Delta'$ on our six PFSP scenarios. The configurations found across all six scenarios are very similar. The combination of `ndom` exploration with either the `oldest` or the `rand` selection strategy seems to lead to the best performance in terms of hypervolume, while the combination of the `imp` exploration strategy with the `rand` selection strategy leads to solution sets with worse hypervolume but better spread. While both the `sol` and `arch` exploration reference choices are found within the final configurations for all scenarios, `arch` is slightly more favored on larger instances, indicating that referencing more stringently against the current archive during exploration is beneficial for larger instance sizes.
The degree of correlation between objectives does not seem to impact the set of optimized configurations.

TSP Tables 3.9 and 3.10 show the final non-dominated configurations found by our three AAC approaches, HV, $HV{+}\Delta'$ and $HV||\Delta'$, on the six TSP scenarios. Compared to the PFSP scenarios, the number of distinct non-dominated configurations is much smaller. The configurations we found vary strongly with both correlation level and problem size. Overall, the `ndom` exploration strategy is preferred, together with either the `rand` or the `oldest` strategy. However, for instance with medium or no correlation, the `arch` exploration reference leads to better HV performance, and may be used together with the `imp_ndom` exploration strategy when the number of cities increases. Furthermore, on smallest high correlated instances the MO-SLS algorithm may benefit from using a bounded archive and restart between iterations, while a large archive of size 1000 is chosen (i.e., basically unbounded), along with a kick-based perturbation strategy, for all other instances. This is consistent with the idea that larger TSP instances benefit from a less aggressive perturbation mechanism in combination with a more diverse archive of candidate tours.
On the TSP, we note that correlation between objective has an impact similar to the problem size, making low (and no) correlated small instances significantly harder and requiring more aggressive mechanisms than equally sized high correlated instances.

QAP Tables 3.11 and 3.12 show the final non-dominated configurations found by our three AAC approaches for the six QAP scenarios. Again, much fewer configurations were obtained than for the PFSP scenarios. These configurations are much more varied than

Table 3.7: PFSP 50 jobs 20 machines *(optimised configurations)*

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.494	1.385	oldest 1	ndom arch 1	1000	kick 1 10
0.494	1.355	oldest 1	ndom arch 1	50	kick 1 5
0.495	1.230	oldest 1	ndom sol 1	20	kick 1 3
0.505	1.112	rand 10	ndom arch 10	1000	restart
0.508	1.084	rand 10	all	50	kick_all 3
0.512	1.044	rand 10	ndom arch 1	1000	kick_all 3
0.513	1.036	all	ndom arch 3	20	restart
0.514	0.753	rand 1	all_imp arch	100	restart
0.515	0.662	rand 1	imp arch 3	100	restart
0.515	0.662	rand 1	imp arch 3	50	restart
0.517	0.328	rand 1	imp sol 1	100	restart
0.519	0.115	rand 1	imp arch 1	20	kick 10 10
0.520	0.090	oldest 1	imp arch 1	20	kick 1 3
		(medium correlation)			
0.503	1.209	oldest 1	ndom sol 3	1000	kick 5 3
0.503	1.186	oldest 1	ndom sol 3	100	kick 5 10
0.505	1.125	oldest 1	ndom arch 1	50	kick 10 10
0.509	1.056	rand 10	all	50	restart
0.509	0.945	oldest 1	ndom sol 3	20	kick 1 3
0.517	0.882	all	ndom arch 3	20	kick 1 5
0.520	0.855	rand 1	imp arch 3	1000	restart
0.521	0.723	rand 1	imp arch 3	100	kick_all 5
0.521	0.723	rand 1	imp arch 3	1000	kick_all 5
0.521	0.723	rand 1	imp arch 3	20	kick_all 10
0.522	0.493	rand 1	imp sol 1	100	restart
0.522	0.493	rand 1	imp sol 1	50	restart
0.522	0.478	rand 1	imp arch 1	1000	restart
0.522	0.478	rand 1	imp arch 1	20	restart
0.525	0.195	rand 1	imp arch 1	20	kick_all 10
0.527	0.116	oldest 1	imp arch 1	20	kick 10 10
0.527	0.028	rand 1	imp arch 1	100	kick 1 5
0.527	0.028	rand 1	imp sol 1	20	kick 1 10
0.527	0.028	rand 1	imp sol 1	50	kick 1 10
0.528	0.025	oldest 1	imp arch 1	20	kick 1 3
		(no correlation)			
0.521	0.971	rand 10	ndom arch 3	1000	kick 5 10
0.521	0.970	rand 10	ndom arch 3	1000	kick 10 3
0.524	0.841	rand 3	imp_ndom sol 10	50	kick_all 10
0.531	0.806	newest 1	ndom arch 3	50	kick 5 10
0.533	0.806	rand 10	all	20	kick 5 10
0.540	0.762	rand 1	imp sol 3	100	kick 10 3
0.541	0.586	all	imp arch 1	100	restart
0.541	0.586	all	imp arch 1	1000	restart
0.541	0.586	all	imp arch 1	50	restart
0.541	0.586	newest 10	imp arch 1	100	restart
0.542	0.579	oldest 3	imp sol 1	20	restart
0.542	0.579	rand 10	imp sol 1	1000	restart
0.542	0.579	rand 10	imp sol 1	50	restart
0.542	0.579	rand 3	imp sol 1	1000	restart
0.545	0.318	rand 1	imp arch 1	20	kick 10 10
0.545	0.317	rand 1	imp sol 1	1000	kick 10 10
0.548	0.227	oldest 1	imp sol 1	1000	kick 10 5
0.550	0.129	rand 1	imp sol 1	50	kick 1 10

Table 3.8: PFSP 100 jobs 20 machines *(optimised configurations)*

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.354	1.885	oldest 1	ndom	arch 1 1000	kick 1 10
0.356	1.760	oldest 1	ndom	arch 1 100	kick 1 3
0.365	1.511	oldest 1	ndom	sol 1 20	kick 10 5
0.375	1.468	oldest 1	ndom	arch 1 20	kick 1 3
0.375	1.499	rand 3	all	100	kick 10 10
0.375	1.429	oldest 1	ndom	sol 1 20	restart
0.377	1.306	rand 10	all	20	restart
0.379	1.268	newest 10	ndom	arch 3 20	restart
0.380	1.240	newest 10	ndom	arch 10 20	restart
0.380	0.996	rand 1	all_imp	arch 1000	kick 10 10
0.380	0.982	rand 1	all_imp	arch 1000	restart
0.382	0.796	rand 1	imp	arch 3 1000	restart
0.382	0.796	rand 1	imp	arch 3 20	restart
0.383	0.779	rand 3	all_imp	sol 50	kick 5 3
0.384	0.120	rand 1	imp	arch 1 20	restart
0.384	0.071	rand 1	imp	arch 1 50	kick_all 5
		(medium correlation)			
0.369	1.748	oldest 1	ndom	arch 1 1000	kick 1 5
0.369	1.747	oldest 1	ndom	arch 1 1000	kick 1 3
0.369	1.746	oldest 1	ndom	arch 1 1000	kick 1 10
0.367	1.594	oldest 1	ndom	arch 1 100	kick 5 3
0.370	1.551	oldest 1	ndom	arch 1 100	kick 1 3
0.370	1.548	oldest 1	ndom	arch 1 100	kick 1 5
0.372	1.440	oldest 1	ndom	arch 1 50	kick 10 3
0.378	1.431	rand 1	ndom	arch 10 1000	restart
0.379	1.325	oldest 1	ndom	arch 1 20	kick 10 10
0.379	1.241	rand 3	ndom	arch 10 100	kick_all 3
0.381	1.083	rand 3	imp_ndom	arch 10 100	kick 5 10
0.381	1.058	rand 1	imp_ndom	sol 10 100	kick 5 5
0.381	1.050	rand 10	imp_ndom	arch 10 100	kick_all 10
0.382	1.015	rand 3	imp_ndom	arch 10 50	restart
0.382	1.007	rand 10	imp_ndom	sol 10 50	kick_all 10
0.384	0.962	rand 10	all	50	kick 1 10
0.385	0.946	all	ndom	arch 3 20	kick 1 3
0.387	0.862	rand 10	all	20	kick_all 10
0.390	0.166	rand 1	imp	arch 1 1000	restart
0.392	0.057	rand 1	imp	sol 1 50	kick 10 10
0.393	0.006	rand 1	imp	sol 1 20	kick 1 10
0.393	0.006	rand 1	imp	sol 1 50	kick 1 10
		(no correlation)			
0.387	1.214	rand 1	ndom	arch 1 1000	restart
0.387	1.169	rand 3	ndom	arch 3 1000	kick_all 3
0.387	1.167	rand 3	ndom	arch 3 1000	kick 5 5
0.387	1.165	rand 3	ndom	arch 3 1000	kick_all 10
0.388	0.996	rand 10	ndom	arch 10 1000	restart
0.388	0.989	rand 10	ndom	arch 10 1000	kick 1 10
0.389	0.957	rand 1	ndom	arch 10 100	kick 5 3
0.389	0.948	rand 10	ndom	arch 10 100	kick_all 3
0.389	0.942	rand 10	ndom	arch 10 100	kick 1 10
0.390	0.923	rand 1	imp_ndom	arch 3 100	kick_all 3
0.390	0.922	rand 1	imp_ndom	arch 3 100	restart
0.393	0.804	all	ndom	arch 3 50	kick 10 5
0.403	0.148	rand 1	imp	sol 1 50	restart
0.403	0.150	rand 1	imp	arch 1 1000	restart
0.403	0.150	rand 1	imp	arch 1 50	restart
0.408	0.148	rand 1	imp	sol 1 50	kick 10 10
0.408	0.130	rand 1	imp	arch 1 1000	kick_all 3
0.411	0.074	all	imp	arch 1 100	kick 1 5
0.411	0.074	all	imp	arch 1 50	kick 1 5
0.411	0.074	all	imp	sol 1 1000	kick 1 3
0.411	0.074	all	imp	sol 1 20	kick 1 10
0.411	0.074	newest 10	imp	arch 1 100	kick 1 3

Table 3.9: TSP 50 cities *(optimised configurations)*

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.157	0.804	oldest 1	ndom sol 3	1000	restart
0.157	0.804	oldest 1	ndom sol 3	100	restart
0.157	0.732	rand 1	ndom sol 1	50	restart
0.158	0.670	rand 1	ndom sol 3	1000	restart
0.158	0.669	rand 1	ndom sol 3	100	restart
0.158	0.646	rand 1	ndom sol 3	50	restart
0.159	0.612	rand 1	ndom sol 10	50	restart
0.159	0.606	all	ndom arch 10	50	restart
0.160	0.591	oldest 1	ndom sol 3	1000	kick 10 3
0.162	0.562	oldest 1	imp sol 10	50	restart
0.162	0.452	oldest 1	imp arch 1	100	restart
		(medium correlation)			
0.163	0.662	rand 1	ndom arch 1	1000	kick_all 3
0.165	0.658	rand 1	ndom arch 10	1000	kick 10 3
0.167	0.658	rand 1	ndom arch 10	1000	kick 10 10
0.167	0.657	rand 1	ndom arch 10	1000	restart
		(no correlation)			
0.185	0.676	rand 1	ndom arch 1	1000	kick_all 5
0.185	0.676	rand 1	ndom arch 1	1000	kick_all 10
0.185	0.676	rand 1	ndom arch 1	1000	kick_all 3
0.190	0.655	rand 1	ndom arch 1	1000	restart
0.195	0.655	rand 1	ndom sol 1	1000	kick_all 10
0.196	0.625	rand 3	ndom sol 3	1000	restart
0.207	0.617	rand 1	ndom sol 1	1000	restart

Table 3.10: TSP 100 cities *(optimised configurations)*

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.115	0.629	rand 1	ndom sol 3	1000	kick 10 3
0.115	0.6232	rand 3	ndom sol 10	1000	kick_all 3
		(medium correlation)			
0.123	0.816	rand 10	imp_ndom arch 1	1000	kick 10 10
0.123	0.662	rand 3	ndom sol 10	1000	kick_all 10
0.123	0.662	rand 3	ndom sol 10	1000	restart
0.123	0.661	rand 3	ndom sol 10	1000	kick 1 3
0.123	0.661	rand 3	ndom sol 10	1000	kick_all 3
0.123	0.660	rand 3	ndom sol 10	1000	kick_all 5
0.125	0.654	rand 1	ndom sol 3	1000	restart
0.125	0.653	rand 1	ndom sol 1	1000	kick_all 5
0.126	0.644	rand 1	ndom sol 1	1000	restart
		(no correlation)			
0.139	0.956	oldest 1	imp_ndom arch 1	1000	kick 5 3
0.139	0.956	oldest 1	imp_ndom arch 1	1000	kick 10 3
0.139	0.955	oldest 1	imp_ndom arch 1	1000	kick 5 5
0.139	0.902	oldest 3	ndom sol 10	1000	kick 10 10
0.139	0.901	oldest 3	ndom sol 10	1000	kick 10 5
0.140	0.885	oldest 1	ndom sol 10	1000	kick_all 5
0.140	0.857	oldest 10	ndom sol 10	1000	kick 10 10
0.141	0.654	rand 10	ndom sol 10	1000	kick 10 3
0.141	0.653	rand 10	ndom sol 10	1000	kick 10 5
0.145	0.645	rand 1	ndom sol 3	1000	kick 10 3
0.145	0.643	rand 3	ndom sol 3	1000	kick 10 5
0.145	0.636	rand 10	ndom sol 3	1000	kick_all 3

Table 3.11: QAP 50 facilities (optimised configurations)

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.319	0.893	oldest 1	ndom sol 1	1000	restart
0.319	0.882	oldest 1	ndom sol 1	100	restart
0.319	0.872	oldest 1	ndom sol 1	20	restart
0.320	0.443	oldest 1	imp arch 3	50	restart
0.320	0.343	rand 1	imp arch 1	50	restart
0.321	0.309	oldest 1	imp arch 1	100	restart
0.321	0.301	rand 1	imp sol 1	50	restart
0.321	0.266	oldest 1	imp sol 1	20	restart
0.321	0.169	oldest 1	imp arch 1	20	kick_all 10
		(medium correlation)			
0.321	0.884	oldest 1	ndom arch 1	1000	restart
0.321	0.876	oldest 1	ndom arch 1	100	restart
0.321	0.861	oldest 1	ndom sol 1	100	kick 1 3
0.321	0.849	oldest 1	ndom sol 1	1000	kick 1 3
0.321	0.848	oldest 1	ndom sol 1	100	kick 1 5
0.322	0.498	oldest 1	imp arch 3	100	kick 5 3
0.322	0.172	oldest 1	imp arch 1	100	kick 5 3
		(no correlation)			
0.322	0.797	rand 10	ndom arch 3	1000	restart
0.322	0.787	rand 1	ndom arch 3	1000	restart
0.322	0.784	rand 1	ndom arch 3	1000	kick 10 3
0.322	0.782	rand 1	ndom arch 3	1000	kick_all 10
0.322	0.770	rand 1	ndom sol 10	1000	kick_all 10
0.322	0.716	rand 1	imp sol 1	100	restart
0.322	0.710	oldest 1	imp sol 1	50	restart

for the PFSP and TSP, and vary with instance size as well as correlation between the objectives. The `restart` perturbation strategy is favored for small, 50-facility instances, while kick-based perturbation strategies appear to work better for the larger 100-facility instances. Interestingly, the larger instances seem to be amenable to a wider range of exploration strategies. However, the degree of objective correlation affects the choice of exploration strategy; e.g., for the larger instances, the `imp_ndom` exploration strategy is only chosen when the objectives are uncorrelated. Similarly, we found that bounding the archive size appears to work only well for sufficiently correlated objectives, while the same observation holds for the `oldest` selection strategy.

3.4.2.2 Configurator Performance

Table 3.13 summarises the performance of our three AAC approaches, HV, $HV{+}\Delta'$, and $HV||\Delta'$, on all 18 scenarios, and details the number of final configurations and the range of hypervolume and Δ' indicator values.

Clearly, $HV||\Delta'$ produces much larger sets of configurations than HV and $HV{+}\Delta'$, in particular for the PFSP. While $HV{+}\Delta'$ and $HV||\Delta'$ achieve overall similar hypervolume values to the dedicated HV approach, on some scenarios (highly correlated PFSP and uncorrelated 100-city TSP), HV achieves the best hypervolume, as could be expected. Surprisingly, on the uncorrelated 100−job PFSP scenario, the $HV||\Delta'$ approach performs best in terms of hypervolume. Regarding the complementary Δ' spread indicator, $HV||\Delta'$ generally achieves much better results, which are only occasionally matched by the $HV{+}\Delta'$ approach, when the

Table 3.12: QAP 100 facilities (optimised configurations)

1-HV	Δ'	Selection	Explo.	Arch.	Perturb.
		(high correlation)			
0.319	0.839	oldest 1	ndom sol 1	50	restart
0.320	0.815	oldest 1	ndom sol 1	20	restart
0.320	0.525	rand 1	imp arch 3	1000	kick 10 3
0.320	0.297	rand 1	imp sol 3	50	kick 5 3
0.320	0.100	rand 1	imp sol 1	100	kick_all 10
0.321	0.090	newest 10	imp arch 1	1000	kick_all 10
0.321	0.080	all	imp sol 1	50	kick 10 3
		(medium correlation)			
0.320	0.907	oldest 1	ndom arch 1	1000	restart
0.320	0.886	oldest 1	ndom arch 1	1000	kick 10 3
0.320	0.878	oldest 1	ndom sol 1	1000	kick 10 5
0.320	0.868	oldest 1	ndom sol 1	1000	kick 1 5
0.320	0.838	oldest 1	ndom arch 1	1000	kick 1 3
0.321	0.808	rand 1	imp_ndom arch 1	1000	kick_all 3
0.321	0.797	rand 3	imp_ndom arch 1	1000	kick_all 3
0.321	0.408	rand 1	all_imp sol	1000	kick_all 5
0.321	0.302	rand 1	all_imp sol	100	kick 5 5
0.321	0.272	rand 1	all_imp sol	100	kick 10 10
0.321	0.157	rand 1	imp sol 3	50	kick_all 5
0.321	0.000	rand 1	imp arch 1	50	kick 1 10
		(no correlation)			
0.321	0.713	rand 3	imp_ndom arch 1	1000	restart
0.321	0.710	rand 3	imp_ndom arch 1	1000	kick 1 3
0.321	0.708	rand 1	imp_ndom arch 1	1000	kick_all 3
0.321	0.702	rand 3	imp_ndom arch 1	1000	restart
0.321	0.702	rand 10	imp_ndom arch 1	1000	kick_all 10
0.321	0.261	rand 1	imp sol 3	50	kick 5 3
0.321	0.000	rand 1	imp arch 1	1000	kick_all 5

direction of aggregation is compatible with the shape of the optimised front of solutions; but to ensure this is the case, a costly preliminary analysis is required to permit appropriate normalisation of hypervolume and Δ' spread. These observations are consistent with experiments 1 (see Section 3.3, as for correlation between objectives, there is no clear overall impact on the three AAC approaches. We note, however, that the single-objective *HV* approach clearly achieves the best hypervolume for the highly correlated PFSP scenarios.

3.5 Automatic Configuration of a Dynamic MO-SLS

The previous experiments showed the interest of our multi-objective approach to automatically configure a multi-objective stochastic local search algorithm. In automatic algorithm configuration, only one configuration is recommended for the entire run while it is commonly admitted that different configurations should be used. Indeed, a problem has different local structures in different regions of the search space. *Parameter control* approaches (B. Doerr and C. Doerr 2018; Eiben et al. 1999; Karafotias et al. 2015) use techniques such as multi-armed bandits (Fialho et al. 2009) or adaptive pursuit (Thierens 2005) to dynamically determine good parameter settings in response to observations made while trying to solve a given problem instance. However, the number of configurations handled by such approaches is very limited, contrary to AAC approaches.

Table 3.13: AAC performance: number of final configurations and objective ranges

# $1-HV$ values	Δ' values		# $1-HV$ values	Δ' values
PFSP 50 jobs 20 machines			**PFSP 100 jobs 20 machines**	
		(high correlation)		
3	**0.494** – 0.495	1.230 – 1.385	3	**0.354** – 0.375	1.499 – 1.884
5	0.495 – 0.519	0.402 – 1.523	3	0.355 – 0.375	1.485 – 1.930
13	0.495 – 0.519	0.090 – 1.454	15	0.355 – 0.384	0.071 – 1.930
		(medium correlation)		
1	0.503	1.186	3	**0.369** – 0.370	1.551 – 1.748
4	**0.503** – 0.508	1.056 – 1.209	8	**0.369** – 0.382	1.015 – 1.746
19	0.503 – 0.527	0.025 – 1.187	14	0.370 – 0.393	0.006 – 1.548
		(no correlation)		
1	**0.521**	0.970	1	0.387	1.167
1	**0.521**	0.971	5	0.387 – 0.390	0.822 – 1.170
17	0.521 – 0.550	0.129 – 0.975	20	**0.386** – 0.411	0.074 – 1.21
TSP 50 cities			**TSP 100 cities**	
		(high correlation)		
3	**0.157** – 0.158	0.669 – 0.804	2	**0.115** – 0.115	0.623 – 0.629
4	0.158 – 0.160	0.591 – 0.670	2	0.115	0.627 – 0.629
6	**0.157** – 0.164	0.452 – 0.732	2	0.115 – 0.115	0.623 – 0.630
		(medium correlation)		
1	0.167	0.657	2	**0.123**	0.662 – 0.816
2	**0.167** – 0.167	0.658 – 0.662	3	**0.123**	0.660 – 0.662
2	**0.167** – 0.167	0.658 – 0.663	5	**0.123** – 0.126	0.644 – 0.815
		(no correlation)		
1	**0.185**	0.676	4	**0.139** – 0.140	0.884 – 0.956
1	**0.185**	0.676	5	**0.139** – 0.141	0.654 – 0.9566
5	**0.185** – 0.207	0.617 – 0.676	8	**0.139** – 0.145	0.636 – 0.955
QAP 50 facilities			**QAP 100 facilities**	
		(high correlation)		
2	**0.319** – 0.319	0.881 – 0.882	1	**0.319**	0.839
2	**0.319** – 0.319	0.872 – 0.893	1	**0.319**	0.821
8	**0.319** – 0.321	0.169 – 0.891	6	0.320 – 0.321	0.080 – 0.815
		(medium correlation)		
3	**0.321** – 0.321	0.849 – 0.884	1	**0.320**	0.868
3	**0.321** – 0.321	0.848 – 0.878	5	**0.320** – 0.321	0.000 – 0.886
3	0.321 – 0.322	0.172 – 0.861	8	**0.320** – 0.321	0.000 – 0.907
		(no correlation)		
2	**0.322**	0.787 – 0.794	4	**0.321**	0.702 – 1.219
2	**0.322**	0.781 – 0.796	3	**0.321**	0.710 – 1.249
5	**0.322** – 0.322	0.710 – 0.7973	4	**0.321** – 0.321	0.000 – 0.708

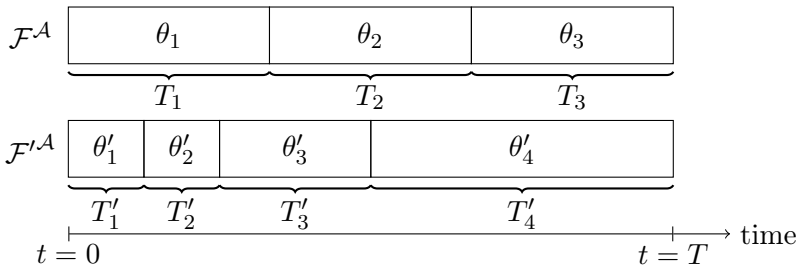


Figure 3.5: Two examples of dynamic algorithms, $\mathcal{F}^{\mathcal{A}}$ and $\mathcal{F}'^{\mathcal{A}}$

Therefore, we want to take advantage of both AAC and parameter control approaches. We introduce a framework that successively runs several configurations, in the form of a static pipeline, which we configure using a standard, general-purpose AAC procedure.

3.5.1 A Dynamic Algorithm Framework

Given a configurable algorithm $\mathcal{A}$ and its configuration space Θ, we use $\mathcal{A}_{\theta,T}$ to denote $\mathcal{A}$ under configuration $\theta \in \Theta$ with cut-off time T. Then, we define the dynamic algorithm $\mathcal{F}^{\mathcal{A}}_{(\theta_i,T_i)^k}$ as a pipeline with k stages, which sequentially runs $\mathcal{A}_{\theta_1,T_1}, \mathcal{A}_{\theta_2,T_2}, \ldots, \mathcal{A}_{\theta_k,T_k}$. Specifically, when applied to a multi-objective optimization problem, we first run $\mathcal{A}$ under configuration θ_1, starting from a initial set of solutions, up to time T_1. At that point, we switch to configuration θ_2 and continue our computation from the current set of solutions, with a cut-off time of T_2. We note that $\mathcal{A}$ is *not* restarted when switching between configurations. Overall, the maximum running time of the dynamic algorithm is then $T = \sum_{i=1}^{k} T_i$.
Figure 3.5 depicts two examples of dynamic algorithms $\mathcal{F}^{\mathcal{A}}$ and $\mathcal{F}'^{\mathcal{A}}$. While $\mathcal{F}$ uses $k = 3$ configurations to divide the total time budget into three intervals of equal duration, $\mathcal{F}'$ uses $k = 4$ configurations, of which two are run quickly in the beginning, after which more time is allocated to last two configurations.
The configuration space of our framework comprises the Cartesian product Θ^k, the time budgets $T_1, \ldots, T_k$ and the integer $k \geq 1$. For $k = 1$, our framework degenerates to the original, static target algorithm $\mathcal{A}$.

3.5.2 Automatic Configuration of our Framework

The purpose of this work is to assess the performance gains that can be obtained by switching between different configurations of an algorithm $\mathcal{A}$ while it is running. Towards this end, we use a general-purpose, static algorithm configurator to configure the framework introduced in the previous section. Since the size of the configuration space exponentially increases with the maximum number of pipeline stages, K, we only consider a fixed number s_k of different cut-off times for each stage, where k is the number of actual pipeline stages used in a specific instantiation of our framework. This leads to a configuration space of size $\sum_{k=1}^{K} s_k \cdot |\Theta|^k$. Using this approach, we can also assess the influence of K and s_k (for $k = 1, \ldots, K$) on the performance achieved by automatically configuring our dynamic algorithm framework.

3.5.3 Related Work

In addition to being conceptually related to adaptive algorithms or hyper-heuristics, since it enables modifications of the configuration of an algorithm while it is running, our approach also bears resemblance to per-instance algorithm scheduling (M. Lindauer et al. 2016). There are, however, several major differences. Firstly, per-instance algorithm scheduling uses instance features to determine which of a given set of distinct algorithms to run, one after the other, on a given problem instance; in contrast, our approach uses different configurations of a single algorithm and does not require instance features. Secondly, in per-instance algorithm scheduling, results are not passed from one stage of the schedule to the next, while in our pipeline approach, each stage continues from the result of the previous stage – as explained previously, it can thus be seen as a single algorithm whose parameter configuration changes while running on a given problem instance. Finally, the primary goal of per-instance algorithm scheduling is *robustness* resulting from performance complementarity between the algorithms in the schedule; the goal of our approach is to achieve improvements over the performance of the static version of the given target algorithm, which uses a single configuration for the entire run, based on the idea that different configurations are best suited for different phases of solving a given problem instance.

3.5.4 Experimental Protocol

3.5.4.1 Problem and Benchmark

Experiments have been conducted on the permutation flowshop scheduling problem minimizing the makespan and the sum of flowtimes (see the formal definition in Section 3.2.4.1). We evaluate our approach on 6 sets of 10 Taillard instances each, with 20 jobs and 20 machines, 50 jobs and 5 machines, 50 jobs and 10 machines, 50 jobs and 20 machines, 100 jobs and 10 machines and 100 jobs and 20 machines, respectively.

3.5.4.2 Configuration Space

Table 3.14: Configuration space of the MO-SLS

Phase	Parameter	Parameter values
Selection	`select-strat`	{`all`, `rand`, `newest`, `oldest`}
Selection	`select-size`	1
Exploration	`explor-strat`	{`imp`, `imp_ndom`, `ndom`, `all`, `all_imp`}
Exploration	`explor-ref`	`arch`
Exploration	`explor-size`	5
Archive	`bound-strat`	`unbounded`
Perturbation	`perturb-strat`	{`kick`, `kick_all`, `restart`}
Perturbation	`perturb-size`	{1}
Perturbation	`perturb-strength`	{3}

The core algorithm of the dynamic framework is the MO-SLS already used in this chapter. Table 3.14 shows all strategies we considered (all numerical values have been fixed); these give rise to 60 ($4 \times 5 \times 3$) different configurations of MO-SLS.
In addition to the own configuration space of the MO-SLS, some parameters of the dynamic framework have to be configured. In the experiments we evaluate the framework with K up to 3 and the following ways of dividing the overall running times between the pipeline stages: For $K = 2$, we use $(T_1, T_2) = (1/4, 3/4) \cdot T$, $(1/2, 1/2) \cdot T$ and $(3/4, 1/4) \cdot T$, where T is the overall cut-off time, while for $K = 3$, we consider $(T_1, T_2, T_3) = (1/3, 1/3, 1/3) \cdot T$, $(1/4, 1/4, 1/2) \cdot T$ and $(1/2, 1/4, 1/4) \cdot T$. Therefore, whilst the basic MO-SLS algorithm has 60 distinct configurations, the dynamic MO-SLS algorithm, dubbed D-MO-SLS, has $60 + 3 \cdot 60^2 \approx 1.1 \cdot 10^4$ configurations for $K = 2$, and $60 + 3 \cdot 60^2 + 3 \times 60^3 = 6.6 \cdot 10^5$ configurations for $K = 3$ stages. We note that this configuration space is very large compared to on-line algorithms from the literature, which typically involve only very few configurations. In our experiments, we choose an overall cut-off time of $T = N^2 \cdot M/1000$ for D-MO-SLS (with N and M the number of jobs and machines respectively).

3.5.4.3 AAC Protocol

In Section 3.3, we showed that a multi-objective AAC is the best approach to automatically configure multi-objective algorithms such as MO-SLS. Therefore, in order to configure D-MO-SLS, we also use MO-ParamILS and the hypervolume and spread indicators as described in Section 3.2.1.
To obtain training sets to be used as the basis for automatic configuration, we generated uniformly at random a set 100 instances for each of the six instance sizes we consider, following the same protocol as Taillard. Since MO-ParamILS is a stochastic algorithm, we perform 20 independent runs for each configuration scenario, each with 1000 and 10 000 runs of D-MO-SLS for $K = 2$ and $K = 3$, respectively. Then, the best of the 20 resulting D-MO-SLS configurations (according to performance on the respective training set) was evaluated on the 10 Taillard instances in each of our testing sets, based on 15 independent runs. The performance indicators – hypervolume and spread – reported for a single D-MO-SLS configuration have been obtained by averaging the respective values over the 15 independent runs and the 10 instances per set.

3.5.5 Results and Analysis

First, we present results for D-MO-SLS, our dynamic version of MO-SLS, for the PFSP for $K = 2$ and $K = 3$ pipeline stages, *i.e.*, one or two changes in configuration during each run. Next, we compare the results for D-MO-SLS with those for static MO-SLS.

3.5.6 Evaluation of Dynamic MO-SLS

Table 3.15 shows the number of D-MO-SLS configurations in the Pareto-optimal sets obtained from automatic configuration using MO-ParamILS; specifically, for $K = 2$ and $K = 3$, we report the number of non-dominated configurations with $k = 1$, 2 and 3 pipeline stages. For example, for the `20x20` scenario and $K = 3$, we obtained 1 configuration for

Table 3.15: Number of non-dominated D-MO-SLS configurations determined through automatic configuration.

Instances NxM	$K=1$	$K=2$		$K=3$		
		$k=1$	$k=2$	$k=1$	$k=2$	$k=3$
20x20	20	5	4	1	7	9
50x5	9	-	7	-	5	9
50x10	9	-	7	-	10	8
50x20	11	-	12	-	3	8
100x10	8	1	9	-	5	5
100x20	8	-	13	N/A	N/A	N/A

static MO-SLS, 7 for dynamic MO-SLS with 2 pipeline stages and 9 for dynamic MO-SLS with 3 pipeline stages. For 8 of the 11 benchmark sets considered ($K = \{2, 3\}$ and 6 instance sizes), all non-dominated D-MO-SLS configurations obtained from MO-ParamILS have at least 2 pipeline stages $k \geq 2$, which clearly indicates the performance advantage gained by switching between configurations during a single run of MO-SLS.

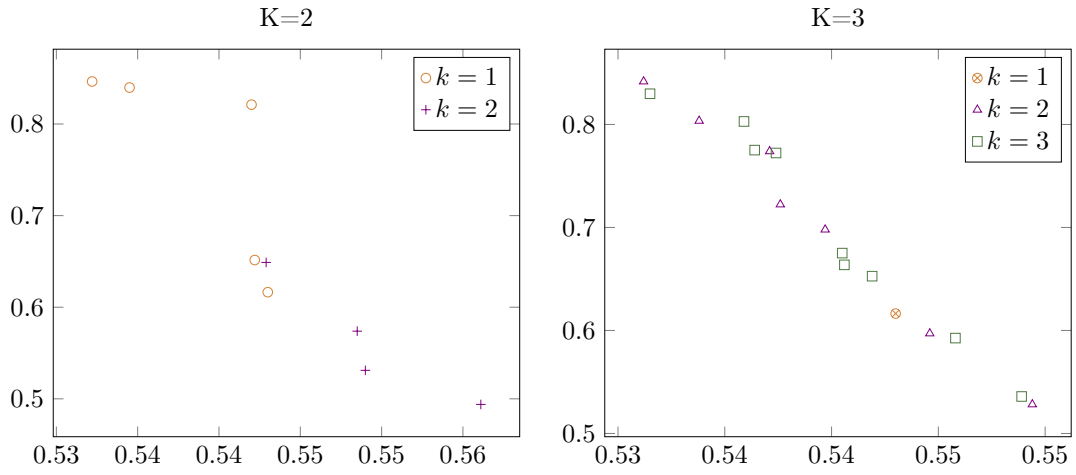


Figure 3.6: Performance of Pareto-optimal D-MO-SLS configurations for the `20x20` benchmarks.

Figure 3.6 shows the Pareto fronts of D-MO-SLS configurations obtained in our experiments with $K = 2$ (left) and $K = 3$ (right), respectively, for the benchmark instances with 20 jobs and 20 machines. For $K = 2$, static MO-SLS ($k = 1$) achieves better hypervolume, while D-MO-SLS(2) obtains better spread; for $K = 3$, on the other hand, D-MO-SLS(2) and D-MO-SLS(3) yield better results w.r.t. both indicators. Figure 3.7 shows the our results for benchmark instances with 50 jobs and 20 machines. As also seen in Table 3.15, no configurations from static MO-SLS are found in the final Pareto sets; furthermore, the sets of configurations from both D-MO-SLS scenarios are well distributed over the Pareto front.

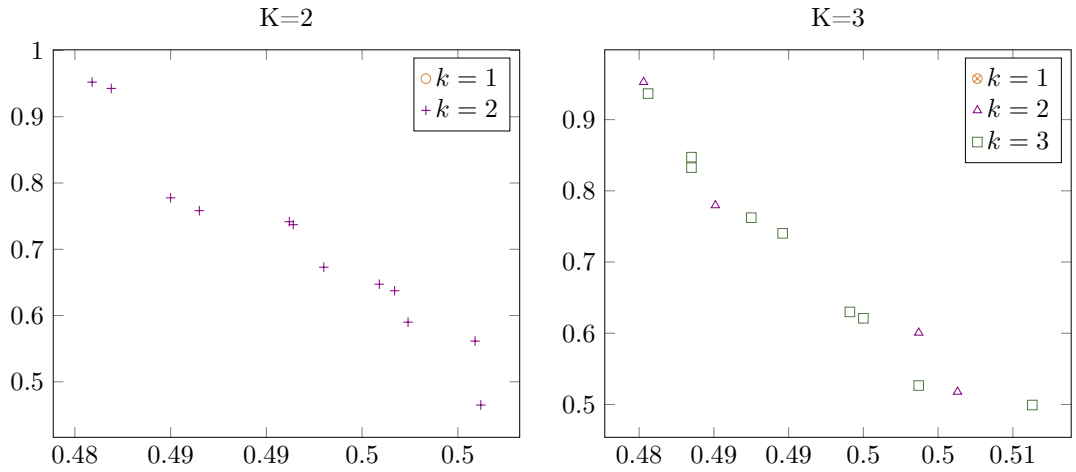


Figure 3.7: Performance of Pareto-optimal D-MO-SLS configurations for the `50x20` benchmarks.

3.5.7 Performance of the Dynamic *vs* Static MO-SLS

In this section, we further assess the performance of our dynamic MO-SLS algorithm against static MO-SLS. Since there are only 60 configurations of static MO-SLS, we were able to evaluate all of them. Figure 3.8 shows the Pareto fronts of configurations for static MO-SLS ($K = 1$) *vs* dynamic MO-SLS for $K = 2$ and $K = 3$. Only few of the 60 configurations of MO-SLS ended up in the Pareto-optimal sets for each of our benchmarks. We further note that for each instance size, the Pareto fronts obtained for $K = 2$ and $K = 3$ are of roughly similar size. For `50x10`, `50x20`, `100x10` and `100x20`, the configurations obtained for D-MO-SLS are better distributed along the respective fronts. For `100x10` and `100x20`, the fronts obtained by static MO-SLS ($K = 1$) are very poorly distributed. Most of the configurations are tightly clustered; this is particularly pronounced for `100x20`, where there are two types of configurations that obtain either good hypervolume or good spread, but never both. The configurations for dynamic MO-SLS ($K \geq 2$), on the other hand, are well distributed and cover a broad range of tradeoffs between the objectives. Furthermore, the configurations for $K = 1$ are all dominated by those for $K \geq 2$. For the smallest instance size, `50x5`, we observed a large improvement in hypervolume, while spread remains comparable; this effect is less obvious for the `20x20` and `50x20` instances. For `50x20`, static MO-SLS dominates parts of the fronts for dynamic MO-SLS, likely as a result of the large configuration spaces for $K \geq 2$; nevertheless, for $K \geq 2$, more homogeneous Pareto fronts of configurations are obtained. For `20x20`, all three fronts are quite close to each other and reasonably well distributed, with the configurations of dynamic MO-SLS ($K \geq 2$) filling some of the gaps in the front obtained for static MO-SLS. We note that, even though the fronts for $K = 2$ and $K = 3$ are roughly similar in size, the one for $K = 3$ contains more configurations and is overall preferable.

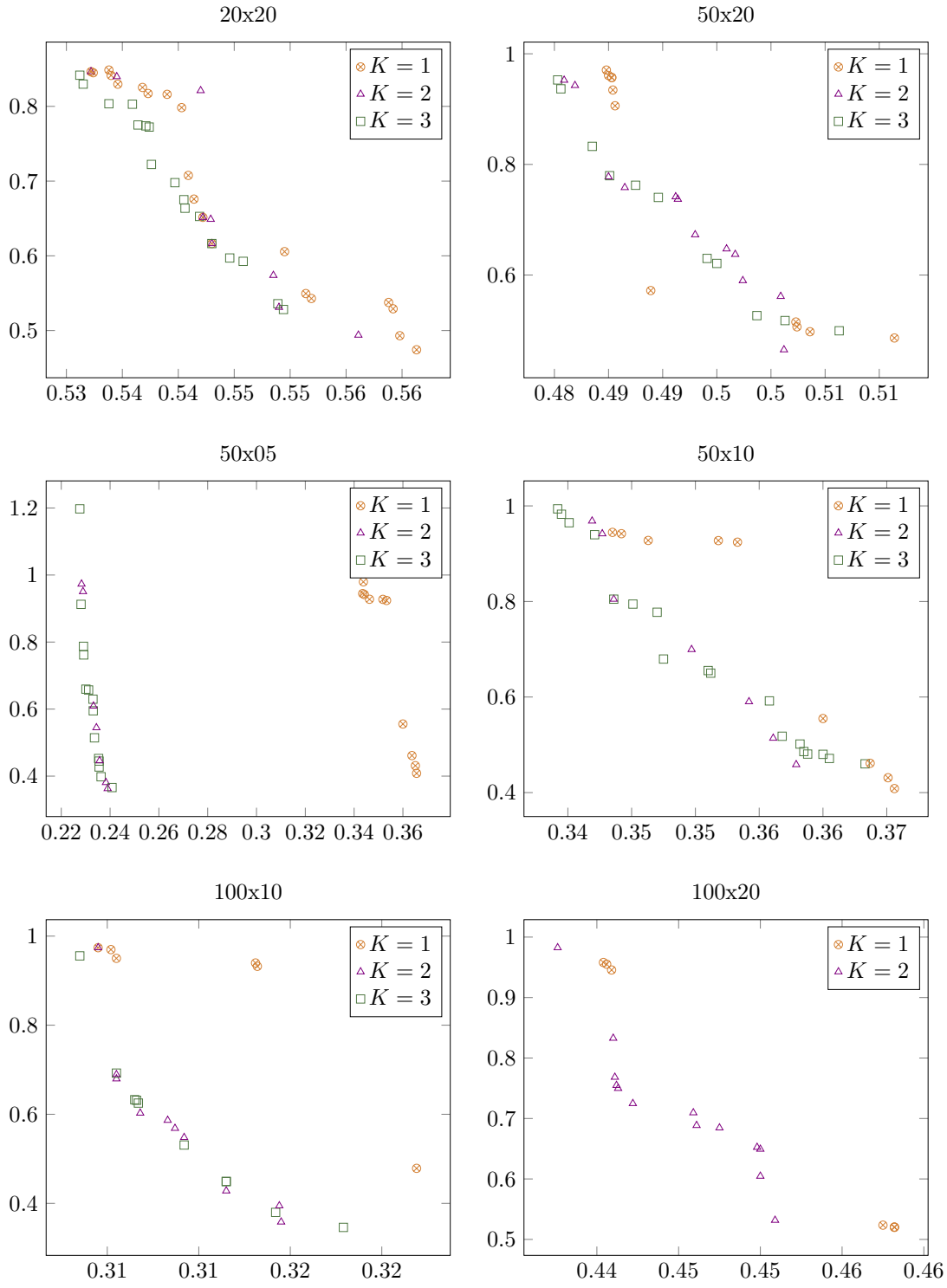


Figure 3.8: Pareto fronts of static ($K = 1$) and dynamic ($K \geq 2$) MO-SLS configurations; for details see text.

3.6 Conclusion and Perspectives

Summary In this chapter, we presented our contribution on multi-objective automatic configuration algorithm (MO-AAC) and on automatic configuration of multi-objective stochastic local search (MO-SLS) algorithms. On the one hand, we proposed a formal definition of MO-AAC, based on the classical definition of AAC, where we introduced the possibility of configuring an algorithm according to two or more criteria. Then, we presented MO-ParamILS, the multi-objective extension of the configurator ParamILS and we proposed a MO-AAC protocol adapted to the stochastic behaviour of MO-ParamILS. On the other hand, we conducted two sets of experiments on classical permutation problems in order to (i) evaluate the benefit of using a multi-objective approach as opposed to single-objective ones regarding the automatic configuration of a multi-objective local search algorithm and, (ii) study the impact of the correlation between objectives on the performance of both single- and multi-objective AAC. The performance was evaluated with the hypervolume and the spread indicators. Our MO-AAC approach is based on the Pareto comparison of both indicators while the first SO-AAC approach only considers the hypervolume and the second one, an aggregation between the two indicators. The first experiments showed that the MO-AAC approach gives better results than the two SO-AAC approaches. Indeed, it was able to cover the entire Pareto-front of configurations, and found the same sets of configurations with a better hypervolume than the two other approaches. The second experiments showed that the correlation between objectives can impact the optimal configuration of a given problem as much as problem size, strongly implying that automatic algorithm configuration should be performed for each instance class independently. Likewise, the correlation between objectives did not impact the average performance between the three AAC approaches. Finally, we presented a dynamic framework that switches between different configurations during the execution in order to better adapt to local structures. However, parameter control approach generally deals with a very few numbers of parameters while algorithms are generally impacted by several parameters. We proposed to use AAC to *a priori* configure this framework. The experiments showed the interest of this approach taking the advantages of both AAC and parameter control.

Automatic Configuration of Hybrid Multi-objective SLS In this chapter, we validated the interest of using a multi-objective approach to configure a multi-objective SLS. However, we used a simpler template of SLS than the unified structure proposed in Chapter 2. A natural perspective would be to consider the entire structure of SLS and enable the configuration of hybrid SLS like it was proposed in Chapter 1. Therefore, we might validate our approach to configure more complex algorithms and compare them to the state-of-the art algorithms for each problem solved.

Improvement of MO-ParamILS MO-ParamILS is based on a particular MO-SLS, chosen from our knowledge about the different strategies in 2015. Following the initial design of MO-ParamILS, we made a literature review of MO-SLS and proposed a unified structure (see Chapter 2) which allowed us to identify numerous strategies. Recently, Cáceres and Stützle (2017) explored variable neighborhood search strategies in ParamILS

and showed that results are promising. We think that a similar study for MO-ParamILS could increase its performance considering the categorical parameters. Besides, the numerical parameters can be better managed. Indeed, both ParamILS and MO-ParamILS discretize numerical parameters to facilitate the use of a neighborhood operator. Franzin et al. (2018) studied the effect of transformations of numerical parameters in automatic algorithm configuration. They showed that a proper transformation can strongly improve the performance of the AAC. This work may also benefit to MO-ParamILS. We, recently, started working with Leslie Pérez Cáceres (Pontifical Catholic University of Valparaíso, Chile) on these improvements of MO-ParamILS.

Evaluating our Dynamic Framework in Single-objective Optimization In Section 3.5, we proposed through an approach between AAC and parameter control, a dynamic framework able to modify the parameter of an algorithm during the run. A natural perspective would be to evaluate the interest of this approach for single-objective metaheuristics. Currently, the time splits are statically parameterized while the modification of the local structure of an instance (or class of instances) are not easily predictable. We may also extend this work and add more flexibility in the time splits in order to better react to the local structures of the search space.

Part II

Landscape-based Knowledge for Algorithm Design

Chapter 4

Neutrality in Multi-Objective Fitness Landscapes

In this chapter, we extend the concept of neutrality for multi-objective landscapes. During my PhD, we worked on single-objective fitness landscapes and more particularly, we studied and proposed some measures that characterize these specific landscapes. The ORKAD team is one of the specialists in multi-objective combinatorial optimization. When I came back in the team after my post-doc, we proposed a multi-objective definition to the neutrality. This work was part of a collaboration with Hernán Aguirre and Kiyoshi Tanaka of Shinshu University (Nagano, Japan) in 2015 and 2016.

The extension of neutrality to multi-objective optimization is not straightforward. In order to develop strategies to exploit neutral neighbors in multi-objective local search algorithms, it is important and necessary to clearly define neutrality in the multi-objective context. An initial work analyzing multi-objective stochastic local search algorithms (see Chapter 2 for a detailled definition) from the point of view of neutrality was presented in Blot et al. (2015), showing that these methods can be improved on bi-objective permutation flowshop scheduling problems by exploiting neutrality. The contributions of this chapter come from Marmion et al. (2016) and Kessaci-Marmion et al. (2017) where definitions and characterizations of the neutrality are given and analyzed on permutation problems.

This chapter is organized as follows:

- Section 4.1 explains the background of this work.
- Section 4.2 proposes a natural definition of neutrality based on Pareto-dominance, widely used in multi-objective optimization. Then, it gives definitions derived from epsilon and hypervolume indicators since such indicators are usually used to compare sets of solutions.
- Section 4.3 studies the distribution of neutral neighbors in permutation problems according to the three neutral definitions proposed.
- Section 4.4 characterizes some of the most promising neutral neighbors and presents some measures to understand the particularities they lead to.
- Section 4.5 studies the distribution of the most promising neutral neighbors in permutation problems where different correlations between the objectives appear.

4.1 Background

In single-objective optimization, the neutrality concept has been used to qualify a problem where different solutions have the same fitness. Formally, let s_1 and s_2 be two different solutions of the search space Ω. s_1 and s_2 are said to be neutral *iff* $f(s_1) = f(s_2)$. Stochastic local search (SLS) algorithms iteratively improve a solution by exploring its neighborhood. The neutral property has been defined between two neighboring solutions. Then, the set of the neutral neighbors of a solution s is $\mathcal{N}_n(s) = \{s' \in \mathcal{N}(s) | \ f(s') = f(s)\}$ where $\mathcal{N}(s)$ is the neighborhood of s. In single-objective optimization, it has been shown that SLS algorithms are sensitive to neutrality and other properties of the landscape and that properly exploiting neutrality can improve performance of local search methods (Marmion et al. 2011a; Vérel et al. 2004).
Contrary to the single objective case, there is not much work on neutrality in multi-objective optimization. A natural extension of the neutrality in multi-objective would be if two solutions have the same objective vectors. However, although two solutions can share the same objective value in single-objective optimization, it is extremely rare to find multi-objective problems where a non null number of solutions would share the same objective vectors. Indeed, multi-objective criteria are generally contradictory and not correlated, explaining that solutions could be equal for one objective but rarely on the others. The extension of neutrality to multi-objective optimization is then, not straightforward and its effects on the dynamics of multi-objective optimization algoithms are not clearly understood. An initial work analyzing multi-objective stochastic local search from the point of view of neutrality defined on the Pareto-dominance property was presented in Blot et al. (2015). This work shows that the classical algorithm can be improved on bi-objective permutation flowshop scheduling problems by exploiting this defined neutrality property.

4.2 Indicator-based Definitions of Neutrality

In the context of multi-objective optimization, given a solution $s \in \Omega$ to explore using a neighborhood structure $\mathcal{N}$, three sets of neighboring solutions s' of s may be defined: *improving* neighbors $\mathcal{I}(s)$ that improve the quality of all the objectives of the solution, *deteriorating* neighbors $\mathcal{D}(s)$ that decrease the quality of all the objectives of the solution, or *undetermined* neighbors $\mathcal{U}(s)$ that are neither improving ones nor deteriorating ones, as shown in Figure 4.1 for a bi-objective minimization problem. These latter neighbors are non-comparable solutions with the initial solution s.
Based on this partition of neighbors, the following will present several definitions of neutrality. We will consider for illustrative purposes a bi-objective minimization problem.

4.2.1 Neutrality based on Pareto Dominance

Pareto dominance is directly linked to the definition of the three partitions of the neighbors presented in the previous section (see Figure 4.1). Indeed, improving neighbors are usually called as *dominating* neighbors ($s' \succ s$ with $s' \in \mathcal{N}(s)$), deteriorating neighbors as *dominated* ($s \succ s'$) and undetermined neighbors are solutions that are *non-dominated*

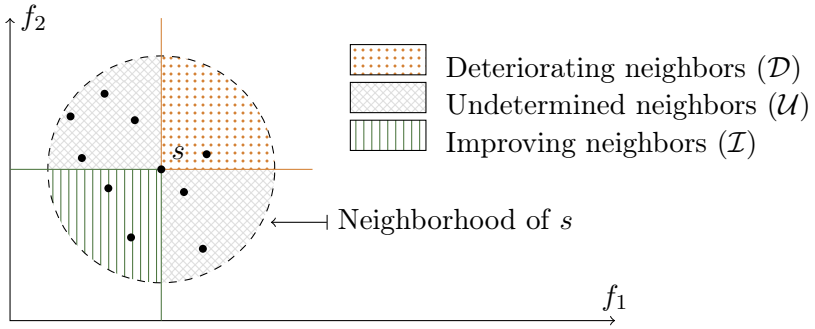


Figure 4.1: Neighbors types in a bi-objective minimization problem.

each other. Hence, neighbors in $\mathcal{U}$ can be either equal neighbors (equal for all the objectives) or viewed as *equivalent* neighbors. Naturally, they define the neutral neighbors in the multi-objective context when comparison of solutions is computed using Pareto dominance. Formally, Pareto-neutrality is defined as follows: given a solution $s \in \Omega$, ${\mathcal{N}_n}^P(s)$ is the set of Pareto-neutral neighbors,

$$ {\mathcal{N}_n}^P(s) = \{s' \in \mathcal{N}(s) \mid s \not\succ s' \texttt{ and } s' \not\succ s\} \tag{4.1} $$

This definition of neutrality has been efficiently exploited in a multi-objective local search (Blot et al. 2015) that shows the interest of clearly define the neutrality between neighbors. In the following, other definitions of neutrality based on the epsilon and the hypervolume difference indicators are proposed.

4.2.2 Neutrality based on ε-indicator

4.2.2.1 Partition of the neighborhood with ε-indicator

In the Introduction chapter, the binary additive epsilon indicator ε^+ (resp. multiplicative ε^*) was introduced to compare two Pareto set approximations. While comparing two solutions, a solution s and a neighbor solution s', this relation may be written as:

$$ I_{\varepsilon^+}(s', s) = max\{max_{\forall i}\{f_i(s') - f_i(s)\}; 0\} \tag{4.2} $$

$$ (\text{resp. } I_{\varepsilon^*}(s', s) = max\{max_{\forall i}\{f_i(s')/f_i(s)\}; 1\}) $$

This represents the minimum distance (resp. value) required for neighbor s' to dominate solution s. This definition divides the set of neighbors of a solution into two sets as depicted in Figure 4.2:

- If the distance equals 0 (resp. value equals 1), the neighbor s' already dominates solution s. The set of these neighbors characterizes $\mathcal{I}$ (green area of Figure 4.2). The definition of $\mathcal{I}$ may be written as follows:

$$ \mathcal{I}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^+}(s', s) = 0\} \tag{4.3} $$

$$(\text{resp. } \mathcal{I}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^*}(s', s) = 1\})$$

- If the distance is positive (resp. value greater than 1), the neighbor may be either a deteriorating or an undetermined solution. The set of these neighbors characterizes $\mathcal{D} \cup \mathcal{U}$ (orange area of Figure 4.2). The definition of $\mathcal{D} \cup \mathcal{U}$ may be written as follows:

$$\mathcal{D} \cup \mathcal{U}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^+}(s', s) > 0\} \tag{4.4}$$

$$(\text{resp. } \mathcal{D} \cup \mathcal{U}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^*}(s', s) > 1\})$$

Both indicators I_{ε^+} and I_{ε^*} have very similar behaviors except the special role of values 0 and 1.

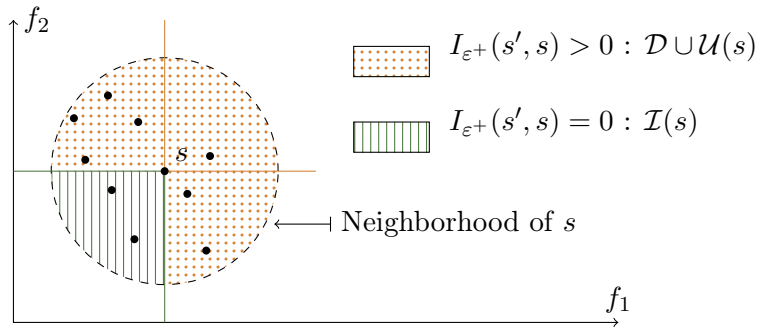


Figure 4.2: Value of $I_\varepsilon(s', s)$ in a bi-objective minimization problem.

4.2.2.2 A symmetric definition of ε-neutrality

Neighbors belonging to $\mathcal{D} \cup \mathcal{U}$ (orange area of Figure 4.2) may be separated into two types of neighbors with a symmetry consideration, i.e. if s' is neighbor of s then s is also neighbor of s'. Indeed, considering a symmetric relation between s' and s, the ε-indicator may take two values:

- $I_{\varepsilon^+}(s, s') = 0$ (resp. $I_{\varepsilon^*}(s, s') = 1$) means that s dominates s', i.e. $s' \in \mathcal{D}(s)$
- $I_{\varepsilon^+}(s, s') > 0$ (resp. $I_{\varepsilon^*}(s, s') > 1$) means that s does not dominate s', i.e. $s' \in \mathcal{U}(s)$

These latter solutions may be considered as neutral neighbors and leads to the proposition of a first definition of neutrality based on the ε-indicator. With this definition, the undetermined set exactly characterizes the set of neutral neighbors $\mathcal{U} = \mathcal{N}_n^{\varepsilon}$.
Formally:

$$\mathcal{N}_n^{\varepsilon^+}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^+}(s', s) > 0 \text{ and } I_{\varepsilon^+}(s, s') > 0\} \tag{4.5}$$

$$(\text{resp. } \mathcal{N}_n^{\varepsilon^*}(s) = \{s' \in \mathcal{N}(s) \mid I_{\varepsilon^*}(s', s) > 1 \text{ and } I_{\varepsilon^*}(s, s') > 1\})$$

Let us remark that this symmetric definition of ε-neutrality characterizes the same set of neutral neighbors as Pareto-neutrality. However, this indicator produces a value that may be used to more precisely describe the relations between neighboring solutions. Indeed, introducing a bound on the indicator value will lead to a restriction of the set of neutral neighbors. This restriction may be controlled by this tunable bound. Hereafter, we propose neutrality definitions based on a bounded indicator value.

4.2.2.3 Bounded definitions of ε-neutrality

Another way to define neutral neighbors is to consider as neutral those neighboring solutions that are close (in objective space) to the initial solution s. Hence, using the ε^+-indicator (resp. ε^*-indicator), we could define a neighbor as neutral if the distance (resp. value) necessary to make the neighbor dominate the initial solution s is smaller than a threshold δ. As explained latter, normalized values are then required in order to give to δ a signification independent of the objective.
First, let us consider the set of neighbors s' in $\mathcal{D} \cup \mathcal{U}(s)$ defined from the ε- indicator without symmetry consideration i.e. $I_{\varepsilon^+}(s', s) > 0$ (resp. $I_{\varepsilon^*}(s', s) > 1$). The set E_0 of neutral neighbors is then defined from these neighbors verifying the distance (resp. value) is bounded by δ. Figure 4.3 (top) illustrates the area of E_0-neutral neighbors of a solution delimited by δ in bi-objective optimization.
Formally:

$$E_0{}^+(s) = \{s' \in \mathcal{N}(s) \mid 0 < I_{\varepsilon^+}(s', s) < \delta\} \tag{4.6}$$

$$(\text{resp. } E_0{}^*(s) = \{s' \in \mathcal{N}(s) \mid 1 < I_{\varepsilon^*}(s', s) < \delta\})$$

Equation 4.6 considers as neutral some neighbors that are dominated by s since the neighbors belongs to $\mathcal{D} \cup \mathcal{U}(s)$. Removing these ones, only neighbors of $\mathcal{U}(s)$ are kept. In the following definitions, the symmetry consideration is taken into account and two definitions can be proposed. The set E_1 of neutral neighbors is a subset of E_0, where the dominated neighbors have been removed. Figure 4.3 (middle) illustrates the area of E_1-neutral neighbors of a solution delimited by δ in a bi-objective minimization problem.
Formally:

$$E_1{}^+(s) = \{s' \in \mathcal{N}(s) \mid 0 < I_\varepsilon^+(s', s) < \delta \texttt{ and } 0 < I_\varepsilon^+(s, s')\} \tag{4.7}$$

$$(\text{resp. } E_1{}^*(s) = \{s' \in \mathcal{N}(s) \mid 1 < I_\varepsilon^*(s', s) < \delta \texttt{ and } 1 < I_\varepsilon^*(s, s')\})$$

However, this definition is not fully symmetric. Indeed, a solution s' can be a neutral neighbor of s, while s is not a neutral neighbor of s'. This may be contradictory with the reciprocity notion. Thus we define the set E_2 of neutral neighbors as a subset of E_1, where the symmetric value of the ε-indicator is also bounded by δ. Figure 4.3 (bottom) illustrates the area of E_2-neutral neighbors of a solution delimited by δ in a bi-objective minimization problem.
Formally:

$$E_2{}^+(s) = \{s' \in \mathcal{N}(s) \mid 0 < I_\varepsilon^+(s', s) < \delta \texttt{ and } 0 < I_\varepsilon^+(s, s') < \delta\} \tag{4.8}$$

$$(\text{resp. } E_2{}^*(s) = \{s' \in \mathcal{N}(s) \mid 1 < I_\varepsilon^*(s', s) < \delta \texttt{ and } 1 < I_\varepsilon^*(s, s') < \delta\})$$

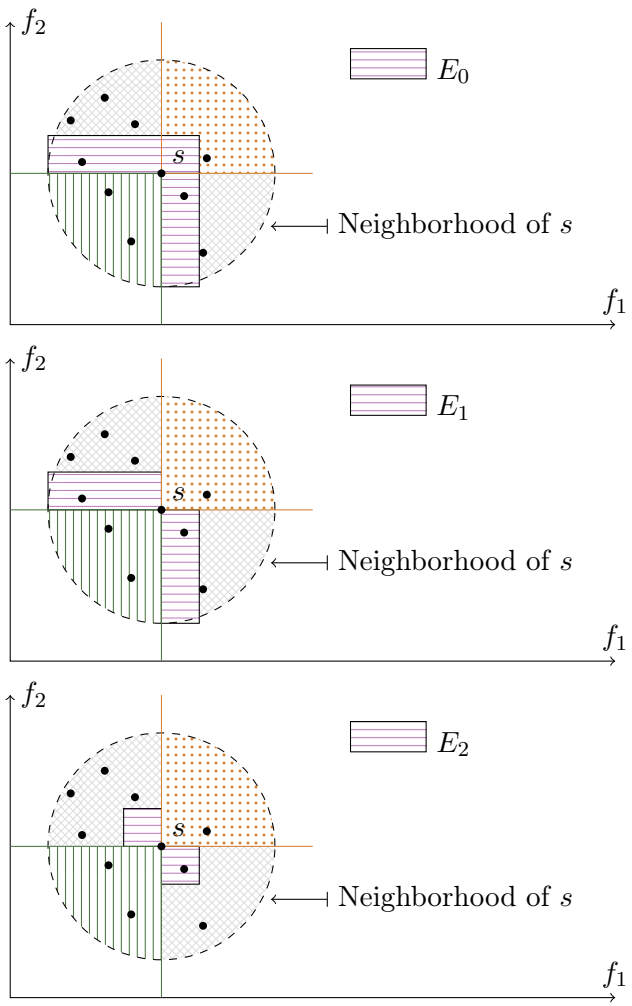


Figure 4.3: Bounded definitions of ε-neutrality in a bi-objective minimization problem.

As seen on Figure 4.3 the different sets of neutral neighbors given by Eq 4.6, 4.7 and 4.8 are represented by rectangular areas that become increasingly smaller conform bounded constraints are added on the initial set $\mathcal{D} \cup \mathcal{U}$. The width of the areas is tuned by δ. Moreover, having such different definitions of neutrality induces the following questions: Which are the most interesting neutral neighbors? Which definition should we use? For example, considering sets E_1 and E_2 (i.e. Equation 4.7 and 4.8), a neutral neighbor improves at least one objective while degrading at least a second one. If the degradation is smaller than the improvement (let us remember that values are normalized, and so comparable), this neighbor may be interesting to explore later. Such neighboring solutions belong to $E_1 \setminus E_2$, i.e. the degradation on one objective is limited by δ while the improvement of other ones is over δ and then, may be large even if δ is small. One of the important question is: Do such neighbors exist? Are they numerous?

4.2.3 Neutrality based on *HD*-indicator

4.2.3.1 Partition of the neighborhood with *HD*-indicator

Another way to compare two solutions, and in particular a solution s and one of its neighboring solutions s', is to calculate the "additional space" dominated by the neighboring solution. This is computed by the Hypervolume Difference $I_{HD}(s', s)$ between s' and s as follows:

$$I_{HD}(s', s) = I_H(s + s') - I_H(s)$$

Let us remark that $I_H(s + s')$ is the hypervolume covered by both points $\{s, s'\}$. This definition of *HD*-indicator divides the set of neighbors of a solution into two sets as depicted in Figure 4.4:

- If the value is null, the neighbor s' is dominated by the solution s. The set of these neighbors characterizes $\mathcal{D}$ (orange area of Figure 4.4). The definition of $\mathcal{D}$ may be written as follows:
$$\mathcal{D}(s) = \{s' \in \mathcal{N}(s) \mid I_{HD}(s', s) = 0\} \tag{4.9}$$
- If the value is positive, the solution s does not dominate the neighbor s'. The set of these neighbors characterizes $\mathcal{I} \cup \mathcal{U}$ (green area of Figure 4.4). The definition of $\mathcal{I} \cup \mathcal{U}$ may be written as follows:
$$\mathcal{I} \cup \mathcal{U}(s) = \{s' \in \mathcal{N}(s) \mid I_{HD}(s', s) > 0\} \tag{4.10}$$

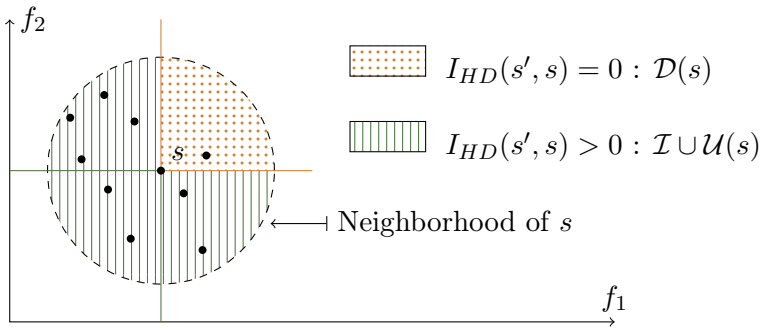


Figure 4.4: Value of $I_{HD}(s', s)$ in a bi-objective minimization problem.

4.2.3.2 A symmetric definition of *HD*-neutrality

Neighbors belonging to $\mathcal{I} \cup \mathcal{U}$ (green area of Figure 4.4) may be separated into two types of neighbors with a symmetry consideration. Indeed, regarding the symmetric relation between s' and s, the *HD*-indicator may take two values:

- $I_{HD}(s, s') = 0$ means that s' dominates s, i.e. $s' \in \mathcal{I}(s)$
- $I_{HD}(s, s') > 0$ means that s does not dominate s', i.e. $s' \in \mathcal{U}(s)$)

These latter may be considered as neutral neighbors and leads to the proposition of a first definition of neutrality based on *HD*-indicator. With this definition, the undetermined set exactly characterizes the set of neutral neighbors $\mathcal{U} = {\mathcal{N}_n}^{HD}$.
Formally:

$$ {\mathcal{N}_n}^{HD}(s) = \{s' \in \mathcal{N}(s) \mid I_{HD}(s', s) > 0 \texttt{ and } I_{HD}(s, s') > 0\} \tag{4.11} $$

Let us remark that this symmetric definition of *HD*-neutrality characterizes the same set of neutral neighbors as Pareto-neutrality. As for the ε-indicator, the *HD*-indicator computes a value that may be exploited to give some more precise definitions through the introduction of a bound.

4.2.3.3 Bounded definitions

Once again, in order to compare improvement/degradation of the objective functions, a normalization of values should be done. A normalization method will be clarified in the experimental protocol.
In order to give a similar meaning to the bound δ for both ε- and *HD*- indicators, we define $\widehat{I}_{HD}(s, s') = 1 - I_{HD}(s, s')$, where $I_{HD}(s, s')$ is a (positive) normalized value lower than 1. Here, we consider the symmetric definition only since improving neighbors cannot be neutral neighbors. The set HD_1 of neutral neighbors s' is then defined as a subset of $\mathcal{U}(s)$ where $\widehat{I}_{HD}(s', s)$ is bounded by δ. Figure 4.5 (top) illustrates the area of HD_1-neutral neighbors of a solution delimited by δ in a bi-objective minimization problem.
Formally:

$$ HD_1(s) = \{s' \in \mathcal{N}(s) \mid \widehat{I}_{HD}(s', s) \leq \delta \texttt{ and } \widehat{I}_{HD}(s, s') > 0\} \tag{4.12} $$

In the same way as ε-neutrality (Equation 4.7 and 4.8), it is important to propose a fully symmetric definition to get the reciprocity notion. The set HD_2 of neutral neighbors is a subset of HD_1, where the symmetric value of the *HD*-indicator is also bounded by δ. Figure 4.5 (middle) illustrates the area of HD_2-neutral neighbors of a solution delimited by δ in a bi-objective minimization problem.
Formally:

$$ HD_2(s) = \{s' \in \mathcal{N}(s) \mid \widehat{I}_{HD}(s', s) \leq \delta \texttt{ and } \widehat{I}_{HD}(s, s') \leq \delta\} \tag{4.13} $$

Let us note that, δ represents an upper bound that delimits an area to define a set of neutral neighbors. In Equation 4.12 and 4.13, $\widehat{I}_{HD}(s', s) \leq \delta$ can be replaced by $I_{HD}(s', s) \geq \delta$ to get the adaptation of these definitions if the similar meaning with ε-indicator is not needed.
As seen on Figure 4.5 (top and middle) the different sets of neutral neighbors given by Equation 4.12 and 4.13 are represented by convex areas that become increasingly smaller conform bounded constraints are added on the initial set $\mathcal{U}$ (i.e. δ is close to zero). Let us remark that the convex line from s represents points of same hypervolume value $I_{HD}(s', s)$ i.e. each point which covers an equivalent area ($I_H(s + s')$) in the objective space. The width of the convex areas is tuned by δ. Figure 4.5 (bottom) gives a zoom on a part of HD_2-neutral neighbors. Two cases are illustrated: the first one with a small value of δ and the other one with a high value. If δ is very small, the convex area is very far from the initial solution. Convex areas are less predictable than rectangular ones and then the

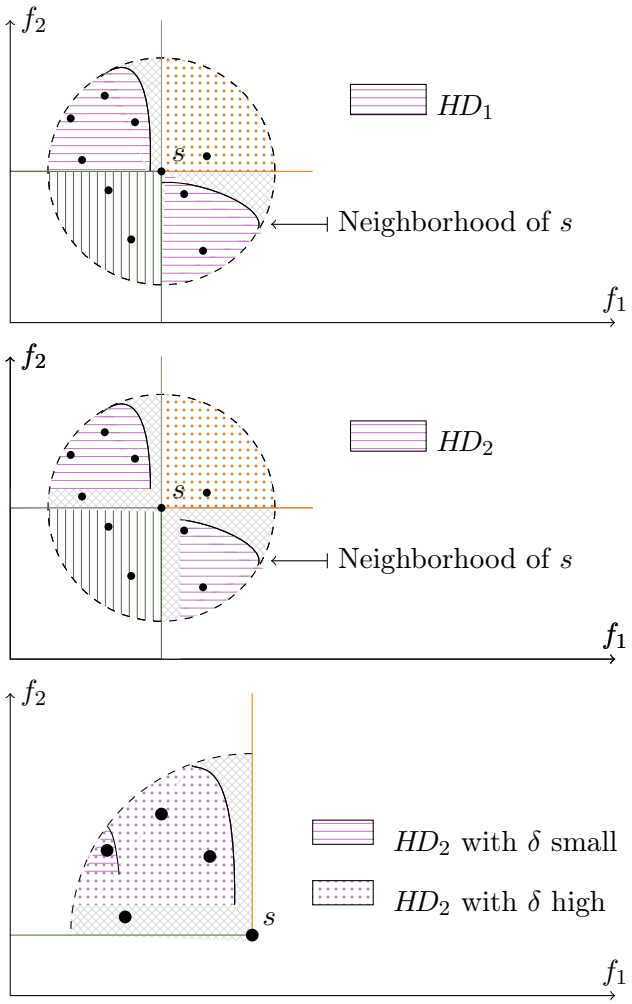


Figure 4.5: Bounded definitions of *HD*-neutrality in a bi-objective minimization problem.

impact of the choice of the δ value is not easy to anticipate. Moreover, the interest of the definitions may also be discussed. Neighboring solutions belonging to $HD_1 \setminus HD_2$ seem to represent a good trade-off between the degradation on one objective and the improvement of other ones. Let us remark that these latter neighbors s' of s are in the space where $\widehat{I}_{HD}(s', s)$ value is small and so for $I_\varepsilon(s', s)$. Then, both values have to be minimized that strengthens the decision of reversing I_{HD} over 1.

4.3 Distribution of Neutral Neighbors in Permutation Problems

We analyze the previously presented definitions of neutrality for two multi-objective permutation problems: the permutation flowshop scheduling problem and the traveling sales-

man problem. Here, only the 2- and 3-objective versions of these problems are considered. Therefore, we present some measures that will enable to better characterize the fitness landscape of these permutation problems.

4.3.1 Experimental Protocol

Problems Description

The permutation flowshop scheduling problem (FSP) has already been introduced in Section 3.2.4.1. In the following experiments, we consider three objectives to be minimized, being (in this order) the makespan (Equation 3.3 in Section 3.2.4), the total tardiness (Equation 4.14) and the sum of flowtimes (Equation 3.4).

$$f_2(\pi) = TT = \sum_{i=1}^{N} \{ \max \ \{0, \ C_i - d_i\}\} \tag{4.14}$$

The neighborhood operator considered is the insertion operator and, Minella instances (already presented in Section 1.3.3) with a number of jobs $N \in \{20, 50, 100\}$ and 20 machines are analyzed.

The traveling salesman problem (TSP) has also already been introduced in Section 3.2.4.2. Problems with two are three independent objectives are investigated. The neighborhood operator is the two-opt operator. TSP instances were generated with the generator provided in the 8^{th} DIMACS challenge [1] with the same problem sizes than those considered for the FSP i.e. $N \in \{20, 50, 100\}$ has been analyzed.

Indicators Computation and Normalization

As mentioned before, both additive and multiplicative epsilon indicators can be identically used and so, in the following, only the additive version is considered. Here, we describe how to normalize the computed ε- and *HD*-values. First, considering one solution s, all of its neighbors $s' \in \mathcal{N}(s)$ have to be exhaustively evaluated. Then, for each indicator, a reference point is defined in order to compute the normalized values. About ε-indicator, the reference point is s and for each objective $i \in K$, $\texttt{ref}_i = max(f_i(s'))$ as the maximum value found among the neighbors per objective and $\varepsilon_i = \dfrac{f_i(s') - f_i(s)}{\texttt{ref}_i - f_i(s)}$ where $max_{f_i} = max(f_i(s'))$ is used for the normalization. Then, $\varepsilon = max_i(\varepsilon_i)$. About *HD*-indicator, the reference point Z is defined as $z_i = max(f_i) + 1$ and the *HD*-values are computed with Pisa (Brockhoff 2015). The normalization is made using $min(f_i)$ of each objective i.

Fitness Landscape Measures

The structure of a problem, and in particular the neutrality property can vary between regions of the search space. Hence, we propose to analyze the distribution of neutral neighbors according to the several definitions of neutrality for both *random* and *Pareto*

[1] `http://dimacs.rutgers.edu/Challenges/TSP/`

local optimum (PLO) solutions. In order to statistically ensure this study, for each instance, two populations $\mathcal{P}_{rnd}$ and $\mathcal{P}_{opt}$ of 30 random and PLO solutions are respectively considered. Then, for all neighbors of each solution in $\mathcal{P}_{rnd}$ and $\mathcal{P}_{opt}$ ε- and *HD*- values are computed.
In Section 4.2, three initial sets $(\mathcal{I}, \mathcal{D}, \mathcal{U})$ have been defined to partition the neighbors of a solution. According to the first definitions of Pareto-, ε- and *HD*-neutrality, $\mathcal{U}(s)$ is the set of neutral neighbors of a solution s. Both definitions of ε- and *HD*- indicator divide the neighborhood into two sets $\mathcal{D} \cup \mathcal{U}(s)$ and $\mathcal{I} \cup \mathcal{U}(s)$, respectively. Let us remark that, for a PLO solution, $\mathcal{I}$ is an empty set. Let us introduce, the *neutral degree* of a solution as the number of neutral solutions in its neighborhood (Marmion et al. 2011b). To compare problems with different neighborhood size, we compute the neutral ratio as the neutral degree divided by the neighborhood size. In order to analyze the impact of the definitions of the neutrality on the distribution of neighbors in the decision space, the sets E_0, E_1, E_2, HD_1 and HD_2 are determined. The cardinality of these sets gives the number of neutral neighbors included within the considered bounded definition.

4.3.2 Neighbors Distribution in Initial Partitions

Table 4.1 gives the average distribution of improving ($\mathcal{I}$) and deteriorating ($\mathcal{D}$) neighbors together with different partition of neighbors ($\mathcal{U}$, $\mathcal{I} \cup \mathcal{U}$, $\mathcal{D} \cup \mathcal{U}$) for both random and Pareto Local Optimum (PLO) solutions. At first glance, whatever the definition of neutrality and the type of solutions, TSP neutrality is higher than FSP one. Moreover, the neutral ratio of random solutions is always higher than the one of PLO solutions for both TSP and FSP problems. Let us remark that the trend is the same for problems with 2 or 3 objectives. First, for PLO solutions, the neutral ratio decreases when the size of the problem (N) increases. For FSP, the neutral ratio is almost null for more than 50 jobs. Secondly, the Pareto-neutrality does not vary in the same way between FSP and TSP for random solutions. Indeed, the number of neutral neighbors decreases while the numbers of improving or deteriorating neighbors are almost the same and increase with the size of the FSP problem. It seems that potential neutral neighbors are equally distributed between improving or deteriorating when the size increases. Hence, both ε-partition (see $\mathcal{D} \cup \mathcal{U}$) and *HD*-partition (see $\mathcal{I} \cup \mathcal{U}$) are invariant according to the size. For TSP, 50% of neighbors are Pareto-neutral no matter the size and the other neighbors are equally distributed into improving and deteriorating ones. Therefore, ε-partition and *HD*-partition leads to the same neutral ratio when no bound is used. Finally, neutrality is almost the same between 2 and 3 objectives for FSP while it increases significantly for TSP, where more than 75% of neighbors of random solutions are neutral. These two permutation-based problems present some differences between neutral features. Knowing that FSP objectives are dependent, contrary to TSP ones, may explain in part these differences.

4.3.3 Neutral Neighbors Distribution according to the Bound

The analysis focus on the distribution of neutral neighbors in the space delimited by the bound δ. First, δ is equal to the average of the indicator values of neutral neighbors. Table 4.2 gives the average cardinality (in percent) of the sets E_0, E_1 and E_2 defined from ε-indicator and HD_1 and HD_2 defined from *HD*-indicator. Note that $|E_0|$ is computed as

Table 4.1: Cardinality (in percent) in different sets of neighbors. s is in $\mathcal{P}_{rnd}$ or $\mathcal{P}_{opt}$, Results are given for both FSP and TSP instances with 2 and 3 objective criteria.

	$\mathcal{P}_{rnd}$					$\mathcal{P}_{opt}$	
2o	$\mathcal{I}(s)$	$\mathcal{D}(s)$	$\mathcal{U}(s)$	$\mathcal{I} \cup \mathcal{U}(s)$	$\mathcal{D} \cup \mathcal{U}(s)$	$\mathcal{D}(s)$	$\mathcal{U}(s)$
FSP							
20	31.05	32.66	36.29	67.34	68.95	87.88	12.12
50	37.22	36.90	25.88	63.10	62.78	96.88	3.12
100	37.65	39.52	22.83	60.48	62.35	98.97	1.03
TSP							
20	24.58	24.66	50.76	75.34	75.42	70.10	29.90
50	24.78	25.16	50.06	74.84	75.22	78.88	21.12
100	25.18	24.75	50.07	75.25	74.82	83.80	16.20
3o	$\mathcal{I}(s)$	$\mathcal{D}(s)$	$\mathcal{U}(s)$	$\mathcal{I} \cup \mathcal{U}(s)$	$\mathcal{D} \cup \mathcal{U}(s)$	$\mathcal{D}(s)$	$\mathcal{U}(s)$
FSP							
20	25.58	26.86	47.56	73.14	74.42	83.10	16.90
50	33.86	33.52	32.62	66.48	66.14	95.23	4.77
100	36.49	38.33	25.18	61.67	63.51	98.81	1.19
TSP							
20	12.14	11.80	76.06	88.20	87.86	41.77	58.23
50	12.42	12.49	75.08	87.51	87.58	49.46	50.54
100	12.49	12.34	75.17	87.66	87.51	55.35	44.65

a percentage of the total number of deteriorating and undetermined neighbors ($\mathcal{D} \cup \mathcal{U}$) while the other ones are computed only with undetermined neighbors ($\mathcal{U}$). It explains that $|E_0|$ and $|E_1|$ are quite similar. First, the number of objectives seems not to impact the distribution of neutral neighbors since the trends and the cardinality are quite similar. Secondly, despite the fact that TSP neutral ratio is much higher than FSP one, the cardinality and the trends between the sets are close, except one of E_2 of PLO solutions. Then, we will start to analyze the similarities that might be generalized for permutation problems. $|E_0|$ and $|E_1|$ are almost equal for both random and PLO solutions. Even if the bound δ is not the same, it seems that the distribution of neighbors is equivalent between the sets $\mathcal{D}$ and $\mathcal{U}$ in terms of ε-indicator values. Concerning random solutions, E_2 contains more than half of E_1. That is, more than half of the neutral neighbors of a solution are close to it in the objective space. These neighbors can not be useful for diversification strategies. However, the remaining ones ($s' \in E_1 \setminus E_2$) could be interesting solutions to explore. Moreover, HD_2 contains less than half of HD_1. That is, more than half of the neutral neighbors of a solution are close to the axis, i.e. the deterioration on a criterion is weak. The set $HD_1 \setminus HD_2$ contains many solutions that have to be carefully considered. The distribution of neutral neighbors of PLO solutions is very different from the one of random solutions. Indeed, although $|E_1|$ and $|HD_1|$ are almost similar between random and PLO solutions, $|E_2|$ and $|HD_2|$ is totally different. All the neighbors in E_1 and HD_1 are also in E_2 and HD_2 respectively. It means that most of the neutral neighbors, where

Table 4.2: Cardinality (in percent) of E_0, E_1, E_2, HD_1 and HD_2 with δ set to the average value. Results are given for both FSP and TSP instances with 2 and 3 objective criteria. Considered solutions are in $\mathcal{P}_{rnd}$ or $\mathcal{P}_{opt}$.

	$\mathcal{P}_{rnd}$					$\mathcal{P}_{opt}$				
2o	E_0	E_1	E_2	HD_1	HD_2	E_0	E_1	E_2	HD_1	HD_2
FSP										
20	58.35	59.94	35.54	38.44	14.75	58.69	58.88	56.44	42.55	40.31
50	58.86	61.13	35.86	37.90	12.98	59.47	60.68	59.09	36.50	33.63
100	58.37	61.88	36.35	37.97	12.57	58.52	64.74	62.54	35.53	31.37
TSP										
20	57.66	59.46	33.97	37.38	12.78	51.03	50.98	49.68	34.66	32.35
50	58.42	60.33	36.57	37.96	14.49	50.92	52.21	51.92	34.95	33.08
100	58.54	60.47	36.36	38.20	14.44	50.54	51.76	51.69	35.46	34.20
3o	E_0	E_1	E_2	HD_1	HD_2	E_0	E_1	E_2	HD_1	HD_2
FSP										
20	58.37	60.03	34.35	35.04	7.87	58.47	58.82	29.31	35.87	33.39
50	59.07	61.01	34.87	35.84	8.56	60.03	60.99	31.79	35.37	33.12
100	58.38	61.79	36.01	36.33	10.17	58.87	64.65	31.97	35.38	33.20
TSP										
20	55.99	56.57	29.31	36.30	9.76	51.42	51.54	48.62	33.51	31.62
50	56.78	57.60	31.79	37.06	10.63	51.33	51.53	50.35	34.08	32.41
100	56.97	57.79	31.97	37.67	10.99	51.03	51.24	50.79	34.54	33.56

the indicator value is lower than the average, are distributed very close to the solution. Let us consider a neutral neighbor that improves a lot one objective criterion with a weak deterioration of the other ones. The results show that random solutions have a non trivial number of this last type of neutral neighbors contrary to PLO solutions. Let us return to the case of E_2 for FSP with 3 objectives. Unlike the other cases, only half of the neighbors in E_1 are in E_2. It can be explained by the high dependency between objective 1 ($C_{\max}$) and objective 3 (SFT) since a weak improving of one can impact significantly the second.

In the following, the distribution of neutral neighbors is analyzed more finely with different settings of the bound δ. Figure 4.6 gives the distribution of neutral neighbors in percent (compared to the whole neighborhood) in the sets E_0, E_1, E_2, HD_1 and HD_2 following different values of δ. Note that, $\delta = 100$ means that all the neutral neighbors (as defined by each set) are considered. Then, $|E_0| = |\mathcal{D} \cup \mathcal{U}|$ and $|E_1| = |HD_1| = |\mathcal{U}|$. First, the trends between the different sets that have been analyzed with Table 4.2 are obvious. Here, the average value is meaningful. Therefore, in this part, we only discuss about new information given by these figures.

The ε- indicator defines rectangular areas around the solution in the objective space. These areas become larger when δ increases. Their increase is linear with δ, hence, the intervals are linearly divided. The HD- indicator defines different convex areas that become larger also when δ increases. The increase of these areas is quadratic with δ, hence, the

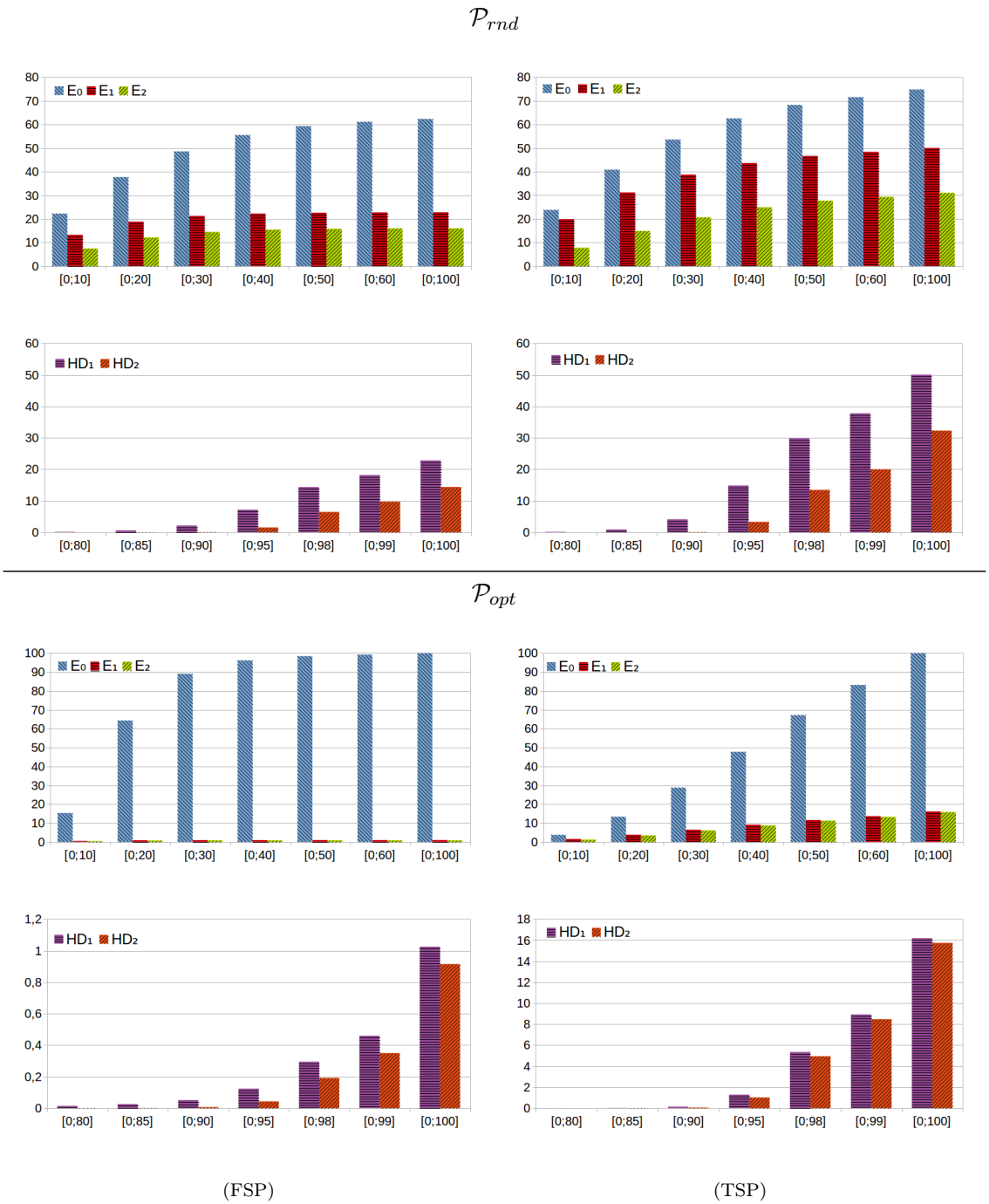


Figure 4.6: Cardinality (in percent) of E_0, E_1, E_2, HD_1 and HD_2 for different values of δ for both bi-objective FSP (left side) and TSP (right side). Here, results are given for the problem of size N=100. Considered solutions are in $\mathcal{P}_{rnd}$ or $\mathcal{P}_{opt}$.

intervals are divided with a logarithmic scale approach. The main conclusion is that the opposite behavior of both indicators is highlighted. Indeed, the cardinality of E_0, E_1 and E_2 logarithmically increases, while the cardinality of HD_1 and HD_2 increases almost at the end. The definition of the indicator impacts significantly the definition of the neutrality and consequently the determination of which neighbors are considered as neutral. Moreover, we observe that TSP is highly neutral compared to FSP. This may be due to the type of the instances. Indeed, in this TSP instances, the objectives are independent as the distance-matrices are; while for the FSP, the 2 or 3 objectives have some correlations.

4.4 Characterization of Promising Neutral Neighbors

In single-objective optimization, considering neutral neighbors during the search has already been successful in the past (Galván-López et al. 2011; Marmion et al. 2011a). However, the experiments presented in the previous section show that such neutral neighbors may be very numerous and, considering all of them, almost at the beginning of the search, can be very costly (for the archiving task for example). In this section, we will investigate if some of these neutral neighbors can be considered as most promising (leading to other parts of the search space, giving more diversity, contribution greater than the storing cost...).
Moreover, the previous experiments left open questions about the link between the neutral degree and the correlation of the objectives. In particular the following questions will be addressed in this section:

- How to define promising neutral neighbors?
- Are these promising neutral neighbors numerous? Is it dependent on the correlation between objectives?
- Where are they located? What is their distribution in the neighborhood?

4.4.1 Definition of Promising Neutral Neighbors

Defining promising neighbors seems to be problem-dependent. However, among properties that are important to consider, many will agree to say that a neutral neighbor could be promising if, for example:

- it allows to increase the spread along the Pareto front,
- it improves the convergence,
- it improves the hypervolume measure.

We propose to summarize these properties by defining promising neutral neighbors as solutions that bring to a deterioration on one objective function smaller than the improvement they bring on the other one (see Figure 4.7(top)). We consider now, the additive epsilon

indicator and the (symmetric) definition of neutrality given by Equation 4.5. Let us note $\mathcal{P}_n(s) \subset \mathcal{N}_n(s)$, the set of promising neutral neighbors. It is defined as follows:

$$\mathcal{P}_n(s) = \{s' \in \mathcal{N}_n(s) \mid I_\varepsilon(s', s) < I_\varepsilon(s, s')\} \tag{4.15}$$

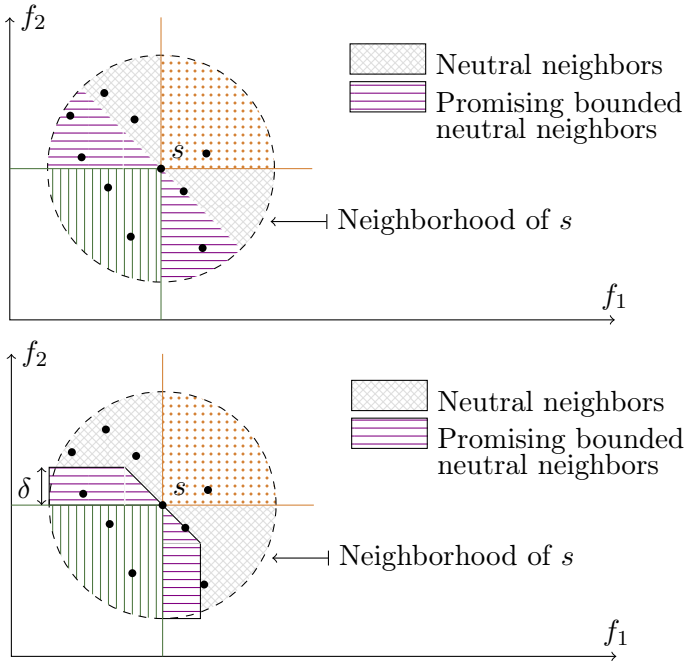


Figure 4.7: Promising neutral neighbors: whole area (top) and δ-bounded area (bottom).

Within those promising neighbors, it is also possible to control the deterioration between s and its neighbors s', represented by the value of $I_\varepsilon(s', s)$ with a bound $\delta \geq 0$ (see Figure 4.7(bottom)). This leads to the definition of $\mathcal{P}_n^\delta(s) \subset \mathcal{P}_n(s)$, the set of promising neutral neighbors depending on δ, that may be defined by:

$$\mathcal{P}_n^\delta(s) = \{s' \in \mathcal{P}_n(s) \mid I_\varepsilon(s', s) \leq \delta\} \tag{4.16}$$

4.4.2 Methodology to Study the Distribution of the Promising Neutral Neighbors

When promising neutral neighbors are numerous in comparison to the number of neutral neighbors and more widely to the whole neighborhood, a deeper study of their location in the search space would help to exploit them into strategies. Therefore, we propose to consider several values for the bound δ to study their distribution among the neutral neighbors. Let us remark that values of objective functions of a solution s and its neighbors s' are first normalized using the whole neighborhood $\mathcal{N}$, so that $I_\varepsilon(s', s) \in [0; 1]$. Let $\Delta = \{\delta_0, \delta_1, \delta_2 \dots \delta_D\}$ a discretization of $[0; 1]$ where $\delta_0 = 0$ and $\delta_D = 1$.
If we focus on the left-upper side quarter of the neighborhood, different values of δ lead to horizontal layers ($\mathcal{H}$-layer) of the promising neighbors area (similar layers may be found

in the other part dealing with neutral neighbors). Such a $\mathcal{H}$-layer allows to study the distribution of neutral neighbors regarding a bounded authorized degradation (see Figure 4.8 (left)). This may be formalized by:

$$\mathcal{H}_n^{\delta_i} = \{s' \in \mathcal{P}_n^{\delta_i}(s) \mid \delta_{i-1} < I_\varepsilon(s', s),\ 1 \leq i \leq D\} \tag{4.17}$$

Let us remark that the first $\mathcal{H}$-layer ($\mathcal{H}_n^{\delta_1}$) is the closest to the dominance area and hence neighbors belonging to it represent a great interest.

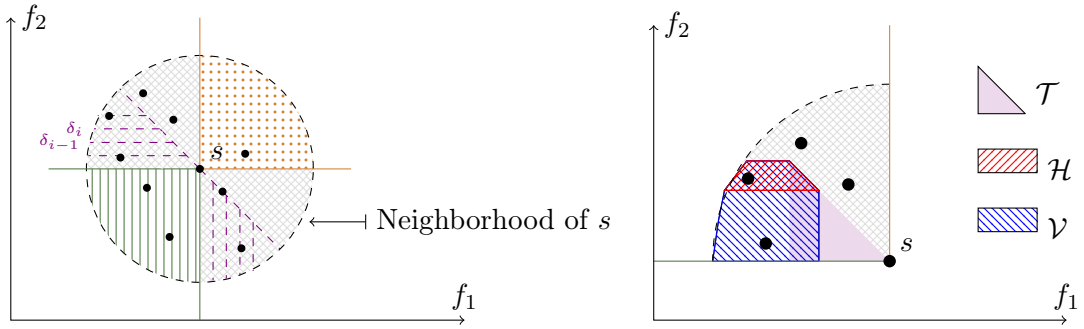


Figure 4.8: Space decomposition

In addition, we propose to define two other sub-areas. First, triangles $\mathcal{T}_n^{\delta_i}$, in which both $I_\varepsilon(s, s')$ and $I_\varepsilon(s', s)$ are bounded by δ_i:

$$\mathcal{T}_n^{\delta_i} = \{s' \in \mathcal{P}_n^{\delta_i}(s) \mid I_\varepsilon(s, s') \leq \delta_i,\ 0 \leq i \leq D\} \tag{4.18}$$

Secondly, vertical layers ($\mathcal{V}$-layer), denoted as $\mathcal{V}_n^{\delta_i}$, to group neighbors further and further from the solution explored s (see Figure 4.8 (right)).

$$\mathcal{V}_n^{\delta_i} = (\sum_{k=\delta_0}^{\delta_i} \mathcal{H}_n^k) \setminus \mathcal{T}_n^{\delta_{i-1}} = \{s' \in \mathcal{P}_n^{\delta_i}(s) \mid I_\varepsilon(s, s') > \delta_{i-1},\ 1 \leq i \leq D\} \tag{4.19}$$

The $\mathcal{V}$-layers allow to study solutions that produce a minimal improvement (δ_{i-1}) with a bounded degradation (δ_i). Neighboring solutions belonging to ${\mathcal{V}_n}^{\delta_D}$ produce an improvement larger than the worst degradation encountered within the whole neighborhood. Let us remark that $\mathcal{V}_n^{\delta_1} == \mathcal{H}_n^{\delta_1} == \mathcal{P}_n^{\delta_1}$. Focusing on the first $\mathcal{H}$-layer, $\mathcal{H}_n^{\delta_1}$, neighbors belonging to this layer are either in the triangle $\mathcal{T}_n^{\delta_1}$ or in the rest of the layer $\mathcal{P}_n^{\delta_1} \setminus \mathcal{T}_n^{\delta_1}$. The triangle groups solutions very close to the initial solution s; these solutions have a very low interest regarding convergence and diversity. The rest of the layer groups solutions that, on the contrary, produce on one objective an improvement larger than the worst possible degradation for the second objective in this layer (i.e. δ_1).

4.5 Distribution of Promising Neutral Neighbors in Permutation Problems

In this section, we analyze where are located the promising neighbors in the neighborhood of bi-objective optimization problems. However, the correlation between the objectives largely influences the distribution of the whole neighborhood. Hence, we propose to study the distribution of the promising neighbors among the neutral ones and more largely, among the whole neighborhood, according to the correlation between objectives. We focus our study on bi-objective permutation problems since they are widely used in the literature to validate new optimization methods. Therefore, knowing the structure of these permutation problems may help in the design of new methods. The permutation problems considered are: the *permutation flowshop scheduling problem* (see Section 3.2.4.1), the *quadratic assignment problem* (see Section 3.2.4.3) and the *traveling salesman problem* (see Section 3.2.4.2).

4.5.1 Experimental Protocol

The experimental protocol we designed is identical for each bi-objective permutation problem. We study three different problem sizes $N \in \{20, 50, 100\}$ as found in the literature benchmarks, with three levels of correlation between the two objectives: *no correlation*, *mid-correlation* and *high-correlation* as explained in Section 3.4. For each couple size-correlation, we generated 10 instances, and for each instance, the neighborhood of 30 random solutions, denoted as *Rnd*, and 30 Pareto local optimal solutions, denoted as *Opt* are evaluated and analyzed. Note that the neighborhood is studied under the best-known neighborhood operator for each permutation problem.
The epsilon-indicator values are normalized following the protocol proposed in Section 4.3.1. We compute $I_\varepsilon(s', s) = max_k(\varepsilon_k, 0)$ and $I_\varepsilon(s, s') = max_k(-\varepsilon_k, 0)$ $(k \in \{1, 2\})$ to obtain a symmetric measure for non dominated solutions in bi-objective optimization with a same reference.
The analysis consists in computing different measures for both random and optimal solutions, for each considered instances. Then, an average of these values are applied to get one value per instance of:

- the percentage of neutral neighbors ($\mathcal{N}_n$),
- the percentage of promising ($\mathcal{P}_n$) neighbors among the neutral ones,
- their distribution in the different subsets ${\mathcal{P}_n}^\delta$, ${\mathcal{T}_n}^\delta$, ${\mathcal{H}_n}^\delta$ and ${\mathcal{V}_n}^\delta$ for $\delta \in \Delta = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 1\}$

4.5.2 Experimental results on FSP

First, Table 4.3 (a) shows that as expected, the number of neutral neighbors $\mathcal{N}_n$ decreases as the correlation between objectives increases. However, the size of the problem has no impact on $\mathcal{N}_n$ for a same level of correlation. For random solutions the proportion of promising neutral neighbors $\mathcal{P}_n$ is around 50 % which shows they are well distributed

among the neutral ones. This observation is not true for Pareto local optimal solutions, that have much less neutral neighbors and a smaller proportion of promising ones.

Table 4.3: Results for the bi-objective FSP (in percentage).

(a) Percentage of neutral and promising neighbors for both random and optimal solutions.

	Rnd		*Opt*	
	$\mathcal{N}_n$	$\mathcal{P}_n$	$\mathcal{N}_n$	$\mathcal{P}_n$
$\rho = 0$				
20	49.44	52.89	19.91	14.38
50	49.89	50.17	11.12	6.94
100	50.07	50.56	6.25	4.01
ρ: mid				
20	38.25	52.61	13.3	13.43
50	41.21	50.3	7.33	7.58
100	41.75	50.19	3.94	4.38
ρ: high				
20	23.11	51.3	5.53	14.47
50	23.01	50.72	2.82	9.7
100	25.76	50.91	1.62	6.15

(b) Distribution of promising neutral neighbors in the first horizontal layer.

	Rnd		*Opt*	
	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$
$\rho = 0$				
20	11,2	22,35	7,19	2,62
50	15,64	21,12	4,66	0,74
100	16,68	21,28	3,37	0,27
ρ: mid				
20	13,44	22,78	7,48	2
50	18,13	21,14	6,25	0,52
100	18,94	20,95	4,25	0,05
ρ: high				
20	21,35	20,76	11,94	1,1
50	26,97	17,97	9,06	0,39
100	26,15	18,73	6,01	0,01

Table 4.3 (b) focuses on the analysis of the first $\mathcal{H}$-layer ($\mathcal{H}_n^{0.1}$). For random solutions, $\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$ represents 2/3 of solutions of the layer for low correlation instances and one half for high correlated ones. On the contrary, for Pareto local optimal solutions, most of the solutions are within the triangle, which means very close to the initial solution s, and hence less interesting to consider for the diversity and the convergence. Therefore, in the following, only details on the distribution of neutral neighbors for random solutions will be given.

Table 4.4 indicates the distribution of promising neutral neighbors over $\mathcal{H}$-layers, according to several values of δ. The first observation is that the number of solutions decreases rapidly as the value of δ increases. Indeed, when a high degradation is allowed on one objective, a high improvement on the other one is required to compensate it.

The aim of Table 4.5 is to study solutions that produce a minimal improvement with a bounded degradation. The distribution among $\mathcal{V}$-layers shows that for low correlated instances, $\mathcal{V}$-layers with a medium δ have still an interesting number of solutions. This is less true for high correlated problems. Let us note that ${\mathcal{V}_n}^1$ is not empty which means that there exist some neighbors that produce an improvement larger than the worst degradation encountered within the whole neighborhood.

As a conclusion for the FSP, it seems interesting to consider neutral neighbors during the search. Indeed, for random solutions, 50 % of these ones are promising, and more than 1/3 of them belong to ${\mathcal{H}_n}^{0.1}$, so very close to the dominance part. Moreover, when objectives are not correlated, many promising neighbors produce an improvement much larger than the degradation they produce on the second objective (they belong to ${\mathcal{V}_n}^{\delta}$ for

Table 4.4: Distribution of the promising neutral neighbors in the $\mathcal{H}$-layers (Random solutions).

		$\mathcal{H}_n^{0.1}$	$\mathcal{H}_n^{0.2}$	$\mathcal{H}_n^{0.3}$	$\mathcal{H}_n^{0.4}$	$\mathcal{H}_n^{0.5}$	$\mathcal{H}_n^{0.6}$	$\mathcal{H}_n^{1}$
$\rho = 0$								
	20	33.55	10.74	4.28	1.84	1.08	0.56	0.84
	50	36.78	9.08	2.71	0.97	0.37	0.17	0.09
	100	37.96	9.17	2.48	0.68	0.2	0.05	0.02
ρ: mid								
	20	36.22	10.25	3.55	1.45	0.62	0.29	0.23
	50	39.28	8.11	1.99	0.6	0.21	0.07	0.04
	100	39.9	7.98	1.77	0.42	0.1	0.02	0
ρ: high								
	20	42.1	6.78	1.66	0.5	0.2	0.03	0.03
	50	44.94	4.78	0.83	0.13	0.03	0.01	0
	100	44.89	5.1	0.77	0.13	0.02	0	0

Table 4.5: Distribution of the promising neutral neighbors in the $\mathcal{V}$-layers (Random solutions).

		$\mathcal{V}_n^{0.1}$	$\mathcal{V}_n^{0.2}$	$\mathcal{V}_n^{0.3}$	$\mathcal{V}_n^{0.4}$	$\mathcal{V}_n^{0.5}$	$\mathcal{V}_n^{0.6}$	$\mathcal{V}_n^{1}$
$\rho = 0$								
	20	33.55	33.09	24.93	18.25	13.56	10.32	8.52
	50	36.78	30.21	18.37	10.82	6.52	4.04	2.56
	100	37.96	30.45	17.25	8.96	4.58	2.37	1.25
ρ: mid								
	20	36.22	33.02	22.82	15.35	10.44	7.24	5.38
	50	39.28	29.25	16.03	8.51	4.61	2.65	1.61
	100	39.9	28.93	14.79	6.96	3.23	1.53	0.74
ρ: high								
	20	42.1	27.53	15.27	8.27	4.76	2.84	1.73
	50	44.94	22.75	9.51	4.15	1.87	0.87	0.43
	100	44.89	23.84	9.66	3.7	1.41	0.55	0.23

large values of δ while belonging to ${\mathcal{H}_n}^{\delta}$ for small values of δ). Regarding Pareto local optimal solutions, very few neutral neighbors are promising, so the interest of using them, except (maybe) to escape from them is not as relevant as during the search.

4.5.3 Experimental results on QAP

The first observation is that results are very similar to the FSP ones. As for the FSP, Table 4.6 (a) shows that the number of neutral neighbors $\mathcal{N}_n$ decreases as the correlation between objectives increases. However, the size of the problem has no impact on $\mathcal{N}_n$ for a same correlation. For random solutions the proportion of promising neutral neighbors

$\mathcal{P}_n$ is around 50 % which shows that they are well distributed among the neutral ones. This observation is not true for Pareto Pareto local optimal solutions, that have much less neutral neighbors and a smaller proportion of promising ones.

Table 4.6: Results for the bi-objective QAP (in percentage).

(a) Percentage of neutral and promising neighbors for both random and optimal solutions.

	Rnd		*Opt*	
	$\mathcal{N}_n$	$\mathcal{P}_n$	$\mathcal{N}_n$	$\mathcal{P}_n$
$\rho = 0$				
20	50,29	52,36	26,17	16,46
50	50,23	51,18	17,94	11,36
100	50,11	50,8	14,19	8,31
ρ: mid				
20	28,87	51,56	7,73	18,99
50	29,27	50,33	5,22	18,16
100	29,16	50,4	3,69	15,19
ρ: high				
20	13,39	50,54	2,05	28,51
50	14,23	50,24	1,41	26,86
100	13,85	50,25	0,94	25,24

(b) Distribution of promising neutral neighbors in the first horizontal layer.

	Rnd		*Opt*	
	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$
$\rho = 0$				
20	5,9	17,97	7,8	2,73
50	8,47	20,14	8,68	1,23
100	10,78	21,1	7,51	0,37
ρ: mid				
20	10,25	23,31	15,29	2,44
50	13,99	23,53	17,45	0,49
100	17,35	23,16	15,01	0,14
ρ: high				
20	22,36	20,89	28,2	0,17
50	28,22	18,46	26,78	0,08
100	33,46	15	25,23	0,01

Table 4.6 (b) focuses on the analysis of the first $\mathcal{H}$-layer. As for the FSP, for random solutions, $\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$ represents around 2/3 of solutions of the layer for low correlation instances and one half for high correlated ones. On the contrary, for Pareto local optimal solutions, most of the solutions are within the triangle, which means very close to the initial solution s, and hence less interesting to consider. Therefore, in the following, only details on the distribution of neutral neighbors for random solutions will be given.
Table 4.7 indicates the distribution of promising neutral neighbors over $\mathcal{H}$-layers, according to several values of δ. The first observation is that the number of solutions decreases rapidly as the value of δ increases. Indeed, when a high degradation is allowed on one objective, a high improvement on the other one is required to compensate it and to belong to the first layer.
Table 4.8 studies solutions that produce a minimal improvement with a bounded degradation. The distribution among $\mathcal{V}$-layers shows that for low correlated instances, $\mathcal{V}$-layers with a medium δ have still an interesting number of solutions. This is less true for high correlated problems. As for the FSP, there are some neighbors belonging to ${\mathcal{V}_n}^1$; such solutions produce an improvement larger than the worst degradation encountered within the whole neighborhood.
As a conclusion on the QAP problem, and for the same reasons than for the FSP, it seems interesting to consider neutral neighbors during the search, as a large part of them are promising ones. The same question stays open as for the consideration of neutral neighbors when the Pareto local optima are close.

Table 4.7: Distribution of the promising neutral neighbors in the $\mathcal{H}$-layers (Random solutions).

	$\mathcal{H}_n^{0.1}$	$\mathcal{H}_n^{0.2}$	$\mathcal{H}_n^{0.3}$	$\mathcal{H}_n^{0.4}$	$\mathcal{H}_n^{0.5}$	$\mathcal{H}_n^{0.6}$	$\mathcal{H}_n^{1}$
$\rho = 0$							
20	23,88	12,3	7,2	3,94	2,15	1,33	1,56
50	28,61	12,61	5,71	2,49	1,07	0,42	0,27
100	31,87	11,98	4,52	1,62	0,56	0,18	0,07
ρ: mid							
20	33,57	11,2	4,06	1,74	0,57	0,26	0,16
50	37,51	9,65	2,41	0,6	0,13	0,03	0
100	40,51	8,01	1,56	0,27	0,04	0,01	0
ρ: high							
20	43,26	6,01	0,99	0,21	0,05	0,02	0
50	46,67	3,3	0,25	0,02	0	0	0
100	48,47	1,72	0,06	0	0	0	0

Table 4.8: Distribution of the promising neutral neighbors in the $\mathcal{V}$-layers (Random solutions).

	$\mathcal{V}_n^{0.1}$	$\mathcal{V}_n^{0.2}$	$\mathcal{V}_n^{0.3}$	$\mathcal{V}_n^{0.4}$	$\mathcal{V}_n^{0.5}$	$\mathcal{V}_n^{0.6}$	$\mathcal{V}_n^{1}$
$\rho = 0$							
20	23,88	30,28	28,48	24,05	19,02	14,91	12,56
50	28,61	32,74	26,67	19,15	12,79	8,13	5,17
100	31,87	33,07	23,79	14,86	8,58	4,69	2,48
ρ: mid							
20	33,57	34,52	25,76	18,06	12,14	7,78	5,37
50	37,51	33,17	20,17	10,87	5,47	2,71	1,31
100	40,51	31,17	16,09	7,26	3,12	1,27	0,52
ρ: high							
20	43,26	26,9	12,77	5,96	3,01	1,64	0,96
50	46,67	21,75	6,9	1,95	0,57	0,2	0,07
100	48,47	16,73	3,43	0,64	0,11	0,02	0

4.5.4 Experimental results on TSP

First, Table 4.9 (a), shows that the number of neutral neighbors $\mathcal{N}_n$ decreases faster as the correlation between objectives increases. However, the size of the problem has no impact on $\mathcal{N}_n$ for a same correlation. For random solutions the proportion of promising neutral neighbors $\mathcal{P}_n$ is still around 50 % which shows that they are well distributed among the neutral ones. This observation is not true for Pareto local optimal solutions, that have very much less neutral neighbors and a smaller proportion of promising ones.

Table 4.9: Results for the bi-objective TSP (in percentage).

(a) Percentage of neutral and promising neighbors for both random and optimal solutions.

	Rnd		*Opt*	
	$\mathcal{N}_n$	$\mathcal{P}_n$	$\mathcal{N}_n$	$\mathcal{P}_n$
$\rho = 0$				
20	50.76	51.15	29.9	9.84
50	50.06	49.79	21.12	7.29
100	50.08	50.27	16.2	5.01
ρ: mid				
20	26.53	50.79	10.04	17.46
50	27.42	50.26	8.62	13.66
100	27.84	50.28	6.84	10.14
ρ: high				
20	8.98	49.67	1.63	37.39
50	8.48	49.89	1.53	26.55
100	8.55	49.77	1.23	25.88

(b) Distribution of promising neutral neighbors in the first horizontal layer.

	Rnd		*Opt*	
	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$	$\mathcal{T}_n^{0.1}$	$\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$
$\rho = 0$				
20	4.5	21.08	1.81	2.49
50	6.52	22.84	2.49	2.12
100	7.89	23.99	2.39	1.4
ρ: mid				
20	12.64	23.43	7.36	5.13
50	16.89	23.69	8.25	3.51
100	18.66	23.47	7.26	2.02
ρ: high				
20	37.65	10.66	32.65	3.02
50	43.71	5.97	26.21	0.34
100	44.76	4.93	25.61	0.27

The analysis of the $\mathcal{H}$-layer (Table 4.9 (b)) shows that for random solutions, $\mathcal{P}_n^{0.1} \setminus \mathcal{T}_n^{0.1}$ represents 4/5 of solutions of the layer for low correlation instances (more than FSP and QAP) and only 1/3 for high correlated ones (less than FSP and QAP). For Pareto local optimal solutions, these ratio are enforced.
The analysis of the distribution over $\mathcal{H}$-layers (Table 4.10), shows some similarities with the two other problems. Indeed, in the same manner, the number of solutions decreases rapidly as the value of δ increases to reach the value 0 for large δ.
Table 4.11 shows that for low correlated instances, $\mathcal{V}$-layers with a medium δ have still an interesting number of solutions. This is less true for high correlated problems, where for large size problems this number may be null.
As a conclusion for the TSP, we can say that those results are slightly different from the FSP and the QAP. If for low correlated instances, the number of promising neutral neighbors encourages to consider neutral neighbors during the search, this is not true anymore for high correlated instances. This analysis contributes to show the important impact of correlation between objectives.

4.6 Conclusion and Perspectives

Summary In this chapter, we presented our contributions on multi-objective fitness landscape. First, we proposed several definitions of neutrality for multi-objective optimization: the Pareto-neutrality, and different variants of ε-neutrality and *HD*-neutrality. We used the binary epsilon (additive and multiplicative) and hypervolume difference indicators since they enable the comparison between multi-objective solutions. While the Pareto definition divides a solution neighborhood into three regions being the dominating neighbors, the

Table 4.10: Distribution of the promising neutral neighbors in the $\mathcal{H}$-layers (Random solutions).

	$\mathcal{H}_n^{0.1}$	$\mathcal{H}_n^{0.2}$	$\mathcal{H}_n^{0.3}$	$\mathcal{H}_n^{0.4}$	$\mathcal{H}_n^{0.5}$	$\mathcal{H}_n^{0.6}$	$\mathcal{H}_n^{1}$
$\rho = 0$							
20	25,58	11,51	6,25	3,61	1,96	1,09	1,15
50	29,37	11,2	5,24	2,4	0,98	0,38	0,22
100	31,88	11,14	4,61	1,77	0,62	0,18	0,07
ρ: mid							
20	36,08	9,6	3,2	1,16	0,48	0,22	0,05
50	40,58	7,69	1,63	0,3	0,05	0,01	0
100	42,13	6,84	1,16	0,14	0,01	0	0
ρ: high							
20	48,31	1,25	0,11	0	0	0	0
50	49,68	0,21	0	0	0	0	0
100	49,69	0,08	0	0	0	0	0

Table 4.11: Distribution of the promising neutral neighbors in the $\mathcal{V}$-layers (Random solutions).

	$\mathcal{V}_n^{0.1}$	$\mathcal{V}_n^{0.2}$	$\mathcal{V}_n^{0.3}$	$\mathcal{V}_n^{0.4}$	$\mathcal{V}_n^{0.5}$	$\mathcal{V}_n^{0.6}$	$\mathcal{V}_n^{1}$
$\rho = 0$							
20	25.58	32.59	32.22	28.39	23.28	18.83	15.05
50	29.37	34.05	29.46	22.41	15.54	9.94	6.03
100	31.88	35.13	28.23	19.65	12.27	6.96	3.68
ρ: mid							
20	36.08	33.04	23.56	15.2	9.6	6.13	3.66
50	40.58	31.38	17.69	8.66	3.97	1.72	0.7
100	42.13	30.31	15.51	6.67	2.49	0.87	0.28
ρ: high							
20	48.31	11.92	2.81	0.76	0.22	0.09	0.02
50	49.68	6.18	0.35	0.03	0	0	0
100	49.69	5.02	0.16	0	0	0	0

incomparable neighbors and the dominated neighbors, the definitions based on the indicators offer a wealth of information about the distribution of neighbors in the objective space. Specific areas of the neighborhood are characterized and some of them include some promising neighbors, defined later on. The bounded definitions based on the indicators are tunable, and thus for the characterized neutral neighbors areas as well. Then, these ones may be adapted with the local structure of the neighborhood. We considered the permutation flowshop scheduling problem (FSP) and the traveling salesman problem (TSP) with two or three objectives. While the objectives of the FSP were chosen to be

quite correlated, the objectives of the TSP were independent. This particularity leads to different neutral degrees and distributions of the neighbors for each definition investigated. These experiments raised two questions (Should neighbors be considered identically during the search? Does correlation impact the neutrality?) that have been addressed by the following study. Therefore, we proposed indicators to better characterize the neutral neighbors and, more particularly, the promising ones. This definition was based on the additive epsilon only, since as we have seen, it is easier to understand. The incomparable region around a solution was divided into different layers for which the sizes are tunable. We conducted our study on the two previous permutations problems and we added the quadratic assignment problem (QAP). However, we considered 2 objectives only and we constructed different correlation levels between the objectives in order to see if the results, previously obtained, could be partly explained by this. The experiments confirmed our intuition, but also allowed us to better understand the landscape of the different problems.

Neutrality-Aware Multi-objective Stochastic Local Search In order to integrate neutrality into metaheuristics in multi-objective optimization as it has been done in single-objective optimization (Marmion et al. 2011a; Vérel et al. 2004), clear definitions of neutrality were required. This work was a first step towards such definitions. In the future, we would like to use these definitions to analyze and compute indicators during the execution of the algorithm. Indeed, the algorithm should be able to adapt to local properties of the landscape. The number of neutral neighbors and their distribution may help the algorithms, in particularly multi-objective stochastic local search algorithms (see Chapter 2 for a clear description) to embed several mechanisms to perform better. For each of them, we explain our ideas to use neutral neighbors. In the selection phase, some solutions are selected to be explored in the following phase. The best choice might be the promising neighbors identified at the previous iteration. In the exploration phase, an efficient strategy is to stop the exploration of the neighborhood as soon as a dominating solution is found but it also accepts all non-dominated neighbors (i.e. Pareto-neutral neighbors) identified. We could imagine not to accept all neutral neighbors, since they can be numerous, but only the promising ones according to a bound evolving during the run. Hence, the archiving phase would be faster. Adaptive algorithms integrate control mechanisms and feedback computation to easily react to unknown properties of the landscape. According to the recent classification of B. Doerr and C. Doerr (2018), the control mechanisms can be, among others, state-dependent, success-based or learning-inspired. This latter type is the most interesting from a landscape analysis perspective. Indeed, considering adaptive multi-objective SLS algorithms, measures on neutral neighbors, could indicate if the next strategy to use should take into account the neutral neighbors or not, and even the promising ones. Moreover, it could help to fix the value of the bound in the different definitions of indicator-based neutrality.

Landscape-aware Automatic Selection of Multi-objective Algorithms Fitness landscape analysis (FLA) enables a better understanding of the problem/instance solved. Algorithm selection investigates the relation between algorithm performance and problem instance features. However, although it may seem obvious to use FLA measures in the automatic selection process, only few works have already investigated that

approach (Kerschke and Trautmann 2019). In the same way, with Lucas Pavelski and Myriam Delgado, we proposed a similar approach since we integrated simple FLA measures into our algorithm selection system to predict the best algorithm with its best parameters to solve the classical permutation flowshop problems with makespan or sum of flowtimes minimization (Pavelski et al. 2019). A natural perspective may be to extend our algorithm selection system to multi-objective optimization and to integrate multi-objective FLA measures. For example, those measures could be on global or local neutrality of the instances used to train the model. This would allow us to better characterize an instance in order to choose the best strategies in the different cases.

Chapter 5

Landscape-aware Stochastic Local Search Algorithms

In this chapter, we propose two stochastic local search algorithms designed from the *a priori* analysis of the problem for which they are adapted. Indeed, the nature of the landscape plays an important role on the easiness, or not, in solving an optimization problem since a stochastic local search algorithm moves from solution to another through the neighborhood structure. Algorithms, where information from the landscape analysis is computed and exploited at different times during the execution, are adaptive algorithms and more precisely, state-dependent algorithms (B. Doerr and C. Doerr 2018). The contributions presented in this chapter are derived from the PhD of Lucien Mousin.

The autocorrelation measure, the fitness-distance correlation, the neutral degree, the local optima networks are, among others, problem-independent features that can be computed to characterize the landscape. However, some properties of the problem directly impact the landscape structure such as the number of solutions or, the time to explore the neighborhood. In this chapter, our contributions focus on these particularities. Therefore, we present two independent studies on two different optimization problems: the no-wait flowshop scheduling problem (Mousin et al. 2019) and the feature selection problem (Mousin et al. 2016).

This chapter is organized as follows:

- Section 5.1 proposes an iterated greedy algorithm to solve the no-wait flowshop scheduling problem. This variant of the classical permutation scheduling problems imposes constraints on no waiting time when a task switches to the next machine. The analysis of local optima shows the presence of identical sequences. Then, we describe how these sequences are identified during the run and how they are exploited to reduce the search space. The experiments conducted show the interest of this approach.

- Section 5.2 proposes an adaption of a tabu search to solve a feature selection problem. The particularity of this tabu search is that it integrates a learning mechanism that estimates the good combination of features and then, reduces the neighborhood to explore. The experiments on different datasets encourage the use of this approach.

5.1 Reduction of the Search Space

In this section, we first present the target problem being the no-wait flowshop scheduling problem. Then, we describe our methodology to analyze the sequences of consecutive jobs and show how to exploit them into a metaheuristic. Experiments are conducted on the Taillard instances and lead to propose an iterative version that increases the performance.

5.1.1 The No-Wait Flowshop Scheduling Problem

5.1.1.1 Problem description

The No-Wait Flowshop Scheduling Problem (NWFSP) is a variant of the well-known permutation flowshop scheduling problem (FSP), where no waiting time is allowed between the processing of a job on the successive machines (Röck 1984). Regarding the complexity of the problem, it has been proved by Wismer (1972) that the NWFSP can be viewed as an *Asymmetric Traveling Salesman Problem*. In addition, Röck (1984) proved that for m-machines ($m \geq 3$), the NWFSP is NP-hard while, the 2-machines case can be solved in $O(n * log\ n)$ (Gilmore and Gomory 1964). The formal definition of the NWFSP is the same as one of the FSP (see Chapter 3, Section 3.2.4.1). The difference is the computation of the completion times to support the no waiting time constraint.
In this work, we consider the NWFSP with makespan (C_{max}) minimization. Figure 5.1 provides a representation of a solution for an instance with five jobs and four machines. The makespan of this solution is 22 units. The insertion operator is used to define the neighborhood relation.

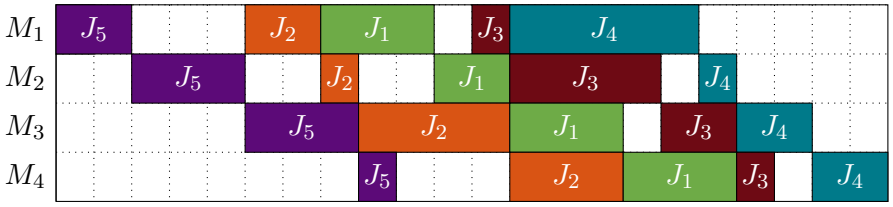


Figure 5.1: Gantt chart of the solution $\pi = \{5, 2, 1, 3, 4\}$ of a 5-jobs 4-machines NWFSP instance.

5.1.1.2 Literature Review

As mentioned before, the NWFSP is NP-Hard when the number of machines is strictly higher than two. Thus, exact methods were not able to find the optimal solution in a reasonable time for large-scale instances. Several heuristics and metaheuristics have been developed to tackle this problem.

Heuristic approaches They are mostly constructive and start from an initial sequence of jobs ordered according to a criterion. Then, they build a solution by inserting jobs in that order, to optimize the (partial) makespan at each step. Some heuristics are either

adaptations of heuristics developed for the classical FSP or have been specifically designed for the NWFSP. Hence, the well-known constructive heuristic NEH (Nawaz et al. 1983), initially designed for the FSP, has been successfully applied on the No-Wait variant. Among heuristics specifically designed for the NWFSP, we may cite GAN-RAJ (Gangadharan and Rajendran 1993) and RAJ (Rajendran 1994), BIH (Bianco et al. 1999), BH (Bertolissi 2000), LC (Laha and Chakraborty 2008)) and IBI (Mousin et al. 2017).

Metaheuristic approaches They are efficient methods to explore large search space and mostly able to find solutions with a higher quality than constructive heuristics. Both nature-inspired and stochastic local search (SLS) algorithms have been proposed to tackle the NWFSP. Regarding nature-inspired algorithms, we may find genetic algorithms (Aldowaisan and Allahverdi 2003), particle swarm optimization (Pan et al. 2008), or differential evolution (Qian et al. 2009). On the other hand, several SLS algorithms have also been proposed, such as tabu search (Grabowski and Pempera 2005; Samarghandi and ElMekkawy 2012), variable neighborhood search (Jarboui et al. 2010) or simulated annealing (Aldowaisan and Allahverdi 2003). At the beginning of the PhD of Lucien Mousin, (Ding et al. 2015) proposed a very efficient approach named TMIIG (Tabu-Mechanism Improved Iterated Greedy) based on a variable neighborhood search (Mladenović and P. Hansen 1997). In the perturbation phase, the authors use the efficient destruction-construction method of the Iterated Greedy (IG) (Ruiz and Stützle 2007), initially proposed for the FSP, and added a tabu mechanism to avoid scheduling a job at its previous positions during the different destruction-construction phases.

In 2015, TMIIG obtained the best performance to solve the Taillard instances, and it found new best solutions for the largest instances. However, during the PhD work of Lucien Mousin, Lin and Ying (2016) used the asymmetric traveling salesman problem formalization and applied the best exact methods, specifically designed for this problem, and were able to reach the optimal solutions for Taillard instances. In the following, the results are compared to TMIIG since the work of Lin and Ying (2016) was conducted at the same time and our approach is, nevertheless, interesting for the development of landscape-aware algorithms.

5.1.2 Super-jobs: Promising sequences of Consecutive Jobs

In the NWFSP, each job is processing without interruption between the successive machines. Therefore, one question arises: does this specificity lead to a particular structure of the best solutions of a given instance. In this section, we conduct an analysis of the global and local optimum solutions in order to extract structural information on them. It leads us to define a promising sequence of consecutive jobs as a *super-job*. Then, we present a methodology to identify *super-jobs* of an unknown instance in order to exploit them in the resolution.

5.1.2.1 Structural Analysis of Optimum Solutions

This analysis aims at extracting similarities in the structure of efficient schedules i.e. good quality solutions. We conduct this analysis on *small* instances (low number of jobs) with

	C_{max}	Solution permutation											
GO	1021	8	3	7	5	10	[9	1]	[0	4]	6	[11	2]
LO	1036	8	3	7	[11	2]	5	10	[9	1]	[0	4]	6
	1075	8	10	[9	1]	[0	4]	7	5	3	[11	2]	6
	1090	8	3	5	10	6	[0	4]	7	[11	2]	[9	1]
	1103	8	3	5	10	6	[0	4]	7	[9	1]	[11	2]
	1132	8	10	6	[0	4]	7	5	3	[9	1]	[11	2]
	1132	8	3	7	[11	2]	5	10	9	6	[0	4]	1
	∗1176	8	3	5	10	[9	1]	[11	2]	7	6	[0	4]
	1189	8	10	6	[0	4]	7	5	3	[11	2]	[9	1]
	1232	8	10	[9	1]	[0	4]	7	3	5	6	[11	2]
	1246	8	3	7	10	[9	1]	[11	2]	5	6	[0	4]

Figure 5.2: Description ($C_{\max}$+ solution permutation) of the global optimum (GO) and the 10 best local optima (LO) for illustrative instance (12 jobs and 5 machines). The sequences [11 2] and [0 4] colored in orange and green respectively, appear in all solutions and the sequence [9 1] colored in violet, appears in 10 solutions over 11. When these 3 sequences are considered as 3 unique jobs, only one local optimum remains (identified with the star ∗), the other ones are no longer local and are moved to the global optimum.

processing times uniformly generated following the methodology of Taillard's instances (see Chapter 3, Section 3.2.4.1). We report here, as an example, the analysis of an instance with 12 jobs and 5 machines. This problem size (12) enables to exhaustively enumerate the search space and therefore, to identify the global optimum and the best local optima considering the insertion neighborhood. Indeed, local optima are interesting to analyze since they may trap local search methods that explore the search space moving iteratively to improving neighbors. In the following, the term optimum solutions (or optima) is used to deal with the global or local optimum solutions more generally.

Figure 5.2 gives the global optimum (GO) and the 10 best local optima (LO) of the studied instance of size 12. For this small instance, it is easy to see that the LO share a similar structure between them and with the GO. Indeed, job 8 is always positioned at the beginning of the schedule and two sequences of two consecutive jobs are present in all of them: [0 4] in green, and [11 2] in orange and one sequence in 10 over 11 solutions: [9 1] colored in violet. LO solutions guarantee that they can not be improved by applying the insertion operator. However, if we consider each identified sequence of consecutive jobs as a unique job, the best LO ($C_{max} = 1036$) only differs from the global optimum ($C_{max} = 1021$) by the single move of the sequence [11 2] at the end of the permutation. Likewise, applying the insertion operator on the other LO with the consideration of the identified sequences instead of single job, moves all of them (except the one of $C_{max} = 1176$ identified with a star) to the GO.

As mentioned before, this study has been conducted on a small instance to be able to exhaustively enumerate the search space. Here, only sequences of two consecutive jobs

were found. However, the size is not limited. Hence, if a job a is always followed by a job b and, the job b is always followed by a job c then, the sequence $[a\ b\ c]$ of three consecutive jobs has to be considered rather than the two sequences $[a\ b]$ and $[b\ c]$ separately.
Observations made in this analysis motivate the substitution of original jobs by promising sequences of jobs to favor to reach better quality solutions. This transformation of the original problem may represent a good opportunity to solve large size instances. Therefore one new question arises: how to define and identify the promising sequences?

5.1.2.2 Definition of Super-Jobs

The previous exhaustive analysis of the structure of local optima for small size instances leads us to suppose that a similar behavior appears on larger ones. In this section, we present the methodology we propose to identify promising sequences of an unknown instance to be solved.
Regardless the size of the search space, good quality solutions and more precisely, good quality local optima, hopefully share a similar structure. When the search space is non enumerable, it is commonly admitted to use a sample of solutions in order to analyze their structure, characteristics ...
Here, we propose to extract the promising sequences from a pool, denoted $\mathcal{P}^*$, of *good quality* local optima and to define a *super-job* with a confidence of σ, any sequence of consecutive jobs that appears at least σ percent of times in solutions of $\mathcal{P}^*$. For example, if $\sigma = 50\%$, the super-jobs are the sequences of consecutive jobs shared by half of the solutions of $\mathcal{P}^*$. Let us note that only the longest sequences are considered as super-jobs. For example, if both $[a\ b]$ and $[b\ c]$ appear at least σ percent of times in $\mathcal{P}^*$, only $[a\ b\ c]$ is defined as a super-job.
This methodology has the advantage to be relevant regardless the problem size. However, the main drawback may be the computational time required to generate the pool of good quality solutions. Therefore, the size of the pool has to be fixed carefully: too large means that too much time would be spent to generate the pool, too small means that the identification of the super-jobs would be insignificant. This aspect will be discussed during the experimental section.

5.1.2.3 Advantages of Super-Jobs

The advantages of considering sequences of consecutive jobs (super-jobs) as unique jobs are several. First, for a complexity point of view, it will reduce the combinatorics of the problem i.e. the size of the search space. Secondly, for a local search point of view, this will modify the search space and the landscape induced by the insertion operator and so, new regions of the search space may become reachable more easily.
In order to provide a better understanding of the impact of super-jobs on the landscape, we present a visualization in 2D of the landscape transformation in Figure 5.3. Each point of the graph represents a solution. Blue points represent the original landscape, considering solutions constructed from original jobs following a neighborhood relationship (in the simplified representation, a solution has two neighbors). Red points represent the modified landscape obtained while considering super-jobs. We expect that the use of

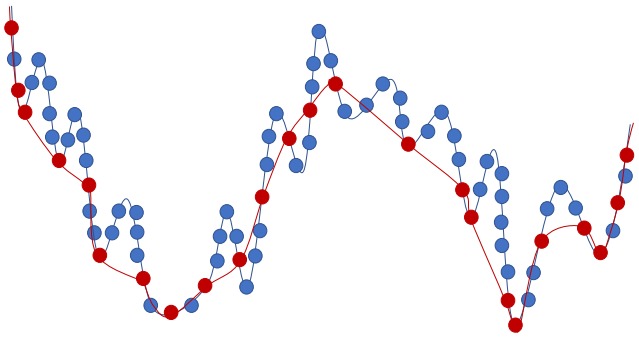


Figure 5.3: Visualization of the landscape in 2D. The blue points represent solutions of the original problem while the red ones represent those remaining with the super-jobs.

super-jobs locally smooths the landscape and makes easier the movement of a stochastic local search to avoid some original local optima which disappeared.

5.1.3 Iterated Greedy with Learning

Super-jobs have been defined as common structural characteristics of good quality solutions. In this section we propose an approach that exploits these super-jobs to improve an already efficient heuristic proposed in the literature for the FSP – the Iterated Greedy (IG) – in order to reach new best solutions for the Taillard instances. Thus, this section presents, first, the IG algorithm, and then our proposed approach.

5.1.3.1 Iterated Greedy Algorithm

The Iterated Greedy (IG) algorithm (Ruiz and Stützle 2007), initially proposed for the classical permutation flowshop scheduling problem, is an iterated local search, based on the insertion operator, whose perturbation phase removes some jobs from a solution, and reinserts them one by one at their best position i.e. the position that minimizes the partial makespan. The local search itself is an iterative improvement: each job of the sequence is considered in a random order and is re-inserted at its best position. This process is repeated until a local optimum is reached. The acceptance criterion of IG is inspired from the one of the simulated annealing and, checks if the new local optimum found is better or not than the best one ever found during the run. IG is known to be efficient to solve many variants of permutation flowshops. However, even if it is able to reach good quality solutions in a reasonable computational time, it is not able to reach, for the NWFSP, the best-known solutions of the largest instances of Taillard. However, the best algorithm for the NWFSP was the recent algorithm TMIIG (Ding et al. 2015), inspired from IG (see Section 5.1.1.2). Since the performance of IG is doubtless to solve small and medium sizes instances and, since the use of super-jobs decreases the problem size, we propose to design a new algorithm taking advantage of both IG and the super-jobs.

Algorithm 7: IG_{SJ}– Iterated Greedy with Learning algorithm.

Input: pool of solutions $\mathcal{P}^*$
Input: list of confidence levels in increasing order $\Sigma = \sigma_1, \sigma_2, \ldots$
Output: solution π
Data: list of super-jobs SJ

```
SJ = identify(P*, σ1) ;          /* Identifies Super-jobs with a confidence σ1  */
π = init(SJ) ;                   /* Initializes π with identified SJ */
foreach σ in Σ do
    SJ = identify(P*, σ) ;       /* Identifies Super-jobs with a confidence σ (i) */
    π = IG(π, SJ) ;              /* Runs IG from π with identified SJ (ii) */
return π
```

5.1.3.2 Iterated Greedy with Super-Jobs Algorithm

The Iterated Greedy with super-jobs algorithm (IG_{SJ}) identifies super-jobs of several increasing levels of confidence during the search and exploits them into the basic IG algorithm. Algorithm 7 gives the pseudo-code of this new algorithm. Given a pool of good quality local optima, whose generation will be discussed later, and an increasing list $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ of confidence levels, IG_{SJ} first identifies super-jobs regarding the first level of confidence σ_1. Afterwards, an initialization method generates a first solution π with these identified super-jobs. A process is then, iterated for each level of confidence of Σ, alternating between (i) a phase of super-jobs identification (except for the first level of confidence σ_1) and (ii) a phase of improvement using IG. The Iterated Greedy algorithm is executed on the solution considering super-jobs as jobs of the problem. As IG has no natural stopping criterion, a maximal time, as well as a maximal number of iterations without improvement, are used to stop the IG phase. Once all the levels of confidence have been used, the algorithm returns the best-found solution over the run.
Table 5.1 presents the evolution of a solution π for all phases of an execution of IG_{SJ} on the instance `ta023` of Taillard (20 jobs, 20 machines) with the list of confidence levels $\Sigma = \{60\%, 80\%, \infty\}$, where $\sigma = \infty$ means that no super-job is created (the problem is solved with all the original jobs). In this example, with $\sigma = 60\%$, in the first *identification* phase, seven super-jobs are identified: one of size 12, two of size 2 and the fourth-remaining ones of size 1. Therefore, the problem size is decreased from 20 to 7. The initialization method then builds, from these 7 jobs, a local optimum with a quality of 3021 that IG is not able to improve. Then, considering a confidence level of $\sigma = 80\%$, some previous super-jobs are decomposed during the next *identification* phase. Indeed, the largest super-job is decomposed into eight smaller ones. The problem size is equal to 14 and IG manages to find a better solution with a quality of 3013. It appears, in this special case, that the global optimum is reached within this second phase. This explains why the last phase ($\sigma = \infty$, super-jobs are all of size 1) is not able to produce any improving solution.

5.1.4 Experiments

In order to assess the efficiency of the IG_{SJ} algorithm, experiments are driven on Taillard instances and results compared to the best-known solutions (at the moment of this work)

σ	Phase	n	C_{max}	Solution π																	
60	*ident*	7	-	**[1**	**7]**	2	**[3**	**19]**	5	11	**[13**	**15**	**14**	**17**	**9**	**4**	**8**	**18**	**0**	**12**	**6**	**10]**	16
	init		3021	**[3**	**19]**	16	5	2	11	**[13**	**15**	**14**	**17**	**9**	**4**	**8**	**18**	**0**	**12**	**6**	**10]**	**[1**	**7]**
	IG		3021	**[3**	**19]**	16	5	2	11	**[13**	**15**	**14**	**17**	**9**	**4**	**8**	**18**	**0**	**12**	**6**	**10]**	**[1**	**7]**
80	*ident*	14	3021	**[3**	**19]**	16	5	2	11	**[13**	**15]**	14	17	**[9**	**4]**	8	**[18**	**0]**	12	**[6**	**10]**	**[1**	**7]**
	IG		3013	**[3**	**19]**	16	5	2	12	17	**[18**	**0]**	11	**[13**	**15]**	14	**[9**	**4]**	8	**[6**	**10]**	**[1**	**7]**
∞	*ident*	20	3013	3	19	16	5	2	12	17	18	0	11	13	15	14	9	4	8	6	10	1	7
	IG		3013	3	19	16	5	2	12	17	18	0	11	13	15	14	9	4	8	6	10	1	7

Table 5.1: Successive phases of IG_{SJ} on instance `ta023` of Taillard (20 jobs, 20 machines) with $\Sigma = \{60\%; 80\%; \infty\}$. For each confidence level, the identified super-jobs are in bold. n gives the number of jobs/super-jobs

of the literature obtained by TMIIG (Ding et al. 2015).

5.1.4.1 Experimental Protocol

In the experiments, we consider all the Taillard instances($N = \{20, 50, 100, 200, 500\}$ jobs and $M = \{5, 10, 20\}$ machines) for a total of 12 sizes and 10 instances per size. Following a preliminary study, several parameters was settled for these experiments:

- **Initial pool of solutions $\mathcal{P}^*$:** 10 solutions (enough to extract knowledge) were generated from 10 independent executions of IG with a maximal time of $n^2 * 10$ ms each.
- **Levels of confidence:** Two lists $\Sigma_1 = \{60\%, 80\%, \infty\}$ and $\Sigma_2 = \{60\%, 70\%, 80\%, 90\%, \infty\}$ were tested in order to evaluate the performance of IG_{SJ} according to the granularity.
- **Initialization method:** Iterated Best Insertion heuristic, a constructive heuristic hybridized with a basic local search (first improvement hill climbing) (Mousin et al. 2017).
- **Stopping criterion of IG:** A maximal time of $n_{sj}^2 * 10$ ms (where n_{sj} is the number of super-jobs of the phase), and a maximal number of iterations without improvement of $50 * n_{sj}$ are defined.

Each execution of IG_{SJ} on a given instance $\mathcal{I}$ returns a solution π of fitness $C_{max}(\pi)$. To measure the quality of the solution, the Relative Percentage Deviation (RPD) is computed relatively to the best-known solution of the literature π^* as follows:

$$RPD = \frac{C_{max}(\pi) - C_{max}(\pi^*)}{C_{max}(\pi^*)} * 100 \tag{5.1}$$

Hence a negative RPD indicates that a new best known solution is found.
IG_{SJ} is stochastic, thus 30 runs were executed to make the experimental results robust, and performance for an instance $\mathcal{I}$ corresponds to the average of the 30 RPD computed.

Machines \ Jobs	50										100										200										500									
5			17	2		21			11		X	14	5	29	13	11	16	5	6	4																				
10											26	20	7	24	18	26	22	X	23	23	24	19	X	X	X	X	29	28	X	X												
20											13	29	12	25	26	X	29	X	19	29	X	X	X	X	X	X	X	X	29	X	X	X	X	28	29	23	X	X	X	X		

Machines \ Jobs	50										100										200										500									
5	1		23	2		25			17		X	14	6	X	13	14	20	8	6	6																				
10											27	17	19	X	24	28	26	X	X	22	27	19	X	X	X	X	X	29	X	X												
20											17	X	22	26	27	X	X	X	27	29	X	X	X	X	X	X	X	X	X	X	X	X	X	29	26	X	X	X	X		

Table 5.2: Results on Taillard instances (organized by size) for $\Sigma_1 = \{60\%, 80\%, \infty\}$ (top) and $\Sigma_2 = \{60\%, 70\%, 80\%, 90\%, \infty\}$ (bottom). Gray cell: the best-known solution of the literature is reached. #/X : Number of times the best-known is improved (X = all runs).

5.1.4.2 Experimental Results

To analyze performance of the IG_{SJ} algorithm, our results are compared with the best-known solutions of the literature, reported in (Ding et al. 2015). Table 5.2 reports the results for both Σ_1 (top) and Σ_2 (bottom). It indicates, for each instance (10 instances per size) if the best-known solution of the literature is reached (cells colored in gray) and if this best-known is improved (non empty cell). Hence, when a number is present in a cell, this indicates the number of times the method improves the previous best-known solution over the 30 executions, an 'x' indicates 30/30. The results obtained for the 30 instances with 20 jobs are not reported here as the original IG ever reaches the optimal solution. The tables show that for all instances the method reaches the best-known of the literature regardless the two lists of confidence levels Σ_1 and Σ_2. In addition they show that all instances with 100, 200 and 500 jobs are improved with the proposed IG_{SJ} algorithm (there is non empty gray cell).

To deeper analyze the behavior of the method, some information about the execution of the method for the two lists Σ_1 (top) and Σ_2 (bottom) are reported in Table 5.3. Results presented are average over the 30 executions for the tem instances per size. The conclusions are similar between the two lists Σ_1 and Σ_2. We will first discuss about the common analyses and then point out differences within the discussion.
Left parts of these tables report the size of the problem i.e. the number of jobs for each phase. These jobs are either original jobs, or the super-jobs constructed by the concatenation of several jobs, as explained before. This measure gives the combinatorics of the problem. For example, for instances of size 200, when $\sigma = 60\%$, the number of jobs is around 80-90. This means that the size of the problem has been divided by more than 2. In the following phase, when $\sigma = 80\%$, some super-jobs are decomposed and the number of jobs is around 150-160. The combinatorics is still reduced. As mentioned before when $\sigma = \infty$ the number of jobs equals the original number of jobs (i.e. 200 for the previous

example). This first observation indicates that there is a real difference between the set of jobs obtained for the two levels of confidence, which shows that the identification of super-jobs is different.
Another observation is that the number of machines has also an impact on the identification of super-jobs. Indeed for a given number of jobs, the more the number of machines, the smaller the combinatorics. As far as instances of size 500 are concerned, the high level of combinatorics may be explained by the use of IG to generate the pool of solutions. Indeed, IG has difficulty to converge for large size problems within the time allowed. Hence the solutions of the pool are too diversified to identify common sequences of jobs.

Instances	Problem size		End of improvement				RPD value				Time (s)			
	60%	80%	Init	$IG_{60\%}$	$IG_{80\%}$	IG_{∞}	Init	$IG_{60\%}$	$IG_{80\%}$	IG_{∞}	60%	80%	∞	total
20×5	1.23	10.26	98.7	0.0	1.3	0.0	0.06	0.06	0.00	0.00	0.14	0.31	0.00	0.45
20×10	1.00	10.00	100.0	0.0	0.0	0.0	0.00	0.00	0.00	0.00	0.14	0.30	0.00	0.45
20×20	1.26	10.35	95.3	2.3	2.3	0.0	0.03	0.03	0.00	0.00	0.14	0.30	0.01	0.45
50×5	9.23	31.65	10.0	31.0	53.7	5.3	0.97	0.22	0.06	0.06	0.59	1.72	1.69	3.99
50×10	5.98	29.83	28.3	6.7	57.7	7.3	0.72	0.47	0.07	0.07	0.82	1.84	0.98	3.64
50×20	4.10	27.85	59.0	3.3	29.3	8.3	0.42	0.24	0.05	0.04	0.89	1.88	0.54	3.30
100×5	45.46	78.32	0.0	43.3	34.7	22.0	3.00	0.03	0.00	-0.01	6.88	7.07	8.98	22.93
100×10	29.04	68.99	0.0	16.3	53.7	30.0	1.69	0.07	-0.05	-0.07	3.25	7.48	9.38	20.11
100×20	27.37	68.44	0.0	9.3	59.0	31.7	1.49	0.09	-0.08	-0.10	2.82	7.71	9.22	19.74
200×10	95.36	157.41	0.0	15.3	45.7	39.0	2.89	-0.13	-0.18	-0.19	30.94	35.48	46.51	112.93
200×20	81.83	150.74	0.0	5.0	41.7	53.3	1.99	-0.20	-0.30	-0.32	22.12	39.35	48.80	110.27
500×20	268.87	409.28	0.0	0.7	30.3	69.0	2.81	-0.18	-0.24	-0.25	598.84	656.63	784.92	2040.38

Instances	Problem size				End of improvement						RPD value						Time (s)					
	60%	70%	80%	90%	Init	$IG_{60\%}$	$IG_{70\%}$	$IG_{80\%}$	$IG_{90\%}$	IG_{∞}	Init	$IG_{60\%}$	$IG_{70\%}$	$IG_{80\%}$	$IG_{90\%}$	$IG_{\in fty}$	60%	70%	80%	90%	∞	total
20×5	1.23	10.19	10.30	10.20	98.70	0.00	1.33	0.00	0.00	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.14	0.14	0.14	0.31	0.00	0.74
20×10	1.00	10.00	10.00	10.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.14	0.14	0.31	0.00	0.73
20×20	1.26	10.22	10.40	10.20	95.30	2.30	1.67	0.67	0.00	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.14	0.14	0.14	0.30	0.01	0.74
50×5	9.23	29.51	31.70	29.50	9.30	23.70	35.67	20.67	9.00	1.67	0.97	0.22	0.06	0.04	0.03	0.03	0.54	1.40	1.40	1.56	1.70	6.60
50×10	5.98	28.09	29.90	28.10	24.30	5.30	30.00	19.67	14.67	6.00	0.72	0.47	0.09	0.07	0.06	0.06	0.76	1.29	1.31	1.77	0.99	6.11
50×20	4.10	26.91	27.90	26.90	56.70	3.00	14.00	13.00	10.67	2.67	0.42	0.24	0.06	0.04	0.04	0.03	0.86	1.14	1.19	1.87	0.54	5.60
100×5	45.46	71.38	78.30	71.40	0.00	31.00	21.33	16.00	19.67	12.00	3.00	0.03	0.00	-0.01	-0.02	-0.03	6.82	6.38	6.94	7.63	8.91	36.68
100×10	29.04	63.72	69.00	63.70	0.00	9.00	25.33	26.67	25.33	13.67	1.69	0.07	-0.04	-0.07	-0.09	-0.09	3.32	6.57	6.80	7.01	8.92	32.62
100×20	27.37	62.91	68.50	62.90	0.00	8.00	16.33	32.00	28.67	15.00	1.49	0.09	-0.05	-0.10	-0.12	-0.13	2.90	6.95	7.11	7.01	8.63	32.60
200×10	95.36	144.20	157.50	144.20	0.00	5.70	19.33	28.00	23.00	24.00	2.89	-0.13	-0.18	-0.20	-0.22	-0.22	31.09	30.83	32.45	35.71	43.83	173.91
200×20	81.83	138.14	150.90	138.10	0.00	0.70	14.67	28.00	28.67	28.00	1.99	-0.20	-0.28	-0.32	-0.33	-0.34	22.09	34.00	33.18	35.29	45.30	169.85
500×20	268.87	376.81	409.50	376.80	0.00	0.00	5.33	15.67	31.33	47.67	2.81	-0.18	-0.23	-0.26	-0.27	-0.28	601.14	566.95	585.10	631.06	744.54	3 128.79

Table 5.3: IG_{SJ} with $\Sigma_1 = \{60\%, 80\%, \infty\}$ (top) and $\Sigma_2 = \{60\%, 70\%, 80\%, 90\%, \infty\}$ (bottom). Reported measures are averages over the 10 instances of each size.

Measures about the convergence of IG_{SJ} are given in the middle part of the tables. The column, called *End of improvement*, indicates in percentage, the number of times (over the 30 executions) the method – *Init* or *IG* – reaches the best solution of the run (average of the 10 instances of each size). We can observe, that in both tables, for instances of size 20, the best solution is mainly reached during the initialization phase. Indeed, for small size instances, IG is very efficient and manages to reach best-known solutions. For size 50, the IG_{SJ} manages to almost always find its best solution before considering the original problem ($\sigma = \infty$) contrary to the original IG. This validates the use of super-jobs. However, for largest size problems, some improvements are still obtained in the last phase when the original problem is considered.
This analysis is re-enforced by the third set of columns that report the average RPD at each phase. Thus it indicates, how far from the best-known solution of the literature, are solutions reached after each phase. For instances of size 20, the best-known solution, which is optimal, is reached (RPD=0). For size 50, the best-known solutions are often reached

(but maybe not in all the executions, which explained a small positive RPD). The most interesting is for large instances, as best known solutions are improved (negative RPD). A complementary interesting observation for large size instances is the high improvement between the initialization and the first phase with $\sigma = 60\%$. In the following phases, even if some better solutions are reached, the improvement is less important but significant.
The right part of the two tables reports time spent at each phase. The stopping criterion, at each phase, is either a maximal time or a maximal number of iterations without improvement, both depending on the number of jobs. So, as we can expect, for most cases, the time spent increases with the value of confidence level. Note that the total time is quite significant (the computational time for the generation of the pool of solutions still have to be added), but the objective of the approach is to be able to find new best solutions. So we did not consider the computing time like an optimization constraint.

5.1.4.3 Discussion

The experimental results proved the performance of our approach (IG_{SJ}) since new best solutions have been found out for every largest Taillard instance. This section provides a discussion on two important aspects of the method: the computational time required for the generation of the pool of solutions, and the analysis of the dynamic of the method.

Discussion on the generation of the pool $\mathcal{P}^*$ The computational time of the proposed method may represent a drawback for the resolution of large instances. This computational time is partly explicated by the generation of the pool of solutions $\mathcal{P}^*$.
The fact is that the performance of the approach is based on the knowledge extraction from this pool of initial solutions used to identify pertinent super-jobs. Preliminary results show that the quality of the solutions of the pool highly impacts the performance, and hence constructive heuristics do not give enough good quality solutions to identify reliable super-jobs. Hence, we chose IG, with a time limit, to give pretty good quality solutions for the pool of solutions, even if it is time consuming. In addition, we tested different sizes for the pool. Indeed, the higher the size, the larger the computational time to generate the pool whereas the lower the size, the lower the chance to have a representative pool. Following these tests we decided to generate a pool of 10 solutions only, as pertinent super-jobs can be found out, even with so few solutions. A small pool reduces a lot the whole computational time of the approach. However, the time still remains important. For example, to solve a 200 jobs instance, 400 seconds are required to generate one solution of the pool, hence around one hour for the 10 solutions of the pool. This is quite long, but not so important since we wanted to obtain new best solutions.
Since our approach is stochastic, in the exposed experiments, the performance is evaluated from 30 executions of IG_{SJ} for each instance. Each one generates its own pool of good quality solutions. We drove some parallel experiments where a single pool of 10 solutions was generated only and was shared between the 30 executions. The experimental results were similar: best-known solutions of the literature were reached for the smallest instances of Taillard, and improved for the largest one. Using such a shared pool decreases the whole computational time by thirty.
During the analysis of the method, we also noticed that for largest instances (with 500

jobs, mainly) a better pool of solutions improves the identification of super-jobs and then still reduces the combinatorics. Thus, we can imagine leaving more computational time to IG to generate a shared pool of better quality solutions, instead of generating one different pool for each execution.

Discussion on the behavior of the method Another interesting aspect is that the performance of the approach lies on the reduction of the combinatorics of the problem (and so the size of the search space) made possible by the particular structure of the best quality local optima. In the search space, each local optimum is the 'center' of a basin of attraction. All the basins make the landscape very rugged for stochastic local search algorithms. The perturbation phase of IG has been designed to escape from local optima, and so, from their attraction basins. However, the basins of attraction are not side-by-side but included in each other; the best local optimum of a large basin may be the center of other ones. Hence, even with a perturbation, a stochastic local search often remains in the same large basin of attraction in which it has started. The reduction of the combinatorics of the problem, with the identification of super-jobs, produces an interesting effect on the landscape. Indeed, some original local optima do not exist in the reduced landscape and so, for its basins of attraction. Therefore, regions of the landscape are smoothed, original basins of attractions get larger and, the performance of IG at each iteration of our approach is improved. For example, for instances with 20 jobs (the easiest of the Taillard instances), IG ends to converge close to the best-known solutions without reaching it, whereas with the reduction of the combinatorics, it reaches it each time. The reduction of the number of basins of attraction helps IG to move towards the best one. Exploiting super-jobs erases rugged regions of the search space and increases the performance of IG. These encouraging results lead us to incorporate (IG_{SJ}) in a more general scheme.

5.1.5 IIG_{SJ}: an Iterative Version

5.1.5.1 Description

The experimental results presented above, showed that the learning mechanism is efficient for both small and large size instances. Indeed, IG_{SJ} is able to either find out new best solutions or at least to reach the best-known solutions for the Taillard instances. For the largest (and actually the most difficult) instances of size 100, 200 and 500, new best solutions were discovered. However the successive RPD values (see Table 5.2) show that IG_{SJ} still improves the solution when the original problem is considered ($\sigma = \infty$). This suggests IG_{SJ} may be improved. For instances of size 500, we noticed that the initial solutions used to identify the super-jobs are very diversified (the size of the problem was barely reduced by two with the lower level of confidence $\sigma = 60\%$) because the original IG is not efficient for this size. Undoubtedly, this has an effect on the performance of the whole execution of IG_{SJ}. Iterating IG_{SJ} from the solutions obtained at the end of the 30 executions may improve the quality as well as the size of super-jobs identified and so the performance of the algorithm.
The proposed iterative approach (IIG_{SJ}) given in Algorithm 8 is based on this idea.
IIG_{SJ} starts with a set $\mathcal{P}_0$ of R solutions, iterates I times the inner procedure and returns the best solution π^* found. The inner procedure aims at building sets of solutions with

Algorithm 8: IIG_{SJ}: Iterative IG_{SJ}

Input: pool of solutions $\mathcal{P}_0$
Input: $\Sigma = \sigma_1, \sigma_2, \ldots$ list of confidence levels in increasing order
Input: number of iterations I
Input: number of solutions built at each iteration i
Input: number of solutions used for learning ρ
Data: set of solutions built at iteration i, $\mathcal{P}_i$
Data: temporary set of solutions $\mathcal{P}_{tmp}$
Data: temporary solution π
Output: best solution found π^*

```
π* =best(P_0) ;            /* Initialize the best solution with the best solutions of P_0 */
for i in 1..I do
    // INNER PROCEDURE
    P_i = ∅ ;                                   /* Initialize P_i as an empty set */
    for k in 1..R do
        P_tmp = pick(ρ, P_{i-1});
        ;                  /* Pick ρ solutions among P_{i-1} to be stored in P_tmp (i) */
        π = IG_SJ(P_tmp, Σ) ;                                          /* (ii) */
        P_i = P_i ∪ π ;                                  /* Store π in P_i (iii) */
        π* =best(π, π*) ;                          /* Memorize the best solution */
return π*
```

better and better qualities in order to identify super-jobs hopefully being those of the optimal solution. At iteration $i \in I$, it starts with $\mathcal{P}_i$ as an empty set where R new solutions will be iteratively added following these three steps: (i) first, ρ solutions are uniformly picked at random from the set $\mathcal{P}_{i-1}$ and stored in a temporary set $\mathcal{P}_{tmp}$ then (ii), IG_{SJ} is applied with $\mathcal{P}_{tmp}$ and Σ to obtain a new solution π that is finally (iii), stored in $\mathcal{P}_i$, the set of the current iteration i; if π is better than the current π^* then it replaces it. The parameter ρ is used to select a subset of solutions and then to maintain the diversity in the constructed pool $\mathcal{P}_{tmp}$, otherwise the same super-jobs would be identified for each confidence level in phase (ii).

5.1.5.2 Experimental Protocol

IIG_{SJ} presents its own parameters to settle in addition to those of IG_{SJ}. The number of solutions R of a pool $(\mathcal{P}_i)_{i\geq 0}$ has been settled to 20 that is statistically reasonable to evaluate the average performance of the approach; the number of solutions ρ to 10 like in the previous experiments where 10 solutions were used to identify the super-jobs, and the number of iterations I to 5 being enough to converge and to prevent over-learning. The original iterated greedy is still the algorithm used in IG_{SJ} to improve solutions. It is stopped when either a maximal time of n_{sj}^2 ms (where n_{sj} is the current number of super-jobs) or a maximal number of iterations without improvement of $25 * n$ is reached. Note that this stopping criterion is shorter than in the previous experiments. Indeed, since the process is iterated, the end of the convergence is not mandatory for each execution of IG. Moreover, the use of IG_{SJ} requires the setting of the parameter Σ to identify the

Instance	Best	Gap	Instance	Best	Gap	Instance	Best	Gap	Instance	Best	Gap
Ta01	1,486	0	Ta31	**3,160**	-1	Ta61	**6,366**	-31	Ta91	**15,248**	-71
Ta02	1,528	0	Ta32	3,432	0	Ta62	**6,219**	-15	Ta92	**15,007**	-78
Ta03	1,460	0	Ta33	**3,210**	-1	Ta63	**6,108**	-13	Ta93	**15,276**	-100
Ta04	1,588	0	Ta34	**3,338**	-1	Ta64	**6,001**	-25	Ta94	**15,117**	-83
Ta05	1,449	0	Ta35	3,356	0	Ta65	**6,183**	-17	Ta95	**15,113**	-96
Ta06	1,481	0	Ta36	**3,346**	-1	Ta66	**6,058**	-16	Ta96	**14,997**	-112
Ta07	1,483	0	Ta37	3,231	0	Ta67	**6,224**	-23	Ta97	**15,300**	-95
Ta08	1,482	0	Ta38	3,235	0	Ta68	**6,115**	-15	Ta98	**15,162**	-75
Ta09	1,469	0	Ta39	**3,070**	-2	Ta69	**6,359**	-11	Ta99	**15,012**	-88
Ta10	1,377	0	Ta40	3,317	0	Ta70	**6,371**	-10	Ta100	**15,259**	-81
Ta11	2,044	0	Ta41	4,274	0	Ta71	**8,059**	-18	Ta101	**19,551**	-130
Ta12	2,166	0	Ta42	4,177	0	Ta72	**7,859**	-21	Ta102	**19,980**	-116
Ta13	1,940	0	Ta43	4,099	0	Ta73	**8,017**	-11	Ta103	**19,791**	-122
Ta14	1,811	0	Ta44	4,399	0	Ta74	**8,330**	-18	Ta104	**19,775**	-153
Ta15	1,933	0	Ta45	4,322	0	Ta75	**7,939**	-19	Ta105	**19,732**	-111
Ta16	1,892	0	Ta46	4,289	0	Ta76	**7,773**	-28	Ta106	**19,852**	-90
Ta17	1,963	0	Ta47	4,420	0	Ta77	**7,851**	-15	Ta107	**19,967**	-145
Ta18	2,057	0	Ta48	4,318	0	Ta78	**7,881**	-32	Ta108	**19,900**	-156
Ta19	1,973	0	Ta49	4,155	0	Ta79	**8,137**	-24	Ta109	**19,817**	-101
Ta20	2,051	0	Ta50	4,283	0	Ta80	**8,095**	-19	Ta110	**19,794**	-141
Ta21	2,973	0	Ta51	6,129	0	Ta81	**10,676**	-24	Ta111	**46,264**	-425
Ta22	2,852	0	Ta52	5,725	0	Ta82	**10,562**	-32	Ta112	**46,797**	-478
Ta23	3,013	0	Ta53	5,862	0	Ta83	**10,591**	-20	Ta113	**46,154**	-390
Ta24	3,001	0	Ta54	5,788	0	Ta84	**10,588**	-19	Ta114	**46,556**	-343
Ta25	3,003	0	Ta55	5,886	0	Ta85	**10,507**	-32	Ta115	**46,402**	-339
Ta26	2,998	0	Ta56	5,863	0	Ta86	**10,624**	-66	Ta116	**46,667**	-274
Ta27	3,052	0	Ta57	5,962	0	Ta87	**10,793**	-32	Ta117	**46,170**	-339
Ta28	2,839	0	Ta58	5,926	0	Ta88	**10,801**	-38	Ta118	**46,495**	-378
Ta29	3,009	0	Ta59	5,876	0	Ta89	**10,703**	-20	Ta119	**46,408**	-335
Ta30	2,979	0	Ta60	5,958	0	Ta90	**10,752**	-46	Ta120	**46,433**	-414

Table 5.4: Best known solutions of Taillard instances. A bold value indicates a new best solution was found out by our approach.

jobs with different levels of confidence. We use $\Sigma = \{60\%, 70\%, 80\%, 90\%, \infty\}$ since the previous experiments showed better results for large size instances with this setting.

5.1.5.3 Experimental Results

Table 5.4 reports the best-known solutions for all Taillard instances and gives the gap value between the results obtained by our approach and the previous best-known solutions of the literature obtained by TMIIG (Ding et al. 2015). A gap equal to 0 means IIG_{SJ} reaches the best-known solutions of TMIIG, and a strictly negative gap means it finds out a solution with a better quality i.e. a new best-known solution. This table shows that for the largest instances of size 100, 200 and 500 jobs, IIG_{SJ} improves the good results already obtained with IG_{SJ} and finds out new best-known solutions. The quality of the best solution has been improved up to 478 like for the instance Ta112 for example. Clearly, IIG_{SJ} is very efficient to solve uniform instances of the no-wait flowshop scheduling problem like Taillard instances.

In the following, we detail the results obtained by IIG_{SJ} and discuss the interest of using

	Iter 0	Iter 1		Iter 2		Iter 3		Iter 4		Iter 5		Total
Instances	RPD	time (s)	RPD	time (s)	RPD	time (s)	RPD	time (s)	RPD	time (s)	RPD	time (s)
20×5	0.000	14	0.000	14	0.000	14	0.000	14	0.000	14	0.000	71
20×10	0.000	14	0.000	14	0.000	14	0.000	14	0.000	14	0.000	71
20×20	0.015	15	0.001	14	0.000	14	0.000	14	0.000	14	0.000	71
50×5	0.276	129	0.073	112	0.029	103	0.020	101	0.013	100	0.009	545
50×10	0.128	124	0.059	107	0.037	101	0.020	97	0.019	98	0.019	527
50×20	0.120	111	0.034	100	0.016	97	0.010	96	0.010	94	0.010	497
100×5	1.015	742	0.317	370	0.162	253	0.134	236	0.107	229	0.102	1830
100×10	0.577	332	0.215	282	0.147	254	0.134	243	0.124	231	0.112	1341
100×20	0.587	324	0.222	265	0.159	244	0.137	226	0.127	206	0.124	1264
200×10	1.431	1835	0.441	1281	0.187	1084	0.102	1005	0.085	976	0.084	6181
200×20	1.180	1726	0.366	1269	0.206	1116	0.134	1048	0.105	908	0.082	6068
500×20	1.748	32906	0.598	20789	0.299	16668	0.157	14747	0.090	11728	0.055	96838

Table 5.5: Analysis of the 5 iterations of IIG_{SJ}. Results are presented according to the 12 different sizes of the Taillard instances $N \times M$. The RPD value of a run is computed from the best-known quality reported in Table5.4. Times are given in seconds. For each iteration, the reported values of RPD and time are the average values computed over 200 runs.

IG_{SJ} iteratively. We present the results by grouping Taillard instances according to the 12 different sizes ($N \times M$) since the number of jobs N and the number of machines M impact the resolution of the problem. In our experiments, one iteration of IIG_{SJ} provides 20 solutions. All the solutions obtained after each iteration during a run of IIG_{SJ} are memorized in order to make an *a posteriori* analysis to validate the interest of iterating the approach of IG_{SJ}. After running IIG_{SJ} on Taillard instances, the qualities of the solutions obtained after each iteration and the (new) best-known quality (values reported in Table 5.4) are compared to compute the RPD value. Table 5.5 gives the average RPD computed from the 200 values obtained (20 solutions per iteration, 10 instances by size) for each iteration of IIG_{SJ}. A null RPD value means that the quality of the 200 solutions provided at the considered iteration is equal to the best-known quality for each of the 10 instances respectively. A strictly positive RPD value means that at least one solution provided at the end of the iteration does not have the best-known quality. As expected, the average RPD decreases with the successive iterations that shows the interest of exploiting the solution provided by an iteration to identify new and better super-jobs.

This decrease is illustrated on Figure 5.4 that shows the associated boxplots for the most difficult instances (20-jobs instances are optimally solved from the beginning). A line separates each graphic: the left part corresponds to the first iteration (like one execution of IG_{SJ}) while the right part corresponds to the next iterations performed in IIG_{SJ}. We observe that the larger the instance size, the larger the improvement of the median quality. Between iteration 4 and iteration 5, for different instances (*eg.* 100× 10, 100× 20, 200× 10), some qualities are even deteriorated. This may be explained by over-learning where the solutions in the sets $\mathcal{P}_3$ or $\mathcal{P}_4$ are too similar, and so the approach has difficulties to detect new super-jobs and get stuck in a particular region of the search space. To break through this drawback, we should think about introducing diversity between each iteration.

Table 5.5 reports also, in seconds, the average execution time for each iteration and the

average of the total time. Since the execution time depends on the number of (super-)jobs, it increases a lot when the number of jobs increases. This value nearly reaches 96,838 seconds for the largest instances (size 500) i.e. almost 28 hours. Obviously, this time is not satisfactory practically. But, in these experiments, our goal was to simply improve the performance of our initial approach IG_{SJ} to find out new best solutions ; what was done. If we analyze more carefully the average times for each iteration, we observe a reduction of the time inversely proportional to the instance size, the larger this reduction the higher the number of jobs. We may explain this by the increase of the size of the super-jobs and the decrease of their numbers. Solutions are more and more similar within the set and so may share larger sequences of jobs that gives a smaller and smaller number of jobs.

5.2 Reduction of the Neighborhood Size

In this section, we investigate an optimization approach able to jointly deal with large datasets where feature selection is needed before using time-consuming classifiers. This approach based on the Tabu Search integrates a learning mechanism in order to evaluate only promising subsets of features.

5.2.1 The Feature Selection Problem in classification

5.2.1.1 Problem description

In a classification problem, a set of observations with known classes is used to learn a classification model to predict the class of any new observations. A feature selection process may be used to select information that may help the classification. In this context, a dataset (in the following called *instance*) is represented by a set of d observations. Each observation i is characterized by n features and one class. Hence an instance is represented by a matrix A of d rows and n columns which represents the value of each feature for each observation, and a vector C of size d which represents the class of each observation, as follows:

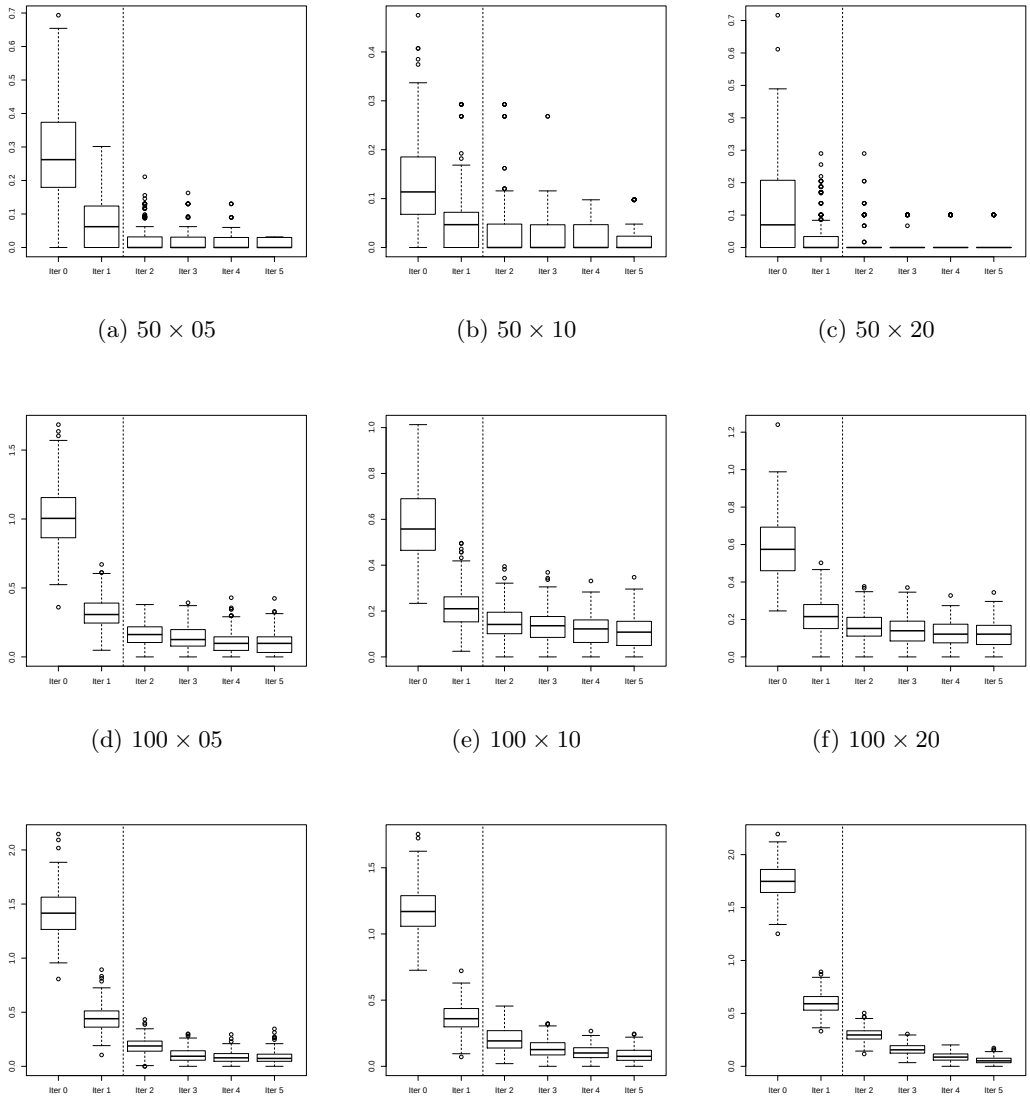
$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{d1} & \cdots & a_{dn} \end{bmatrix}, \; C = \begin{bmatrix} c_1 \\ \vdots \\ c_d \end{bmatrix} \tag{5.2}$$

where $c_i \in \{1, ..., k\}$ with k the number of classes.
An instance is composed by two sets. The first one, called *training set*, allows resolution approaches to learn a model and, the second one, called *validation set*, is used to evaluate that model on new observations.

5.2.1.2 Resolution approaches

For this problem, resolution approaches may be classified in three major types according to the way the search procedure and the classifier are combined:

- **Filter approaches**: Select features independently of the classification method used.



(a) 50×05 (b) 50×10 (c) 50×20

(d) 100×05 (e) 100×10 (f) 100×20

(g) 200×10 (h) 200×20 (i) 500×20

Figure 5.4: Boxplots of the RPD values for 5 iterations of IIG_{SJ}.

- **Wrapper approaches**: Exploit the classifier performance to select features. This type of approaches is used in this chapter, and detailed hereafter.
- **Embedded approaches**: Combine filter and wrapper approaches. They are used to reduce overfitting.

The wrapper model, initiated by R. Kohavi (Kohavi and John 1997), applies a search procedure to find different subsets that are evaluated with a classifier on the training set. The best subset found during the search procedure is then evaluated on the validation set (see

Figure 5.5). An advantage of this approach is to be able to deal with correlations between features and to find relevant associations of them. However, this kind of approaches may generate overfitting, i.e. the specialization of the model to observations used to build the model. Moreover, the computing time may become large with regard to the classifier used, when the dataset contains a large number of observations and/or features.

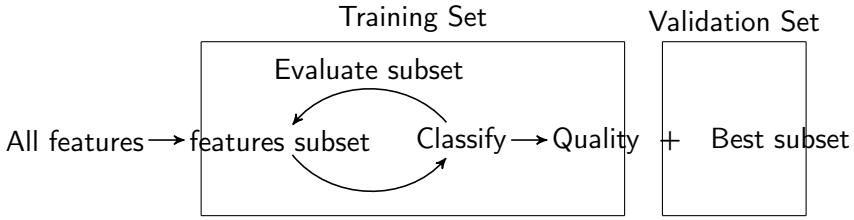


Figure 5.5: Wrapper approach

5.2.1.3 Literature Review

Finding the best subset of features can be viewed as a combinatorial optimization problem. Hence, a lot of methods, such as metaheuristics have been proposed to solve it. Table 5.6 reports some metaheuristics from the literature used to tackle this problem together with the type of approach used for resolution.

Table 5.6: Metaheuristics for the feature selection problem. The reference and the resolution approach are given.

Ref	Algorithm	Approach
Yang and Honavar (1998)	Genetic Algorithm with DistAl	Wrapper
Emmanouilidis et al. (2000)	Niched Pareto Genetic Algorithm	Wrapper
Oliveira et al. (2006)	Genetic Algorithm	Wrapper
Long et al. (2007)	HillClimbing	Filter +Wrapper
Hamdani et al. (2007)	NSGA II	Wrapper
Duval et al. (2009)	Genetic Algorithm + Iterated Local Search	Embedded
Cai et al. (2010)	Multi-Cluster Feature Selection	Wrapper
Gheyas and Smith (2010)	Simulated Annealing and Genetic Algorithm	Wrapper
Cervante et al. (2012)	Particle Swarm Optimization	Wrapper
Xue et al. (2013)	Particle Swarm Optimization	Wrapper
Tan et al. (2014)	Modified micro Genetic Algorithm	Wrapper

This table shows that very recent methods have been proposed and most of them are wrapper approaches. The nature-inspired metaheuristics are highly represented and, in particular, genetic algorithms seem to be the favored metaheuristic for this problem. On the contrary, very few stochastic local search (SLS) algorithms are used.
When using an efficient classifier, such as *SVM* (Support Vector Machine) (Schölkopf and Smola 1998), on large datasets, the evaluation of a subset may be time consuming. In this context, nature-inspired metaheuristics which evolve a population of solutions, need to make many evaluations at each generation, and therefore, are not any more good can-

didate algorithms contrary to SLS algorithms. Indeed, they benefit from a neighborhood structure, and exploit it to guide the search and to spare some evaluations.
Following these remarks, we proposed a SLS that integrates a learning mechanism to reduce the neighborhood size and then, reduce the number of evaluations per iteration.

5.2.2 The Feature Selection Problem with Learning Tabu Search

The Learning Tabu Search (LTS) algorithm is an efficient SLS integrating a learning mechanism (Schindl and Zufferey 2013). This section presents the steps needed to adapt this method to the Feature Selection (FS) problem.

5.2.2.1 Model of the Feature Selection Problem

A solution s represents the features. It is represented by a bit string of size n, the total number of features: $s = [a_1, ..., a_n]$ with $a_i \in \{0, 1\}, \forall i \in \{1, ..., n\}$. The i^{th} bit a_i indicates if the feature i is chosen ($a_i = 1$) or, on the contrary, if it is not ($a_i = 0$). For the FS problem in classification, several criteria are commonly used to measure the quality of a solution. First, it may be measured by the quality of the classification realized using the selected features. Most of classifiers propose to compute the *accuracy*, which is defined as the ratio between the well-classified observations and the total number of observations tested. The *accuracy* is computed as follows:

$$accuracy = \frac{\text{number of well-classified observations}}{\text{total number of observations}} \tag{5.3}$$

Secondly, the number of selected features is an important criterion for FS problem. Indeed, in order to obtain more interpretable models, the number of selected features should be minimized. This criterion is defined as the ratio between the number of selected features (# S_Features) and the total number of features (# Features). In order to obtain a maximization criterion, the criterion, noted *features*, is defined as follows:

$$features = 1 - \frac{\#\text{ S_Features}}{\#\text{ Features}} \tag{5.4}$$

Here, we consider these two maximization criteria, *accuracy* and *features*. Note that, in the literature, other criteria are also used such as sensitivity or specificity. In this work, the FS problem is represented as a single-objective combinatorial optimization problem. Consequently, the fitness function f is defined as a weighted sum between *accuracy* and *features*:

$$f = \alpha * accuracy + (1 - \alpha) * features \tag{5.5}$$

where $\alpha \in [0, 1]$ is a weighting coefficient (set to 0.75 in the experiments). The goal is to find the subset of features that maximizes f.
For neighborhood structure, we consider the well-known *one-flip* operator defined, for all s in the search space, as follows:

$$\mathcal{N}_1^0(s) = \{s' \mid \exists i \in \{1, ..., n\} \text{ s.t. } a'_i \neq a_i \text{ and } \forall j \neq i,\ a'_j = a_j\} \tag{5.6}$$

As the number of selected features has to be minimized, a *good* solution is represented with most of bits equal to 0. Hence, the probability of flipping a bit from 0 to 1 is higher than flipping a bit from 1 to 0. Consequently, in order to give the same chance to both flips *0 to 1* and *1 to 0*, we divided the neighborhood into two sub-neighborhoods. The *add neighborhood* ($\mathcal{N}_A$) is the set of neighboring solutions where one bit has been flipped from 0 to 1. The *drop neighborhood* ($\mathcal{N}_D$) is the set of neighboring solutions where one bit has been flipped from 1 to 0. Then, $\mathcal{N}_1^0(s) = \mathcal{N}_A(s) \cup \mathcal{N}_D(s)$ and $\mathcal{N}_A(s) \cap \mathcal{N}_D(s) = \emptyset$. The neighborhoods $\mathcal{N}_A$ and $\mathcal{N}_D$ are mathematically defined as:

$$\mathcal{N}_A(s) = \{s' \mid \exists i \in \{1, ..., n\} \text{ with } a'_i = 1 \text{ and } a_i = 0 \text{ and } \forall j \neq i,\ a'_j = a_j\} \tag{5.7}$$

and

$$\mathcal{N}_D(s) = \{s' \mid \exists i \in \{1, ..., n\} \text{ with } a'_i = 0 \text{ and } a_i = 1 \text{ and } \forall j \neq i,\ a'_j = a_j\} \tag{5.8}$$

5.2.2.2 From Tabu Search to Learning Tabu Search

In SLS algorithms and particularly in Tabu Search, the exploration of the neighborhood of a solution can be time-consuming. Indeed, in the original Tabu Search method, all the non-tabu neighbors of a solution are evaluated at each iteration. In the FS problem, the evaluation of a solution is computed by applying a classification procedure that can be computationally expensive when the number of observations and/or features becomes large. Hence, the evaluation of the whole neighborhood at each iteration can not be considered. Schindl and Zufferey (2013) designed the Learning Tabu Search (LTS) in order to avoid this. They divide the exploration of the neighborhood into two steps: (i) the quality of all neighbors is estimated and, (ii) the Q most promising ones are *fully* evaluated. LTS is based on an *estimation function* used to estimate the potential quality of each neighboring solution.
The computation of this estimation is based on this idea: "if, some combinations of characteristics often belong to good solutions during the run, such combinations of characteristics should be favored when generating new solutions". The estimation of the quality of one combination is computed from the quality of solutions where this combination appears. Therefore, LTS needs a memory to save the quality of each features combination.

The performance of LTS relies on the definition of this memory that represents the learning mechanism. This mechanism is related to the pheromones concept of ant colony optimization (ACO) algorithms (Dorigo and Birattari 2010). The quality of one combination is then called its *trail* value. The higher the trail value of a combination, the better is its quality. Like in ACO, the memory has to be updated to increase the trail of promising combinations and to decrease those that are associated to bad ones. An *evaporation* procedure is used to forget them.

In LTS, the *update* procedure is applied at regular intervals, called *cycles*. The quality of the best solution found during each cycle is used to update the trail values. The size of the cycle is a sensible parameter of LTS, as it impacts the performance of the learning mechanism. The update procedure aims to concentrate the search in regions containing high quality solutions. In order to visit new regions of the solutions space, a *diversification*

procedure has been introduced. This procedure modifies the policy of choosing the Q most promising neighbors to be evaluated during the neighborhood exploration. Usually, a learning mechanism favors the neighbors with the highest estimation values but, it may lead to a premature convergence of LTS. To avoid this issue, when diversification is triggered, the combinations with the lowest estimation values are favored.

Algorithm 9: Learning Tabu Search (LTS)

Input: Initial solution s
Output: Best solution s^*
Data: Best solution of the current cycle s^*
$s^* \leftarrow$ s;
repeat
  Estimate the quality of non-tabu neighbors of $\mathcal{N}(s)$;
  $N_Q \leftarrow Q$ most promising neighbors of $\mathcal{N}(s)$ according to the diversification policy;
  $s \leftarrow \arg\max_{s' \in \mathcal{N}_Q} f(s')$;
  if $s > s^*$ **then**
    $s^* \leftarrow s$;
  if $s > \hat{s}$ **then**
    $\hat{s} \leftarrow s$;
  Update the tabu list;
  if *End of cycle* **then**
    Update trails of each combination with $\hat{s}$;
until *Stopping condition is met*;
return $s*$

Algorithm 9 gives an insight of LTS. From an initial solution, different steps are applied until the stopping criterion is met. Every non-tabu neighbors are estimated and then, the Q most promising neighbors are evaluated. *Most promising neighbors* stands for neighbors with the highest estimation when *diversification* is disabled but, ones with the lowest estimation when *diversification* is triggered. At the end of the neighborhood exploration, the best solutions s^* of the run, $\hat{s}$ of the current cycle and the tabu list are updated. At the end of each cycle, trail values are updated from the fitness of $\hat{s}$ according to the *diversification* policy.

5.2.3 Learning Tabu Search for Feature Selection

In the following, we explain the adaptation of LTS to the FS problem.

5.2.3.1 Definition of trail

We propose to consider the combination of two features. A combination of two features is interesting if these features are both selected in good solutions i.e. the combination of these two features brings information for the classification task. The trail value $tr(a_i, a_j)$

associated to features a_i and a_j, indicates if the combination of a_i and a_j is promising, thanks to the observations of the search history.

5.2.3.2 Estimation of neighbors

A solution s and each neighbor s' differ from one bit a_i. The estimation of a neighbor ($Estim(s, a_i)$) (i.e. its potential quality), is computed from the relevance of selecting the feature a_i in relation to other features in s:

$$Estim(s'_i) = Estim(s, a_i) = \sum_{a_j \in s, a_j \neq a_i} tr(a_i, a_j) * \delta_s(a_j) \begin{cases} \delta_s(a_j) = 1 & \text{si } a_j = 1 \text{ dans s} \\ \delta_s(a_j) = 0 & \text{sinon.} \end{cases} \tag{5.9}$$

5.2.3.3 Neighborhoods exploration

$\mathcal{N}_A$ and $\mathcal{N}_D$ are the neighborhoods composed with *add* flips and *drop* flips respectively. A *promising add* flip adds a feature a_i to a solution s, if $Estim(s, a_i)$ is high in order to select a feature which brings the most information to solution (i.e. the future classification). A *promising drop* flip removes a feature a_i from a solution s, if $Estim(s, a_i)$ is low. During the exploration of the neighborhood in LTS, only the Q best promising neighbors are evaluated. Then, A_q (resp. D_q) is the subset of non-tabu neighbors of $\mathcal{N}_A$ (resp. $\mathcal{N}_D$) composed of the q neighbors with the highest (resp. lowest) estimations. Finally, all neighbors of $A_q \cup D_q$ are evaluated and the best one is chosen.

5.2.3.4 Update procedure

As mentioned before, the trail values $tr(a_i, a_j)$ are updated at the end of each cycle from the best solution $\hat{s}$ found during the cycle: $tr(a_i, a_j) = \rho * tr(a_i, a_j) + \Delta tr(a_i, a_j)$, where $\rho \in [0, 1]$ is the *evaporation rate* and $\Delta tr(a_i, a_j)$ is proportional to the fitness of $\hat{s}$, if a_i and a_j both belong to $\hat{s}$, and is equal to 0 otherwise.

5.2.3.5 Diversification procedure

It is used to escape from a region of the search space. Therefore, when the mechanism is triggered, the construction of the sets A_q and D_q during the exploration of the neighborhood is inverted i.e. A_q (resp. D_q) is the subset of non-tabu neighbors of $\mathcal{N}_A(s)$ (resp. $\mathcal{N}_D$) composed of the q neighbors with the lowest (resp. highest) trail values. This mechanism depends on two parameters t_1 and t_2: the mechanism is triggered after t_1 iterations without improving s^* (the best solution found during the run), and is disabled as soon as s^* has improved, or after t_2 iterations with diversification.

5.2.4 Experiments

5.2.4.1 Experimental protocol

We choose to compare LTS to other SLS algorithms: a *Hill Climbing* (HC) and a *Tabu Search* (TS). Hill Climbing is a classic SLS algorithm, that has the major inconvenient,

to stop the search when it falls in a local optimum. In order to give the same chance for all algorithms, when HC is traaped in a local optimum, HC is restarted from a new solution until the allowed time is over. The Tabu Search is a SLS that uses a memory to escape from local optima. This memory is used to store recently visited solutions that are qualified as *tabu*. At each step, the tabu search moves to the best non-tabu solution of the neighborhood. Hence, the tabu search is able to escape from a local optimum by moving to the least deteriorating neighbor. In the literature, Tabu Search applies the *best improvement* strategy for the neighborhood exploration. Nevertheless, this strategy may be time-consuming when the evaluation is costly, therefore we choose the *first improvement* strategy.

Each instance used for experiments is divided into two parts. The first one is the *training set*, used by the algorithm to look for the best subset of features. The second one is the *validation set*, used to evaluate the ability of the subset of features previously found, to well classify new data. For each instance with their training and validation sets, we performed for each algorithm the following different steps: (*i*) the algorithm is run on the training set, (*ii*) the best solution found is selected and its accuracy on the validation set is computed, (*iii*) these two steps are executed 30 times per instance per algorithm, (*iv*) the statistical Wilcoxon test is performed on fitness obtained on the training set to compare algorithms, and (*v*) the statistical Wilcoxon test is performed on accuracy obtained on the validation set.

5.2.4.2 Description of the Instances

Experiments are computed using six instances from the literature. Each line of these instances represents an observation. Table 5.7 details information about the instances used for experiments (well-balanced binary classes). An important point is the classifier used to compute the accuracy. In this work, we used the Support Vector Machine (*SVM*) (Schölkopf and Smola 1998) that constructs hyperplanes to separate data into two classes. This procedure becomes time consuming when the number of observations increases and when data are difficult to separate into two classes. Hence, for such instances, the runtime needed by SVM to construct and then evaluate a model is very costly.

In consequence, we choose to distinguish two groups of instances (low evaluation cost vs. high evaluation cost) according to the SVM runtime when it is applied on the training set. The first group contains *Schizophrenia*, *Colon* and *Breast* instances (SVM runtime lower than 1 second) and the second one, *Arcene*, *DNA* and *Madelon* instances. Note that, SVM requires more than 38 seconds on *Madelon* instance to compute the accuracy on the whole training set. Preliminary experiments helped to set the allocated runtime given to HC, TS and LTS. This allocated runtime is the same for the three methods, and is partially dependent on SVM runtime, since it is used within the evaluation to compute the accuracy of a solution. Let us remark, that even if *Arcene* instance requires less than 2 seconds to compute the accuracy, preliminary experiments showed that the convergence is quite low but happened for each algorithm before 3000 seconds.

Table 5.7: Instances description. For each instance, the reference, the number of original features (# Features), the size of the training $|T|$ and validation $|V|$ sets (i.e. number of observations) are given. The runtime (in seconds) needed by SVM to build and evaluate a model on training set (without feature selection), and the runtime (in seconds) allocated to each optimization algorithm are given.

Name	Reference	# Features	$\|T\|$	$\|V\|$	SVM Runtime	Allocated Runtime
Schizophrenia	(Calhoun 2014)	410	56	30	0.01	500
Colon	(Zhu et al. 2007)	2000	62	32	0.052	120
Breast	(Zhu et al. 2007)	24481	78	26	0.734	500
Arcene	(Guyon et al. 2008)	10000	100	100	1.123	3000
DNA	(Guerra-Salcedo and Whitley 1999)	180	1400	600	1.172	500
Madelon	(Guyon et al. 2004)	500	2000	600	38.089	5000

Table 5.8: LTS parameters.

Parameter	Value
Size of Tabu List	7
Size of A_q and D_q (q)	10
Cycle (I)	10
Evaporation rate (ρ)	0.9
Number of iterations with diversification ($t1$)	10
Number of iterations without diversification ($t2$)	10

5.2.4.3 LTS Parameters Setting

Preliminary experiments have been carry out in order to settle the parameters of LTS. Table 5.8 shows parameters involved in this study.
Two parameters deserve special attention. The first one is q that controls the number of promising estimated neighbors from each set, A_q and D_q, that will be evaluated. Indeed, if q is small, LTS converges quickly because the first best solutions are often the same. Otherwise, if q is large, LTS becomes time-consuming because many solutions are evaluated. Note that q could be adapted to the instance size, but preliminary experiments show that $q = 10$ appears to be a good trade-off for these instances. The second one is the size of a cycle (I). If I is small, the learning mechanism will make overfitting because the search has not enough time to find a new best solution. Otherwise, if I is large, the learning mechanism will take much time to discover good combinations and to forgot bad ones. Preliminary experiments show that $I = 10$ appears to be also a good trade-off.

5.2.4.4 Performance analysis

This analysis is organized in two parts. The first part deals with the optimization perspective (capacity of the method to find a good subset of selected features i.e. with a high fitness value) and evaluates its performance on the training set. The second part

concerns the datamining perspective (capacity of the model to predict class of unknown observations) and evaluates results obtained on the validation set.

5.2.4.5 Analysis of the Optimization Approach

Table 5.9 shows a comparative study between the proposed approach LTS and the other approaches. For each instance, the accuracy computed with SVM from the whole features is pointed out in order to exhibit the benefit of the feature selection. This table shows that concerning results about the fitness, LTS gives in most of the cases the best results with a standard deviation close to zero. In details, we can see that LTS often gives the best accuracy and selects always the least number of features. This may be explained by the neighborhood exploration strategy. Indeed, LTS selects for evaluation the q best add flips as well as the q best drop ones. Consequently, drop flips have as much chances to be chosen as add flips. On the contrary, other approaches have a random neighborhood. As the number of selected features is small, the probability to find a drop flip is low and may required the evaluation of many neighbors. As a result, LTS can find a solution with a good accuracy with the least number of features faster than other algorithms. Table 5.9 also shows that, for each instance, LTS improves results obtained by the original Tabu Search. These results show the improvement obtained by the introduction of the learning mechanism.
In order to analyze the behavior of the different algorithms, we computed their evolution over time. Figure 5.6 shows the evolution of the average fitness of each approach over the time and gives the box-and-whisker plots (after one third, two thirds, and at the end of the allocated runtime) on *Madelon* instance, which is the most difficult instance to solve. For this one, LTS has a quick progression compared to the two other methods. Indeed, *Madelon* is a high-cost instance, so thanks to the estimation function, LTS avoids a large number of evaluations.
Consequently, LTS finds the potential good solutions more quickly than other approaches. These results show the interest of the estimation function.
To understand the behavior of the learning mechanism, we also investigate the dynamic of add and drop flips over time. Thus, Figure 5.7 shows, for one execution, the evolution of the different metrics (Fitness, Accuracy and # S_Features) for LTS on *Madelon* instance. We can observe several phases on this figure. The first one (from the beginning to 1000 sec approximately) adds features to increase the accuracy, and in consequence also the fitness. The second one starts when fitness is high. In this phase, the learning mechanism chooses the worst features to remove thanks to the trail values. As LTS removes features, which bring the least information, the accuracy decreases slightly while the second *part* of the fitness that favors small subsets of features increases. Consequently LTS makes a good trade-off between the accuracy and the number of selected features.

5.2.4.6 Analysis of the datamining approach

Table 5.10 shows the results about the accuracy values on both training and validation sets for each instance. The objective is to analyze the ability to make a good classification on the validation set, using features selected on the training set. A first observation is that performance decreases between the training set and the validation one. This difference

Table 5.9: Average and standard deviation (in brackets) of Fitness, Accuracy and # S_Features values obtained on training sets for HC, TS and LTS. Fitness values in bold stand for algorithms outperforming the other one(s) according to the Wilcoxon test. For each instance, the value of the accuracy obtained by SVM without any feature selection is pointed out in brackets. The statistical comparison between algorithms is given under the instance name.

Instance	Algorithm	Fitness	Accuracy (%)	# S_Features
Schizophrenia (69.64%) $LTS > HC > TS$	HC	$0.992_{(0)}$	$99.946_{(0.311)}$	$11.788_{(1.933)}$
	TS	$0.968_{(0)}$	$97.132_{(2.043)}$	$16.939_{(5.123)}$
	LTS	$\mathbf{0.995}_{(0)}$	$100_{(0)}$	$8.758_{(0.792)}$
Colon (87.09%) $(LTS = HC) > TS$	**HC**	$\mathbf{0.998}_{(0)}$	$99.951_{(0.281)}$	$10.909_{(3.440)}$
	TS	$0.982_{(0)}$	$97.752_{(1.845)}$	$10.97_{(2.628)}$
	LTS	$\mathbf{0.996}_{(0)}$	$99.609_{(0.809)}$	$6.788_{(1.745)}$
Breast (67.3%) $HC > (LTS = TS)$	**HC**	$\mathbf{0.98}_{(0)}$	$97.319_{(2.717)}$	$18.394_{(6.685)}$
	TS	$0.94_{(0)}$	$92.308_{(4.022)}$	$13.121_{(2.804)}$
	LTS	$0.94_{(0)}$	$92.075_{(4.100)}$	$11.879_{(2.858)}$
Arcene (83%) $LTS > HC > TS$	HC	$0.999_{(0)}$	$99.879_{(0.331)}$	$21.97_{(6.356)}$
	TS	$0.971_{(0)}$	$96.273_{(3.224)}$	$22.273_{(4.252)}$
	LTS	$\mathbf{0.999}_{(0)}$	$100_{(0)}$	$14.97_{(3.147)}$
DNA (89.57%) $LTS > (HC = TS)$	HC	$0.941_{(0)}$	$95.71_{(0.603)}$	$19.152_{(5.274)}$
	TS	$0.941_{(0)}$	$95.762_{(0.585)}$	$19.485_{(4.651)}$
	LTS	$\mathbf{0.945}_{(0)}$	$95.234_{(0.67)}$	$13.606_{(2.904)}$
Madelon (56,45%) $LTS > (HC = TS)$	HC	$0.712_{(0)}$	$64.135_{(0.576)}$	$37.273_{(9.593)}$
	TS	$0.714_{(0)}$	$63.885_{(0.673)}$	$31.03_{(6.849)}$
	LTS	$\mathbf{0.731}_{(0)}$	$65.152_{(0.327)}$	$15.606_{(3.201)}$

reveals overfitting, that is to say, the solution built on the training set is specific for these data. Consequently, the solution looses in prediction quality for new data. This is especially true for instances with few observations and confirms the difficulty to find a good classification model. The standard deviations obtained with the validation set on these instances are high and show a bad stability of the results produced. Conversely, in instances with a large numbers of observations, the standard deviations obtained with the validation sets are reasonable. Solutions are less sensitive to the data used for the validation. As for the training set, LTS obtains better or equivalent results than other approaches on the validation set. In particular, we observe an improvement about the results obtained by LTS compared to TS. In conclusion, these experiments show the good performance of LTS regarding both the optimization and the datamining perspectives. In particular, these experiments show the contribution of the learning mechanism, as LTS is able to find better subsets of features than the classical Tabu Search although they are based on the same components.

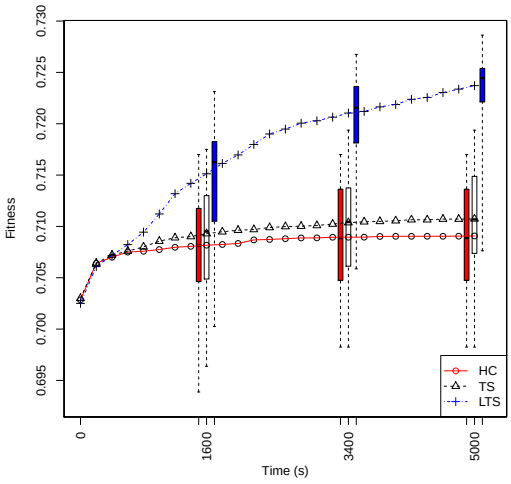


Figure 5.6: Evolution of the three algorithms on *Madelon* instance.

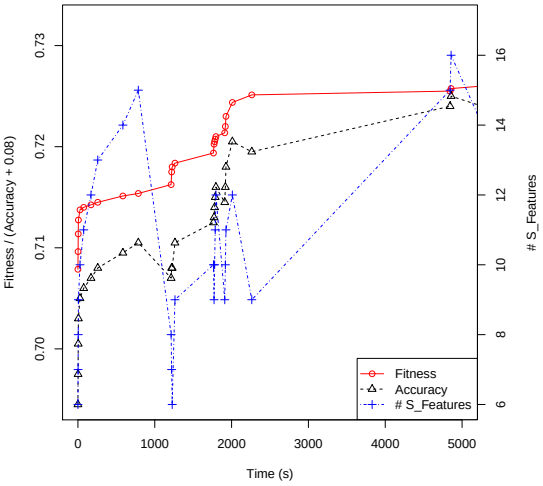


Figure 5.7: Evolution of Fitness, Accuracy and # S_Features values for LTS on *Madelon* instance. For more readability, the accuracy curve has been translated by +0.08.

5.3 Conclusion and Perspectives

Summary In this chapter, we presented two landscape-aware stochastic local search algorithms that modify the landscape differently. The first algorithm was specifically designed for the no-wait flowshop scheduling problem where the constraint imposed on switching between machines gives a property to be exploited. Indeed, an *a priori* landscape analysis of small instances reveals that similar sequences of jobs are found in the best solutions. In the first section of this chapter, we proposed a methodology to identify these sequences during the run and a way to directly exploit them. Actually, the proposed

Table 5.10: Average and standard deviation (in brackets) of {T,V}_Accuracy values obtained on respectively training and validation sets for HC, TS and LTS. Accuracy values in bold stand for algorithms outperforming the other one(s) according to the Wilcoxon test. The statistical comparison between algorithms is given. The double line shows the separation between low and high evaluation time cost.

Instance	Algorithm	T_Accuracy (%)	V_Accuracy (%)
Schizophrenia	HC	$\mathbf{99.946}_{(0.311)}$	$60.208_{(9.352)}$
	TS	$97.132_{(2.043)}$	$55.457_{(10.923)}$
	LTS	$\mathbf{100}_{(0)}$	$\mathbf{61.319}_{(8.162)}$
		$(LTS = HC) > TS$	$LTS > HC > TS$
Colon	HC	$\mathbf{99.951}_{(0.281)}$	$\mathbf{94.318}_{(5.150)}$
	TS	$97.752_{(1.845)}$	$91.004_{(6.287)}$
	LTS	$99.609_{(0.809)}$	$\mathbf{94.127}_{(4.863)}$
		$HC > LTS > TS$	$(LTS = HC) > TS$
Breast	HC	$\mathbf{97.319}_{(2.717)}$	$54.079_{(10.263)}$
	TS	$92.308_{(4.022)}$	$50.116_{(10.333)}$
	LTS	$92.075_{(4.100)}$	$52.098_{(9.904)}$
		$HC > (LTS = TS)$	$LTS = HC = TS$
Arcene	HC	$\mathbf{99.879}_{(0.331)}$	$72.18_{(4.004)}$
	TS	$96.273_{(3.224)}$	$71.69_{(4.707)}$
	LTS	$\mathbf{100}_{(0)}$	$\mathbf{74.60}_{(4.629)}$
		$(LTS = HC) > TS$	$LTS > HC > TS$
DNA	HC	$\mathbf{95.71}_{(0.603)}$	$93.63_{(1.582)}$
	TS	$\mathbf{95.762}_{(0.585)}$	$93.25_{(1.604)}$
	LTS	$95.234_{(0.67)}$	$94.09_{(1.540)}$
		$(HC = TS) > LTS$	$LTS = HC = TS$
Madelon	HC	$64.135_{(0.576)}$	$57.07_{(2.200)}$
	TS	$63.885_{(0.673)}$	$56.56_{(2.059)}$
	LTS	$\mathbf{65.152}_{(0.327)}$	$\mathbf{59.69}_{(1.219)}$
		$LTS > (HC = TS)$	$LTS > (HC = TS)$

algorithm extremely reduces the size of the problem solved and increases it at each iteration that reveals new regions of the search space. The experiments conducted on the Taillard instances show the efficiency of our algorithm to give new best solutions (to the best of our knowledge at the time) to the largest instances. However, this work was carried out in parallel to other people who modeled the problem like an asymmetric traveling salesman

problem. They used high performing exact methods and they found the optimal solutions (with guarantee). Our goal, in this PhD work, was to study the advantage of using or not knowledge into metaheuristics. In that sense, we achieved our objective.
The second algorithm is an adaptation to the feature selection problem of a tabu search integrating a learning mechanism to reduce the number of neighbors to be explored. We presented the whole model of the problem and defined all the components needed by the Learning Tabu Search to run. The experiments show the efficacy of the algorithm on a datamining problem. The idea of reducing the number of neighbors is interesting to keep in mind when the runtime represents a crucial point, but more generally, to speed up classical algorithms.
This work is interesting since it proves that integrating learning mechanisms into metaheuristics can be efficient. Nevertheless, we realized that extracting problem-dependent knowledge is a difficult challenge and represents a big obstacle that specialists of the problem may help to overcome.

Landscape-based Knowledge in VRP For example, recently, in the vehicule routing problem (VRP) community, Arnold and Sörensen (2019a,b) reached new best solutions for the large instances thanks to new algorithms integrating a learning mechanism. They learn about good sequences of clients and directly exploit them during the run. Since 2018, we are working with Diego Cattaruzza (INOCS team of CRIStAL) and Daniele Vigo (University of Bologna, Italy), who are both specialists in VRP, in order to improve metaheuristics with learning mechanisms driven by problem-dependent landscape characteristics.

Landscape Visualization to Help Decisions In my team, Nadarajen Veerapen is an Associate Professor who is specialized in landscape visualization (Ochoa et al. 2015; Veerapen and Ochoa 2018). The work of Lucien Mousin shows that it is very difficult to identify characteristics that may be useful to be exploited in the resolution of an optimization problem. Landscape analysis gives some *problem-independent* measures that are meaningful and understandable by metaheuristics optimizers, without being specialist of one type of problem. Therefore, landscape visualization might help the optimization specialist and the problem specialist to understand the characteristics of a problem. A collaboration in that sense might be fruitful.

General Perspectives

Every chapter of this manuscript ends with a conclusion and short-term perspectives. Our contributions focus on stochastic local search algorithms where permutation problems have been mainly used as case study. However, the different approaches such as the automatic configuration of a generalized structure, the multi-objective configuration and the landscape-based design are generalizable and obvious perspectives may be to apply them on nature-inspired algorithms as well as on other combinatorial optimization problems. In the following, we will present one perspective on automatic algorithm configuration (AAC) and three perspectives that join AAC with fitness landscape analysis.

Training Instances in AAC Automatic algorithm configuration can be formulated as an optimization problem where the objective is to find the optimal configuration (i.e. strategies and/or parameters values) of an algorithm. However, the optimal configuration is only adapted to the class of instances used to train the model. When we are defining the experimental protocol, the same question arises every time: How to choose the training instances? For example, let us consider the case of the permutation flowshop scheduling problem. Taillard instances are widely used to validate the performance. These instances give 120 instances where 10 instances share the same number of jobs $N \in \{20, 50, 100, 200, 500\}$ and machines $M \in \{5, 10, 20\}$. It is understood that the larger the number of jobs and machines, the more difficult the resolution. Therefore, the optimal configuration is clearly not the same regardless the size (i.e. numbers of jobs and machines). If the objective is to find the best configuration to solve instances with 100 jobs and 20 machines, are we supposed to use training instances with all the sizes? Or only the ones with 100 jobs and/or 20 machines? Or with a (uniform or Gaussian) distribution around 100 jobs and/or 20 machines? We would like to investigate this topic in order to give guidelines to better define experimental protocols.

Algorithm Configuration Landscapes An obvious, and currently popular in our field, perspective combining AAC and landscape analysis would be to study the landscape of the configuration space. Indeed, the configurator moves in a search space composed with the possible configurations of the target algorithm. The landscape is defined from a neighborhood relation between solutions. This relation is generally based on one single operator. However, a configuration is generally a set of both numerical and categorical parameters. While in ParamILS/MO-ParamILS, numerical parameters are discretized and so considered as categorical ones, in irace, discrete or continuous distributions are used according to the type of the parameter. Therefore, the definition of algorithm configuration landscape is not straightforward. One year ago, Pushak and Hoos (2018) studied these particular landscapes. They limited their study to numerical parameters and showed that the space is almost convex for each one. This study is a valuable starting point. We could imagine, in the future, to deeper analyze the algorithm configuration landscapes through different AAC approaches (e.g., statistical racing, stochastic local search, genetic

algorithm). Moreover, numerical parameters are often discretized when metaheuristic-based approaches are considered to configure an algorithm. In that case, it would be interesting to analyze the impact of discretization on the configuration space and so, on the performance of the metaheuristics.

Parameter Control Approach exploiting AAC and FLA Unlike AAC, parameter control is an on-line approach (Eiben et al. 1999) that modifies the configuration during the run to adapt the algorithm to the local structure. However, only a small number of parameters can be handled at the same time. B. Doerr and C. Doerr (2018) revised the former classification to better take into account the recent improvements on parameter control. They proposed to divide it into five categories namely *state-dependent*, *success-based*, *learning-inspired*, *self-adaptive* and *hyper-heuristic*. In Chapter 3, we presented an approach between AAC and parameter control. Following this new classification, we can characterize our contribution into *state-dependent* approaches since a time parameter gives the moment during the execution where the configuration has to be modified. In Blot et al. (2018c), we proposed a *success-based* approach to control the exploration strategy of a multi-objective local search and we obtained several interesting results. Indeed, we demonstrated that on the studied problem, taken individually, the three considered exploration strategies perform very differently, and that our approach can achieve results statistically equivalent to the best strategy. We also introduced a mechanism to discard some bad values during the execution, which is equivalent to a *learning-inspired* approach. Our preliminary experiments showed that it should be possible to adaptively identify surviving values of parameters/strategies during the execution for each tackled instance only from performance feedback. However, the learning mechanisms could have the opposite effect if it is only inspired by the execution. This may be due to the modification of the local structure. Therefore we would propose to use AAC, before the execution, on the same class of instances but in different regions of the search space, pre-identified with fitness landscape analysis (FLA). We could propose a *success-based* and *learning-inspired* parameter control approach where information computed/analyzed during the run would be interpreted on-line to trigger the modification of the configuration. Different steps may need to be studied: (i) detect different local structures using FLA, (ii) find the best configuration depending on the local structures with AAC and, (iii) integrate control/feedback mechanisms in order to switch to the best configuration according to the local structure identified and the configurations performance. Evidently, these perspectives can be carried out in both single- and multi-objective optimization.

Improving AAS with AAC and FLA Automatic algorithm selection (AAS) can be formulated as an optimization problem where the objective is to find the best algorithm for every instance. However, the configuration of the algorithms available in the portfolio is rarely optimal for the instance solved. Integrating AAC into the AAS process would enable the optimization of the algorithm together with its configuration. This approach has already been evaluated by M. T. Lindauer et al. (2015) on AI problems, such as SAT, CSP, ASP, MAXSAT and QBF. It gives encouraging results but, like in parameter control approaches, only few configurations are available per algorithm and another drawback is the very high computational cost required. AI problems are different from combinatorial

optimization problems and it would be interesting to adapt the proposed approach for permutation problems and more largely for other problems of operational research. However, based on our knowledge on these problems, we know that FLA measures can wisely be used to help the learning. Kerschke and Trautmann (2019) have proposed an AAS approach by combining FLA and machine learning to solve numerical optimization problems. Currently, we already use FLA into an AAS approach (Pavelski et al. 2019) but, the algorithms are configured *a priori* and the FLA measures are used to classify instances and then to suggest an optimal configuration for each algorithm. Moreover, the configuration of the algorithms would not be static as it has been discussed above. The goal of this work would be to directly integrate FLA into the AAS approach in addition to the AAC. Different steps may need to be studied: (i) characterize the instance to solve with FLA measures, (ii) generate a configuration space of a reasonable size and, (iii) integrate the adaptive approach presented above.

Appendix A

Extended CV

Marie-Eléonore Kessaci (Marmion)

4 July 1985 (34 years old)
Married, 1 child (2018)

http://www.cristal.univ-lille.fr/~marmion/

Work Experience

Since 2013 Associate Professor, CNU27, Classe normale, échelon 5 (sept. 2018)
Université de Lille – Polytech Lille
Centre de recherche en Informatique, Signal et Automatique de Lille (CRIStAL), UMR 9189
Team: ORKAD

2012-2013 Postdoctoral Researcher (ERCIM Alain Bensoussan Fellowship)
Université Libre de Bruxelles (Belgique)
Team: IRIDIA-CoDE

2011-2012 Research & Teaching Assistant
Université Lille 1 – UFR Informatique, Electronique, Electrotechnique, Automatique
Laboratoire d'Informatique Fondamentale de Lille (LIFL), UMR 8022
Team: DOLPHIN

2011-2012 Teaching Assistant
Université Lille 1 – UFR Informatique, Electronique, Electrotechnique, Automatique

Education

2008-2011 PhD in Computer Science (MENRT Grant)
Université Lille 1
Laboratoire d'Informatique Fondamentale de Lille (LIFL), UMR 8022
Team DOLPHIN (LIFL/CNRS/INRIA Lille-Nord Europe)
Advisors: Clarisse Dhaenens, Laetitia Jourdan
Title: "Local search and combinatorial optimization: from structural analysis of a problem to design efficient algorithms "
Defense board: Frédéric Saubion, Alexandre Caminada, Thomas Stützle, Sébastien Verel and Laurence Duchien

2006-2008 Master in applied Mathematics
Université Paris 6 Pierre et Marie Curie
Specialization: Mathematical engineering
Internship at Air Liquide R&D (team: Process Control and Logistics)
Topic: Design of a model to forecast liquid consumption of *bulk* clients

2005-2006 Bachelor in Applied Mathematics
Université François Rabelais (Tours)

PhD student Supervision

On going PhD thesis (3)

- Since Jan. 2019: Laurent Parmentier – Multi-objective automatic design of machine learning systems, co-supervision (50%) with Laetitia Jourdan (HDR).
  CIFRE with OVH company
- Since Oct. 2018: Weerapan Sae-dan – Automatic Design of Dynamic Local Search Algorithms, co-supervision (35%) with Nadarajen Veerapen and Laetitia Jourdan (HDR).
  Excellence Grant from Thai government
- Since Apr. 2017: Lucas Marcondes Pavelski – Meta-learning to design efficient optimization algorithms, co-supervision (50%) with Myriam Delgado (University of Parana, Brazil). Brasilian Grant.

Past PhD thesis (2)

- Oct. 2015 – Nov. 2018: Lucien Mousin – Extracting and exploiting knowledge to improve optimization performance, co-supervision (50%) with Clarisse Dhaenens (HDR). Grant from French government
  Now: Associate Professor at Université Catholique de Lille
- Sept. 2015 – Sept. 2018: Aymeric Blot – Designing Autonomous Methods for Multi-objective Combinatorial Optimisation, co-supervision (50%) with Laetitia Jourdan (HDR).
  ENS Grant.
  Now: Postdoctoral Researcher at the University College of London (UK)

Scientific Dissemination

International Collaborations

Myriam Delgado University of Technology of Paraná (Brazil)
Topic: Landscape-based algorithm selection in combinatorial optimization
Co-supervision of Lucas Marcondes Pavelski (PhD student).
Publications: GECCO 2019, Bracis 2018, CEC 2018

Manuel López-Ibáñez University of Manchester (UK)
Topic: Initialization techniques for multiobjective combinatorial optimization
Publication: PPSN 2018 (nominated for best paper award)

Holger Hoos LIACS, Leiden University (Leiden, The Netherlands)
Topic: Multi-objective automatic algorithm configuration. Design of MO-ParamILS
Publications: ECJ (2019), ICTAI 2018, EMO 2017, LION 2016

Patrick De Causmaecker KU Leuven University (Kortrijk, Belgique)
Topic: Design of adaptive multi-objective stochastic local search algorithms
Publication : LION 2018

Hernán Aguirre & Kyoshi Tanaka Shinshu University (Nagano, Japon)
Topic: Neutrality in multiobjective optimization: definition, analysis, interests
Publication: GECCO 2016 (nominated for best paper award)

Olivier Regnier-Coudert Gordon University (Aberdeen, UK)
Topic: Exploiting the factoradics representation to solve permutation problems
Publication: LION 2015

Scientific Animation and Administration

Journal Reviewing

- COR – Computers and Operations Research
- EJOR – European Journal of Operational Research
- ITOR – Transactions in Operational Research
- OMEGA – The International Journal of Management Science
- CAIE – Computers & Industrial Engineering
- Swarm Intelligence Journal
- International Journal of Metaheuristics

Program Committee Memberships

- Learning and Intelligent Optimisation LION (depuis 2015)
- Genetic and Evolutionary Computation Conference GECCO (2013, 2014)

Organisation of Scientific Events

- SLS workshop 2019: Lille (France) (in Sept.)
- Special Session in CEC conference: 2018 on landscape analysis with Katherine Malan (University of South Africa)
- Summer school ATOM 2017: Lille (France) (35 participants)
- International conference LION 2015: Lille (France) (70 participants)
  General chair with C. Dhaenens et L. Jourdan
- International conference META 2014: Marrakech (Morocco) (100 participants)

Member of Selection Committee for Associate Professor Recruitment

- Université Polytechniques des Hauts de France, 2019, poste 0312, optimization
- Université de Lille, 2018, poste 0048, optimization
- Université Lille 1, 2017, poste 1339, optimization
- Université Lille 1, 2017, poste 0591, software engeenering
- Université Lille 2, 2017, poste 4188, data sciences
- ENSIAME, Université de Valenciennes, 2017, poste 4152, optimization

Research Funding

as leader:

- BQR Internationalization 2018 (Université de Lille) – budget 3k€
- BQR Invited Professor 2017 (Université de Lille) for Myriam Delgado (University of Technology of Paraná, Brazil)

as participant:

- Interreg V France-Wallonie-Vlaanderen, PATHACOV, 2018-2022 – total budget: 4,6M€
- PHRCI-16-089, CATOCOV, 2018-2021 – total budget: 41k€
- CPER Data (PARSAT project), 2018-2021 – total budget: 50k€
- ANR ClinMine, 2013-2017 – total budget: 500k€

Teaching

Synthesis (around 220h ETD per year)

Year	Subject	Type	Level	# S	# h
2015-2019	Linear optimization$^\diamond$	LC, TC	4th (SE)	50	17
2018-2019	Databases	Project	3rd (SE)	50	22
2018-2019	Graph & combinatorics$^\diamond$	LC, TC, PC, Project	3rd (AE)	18	52
2015-2019	Graph & combinatorics$^\diamond$	LC, TC, PC, Project	3rd (SE)	50	52
2013-2018*	Algorithms	TC, PC, Project	3rd (SE)	50	46
2013-2018	Databases$^\diamond$	LC, TC, PC, Project	3rd (SE)	50	90
2016-2018*	Mathematics$^\diamond$ (upgrade)	LC-TC	3rd (SE)	25	25
2016-2017	Perational research$^\diamond$	LC-TC	4th (AE)	18	17
2014-2016	Optimization	PC	5th (SE)	25	10
2013-2016	Data mining	PC	5th (SE)	25	10

Meaning of the acronyms and signs:

- **SE**: Standard Education – **AE**: Apprenticeship Education
- **LC**: Lecture Course – **TC**: Tutorial Course – **PC**: Practical Course
- **3rd** year: Undergraduate's students, **4th** year: Master's students

- # **S**: number of students
- # **h**: number of hours ETD per year

- (*****) Maternity leave from September 2018 to October 2018, which made impossible for me to give these courses. However I will give them again next year.
- ($^\diamond$) Module coordinator

Pedagogical Responsibilities

2019 Responsible of the working group for the reorganization of maths education for the 3rd year students of GIS (Computer Engineering and Statistics) department of Polytech Lille.

2018 Responsible of the working group for the renewal of the pedagogical model of the GIS apprenticeship program to include modules related to Data Sciences.

Since 2016 Pedagogical manager for the 3rd year (SE) of GIS department (about 50 students)
Role: preparation of semester juries, timetable, 3rd year internship coordinator, management of temporary lecturers

Since 2016 Nominated member of the examination board of Polytech Lille (3 times a year)

Since 2014 Nominated member of the pedagogical commission of Polytech Lille (3 times a year)

2014-2016 Pedagogical manager for the 3rd year (AE) of GIS department (about 15 apprentices) Role: Organization of apprenticeship, design and update of the apprentice's electronic booklet, timetable, preparation of the semester juries...

Appendix B

Personal Bibliography after PhD

International peer reviewed journals

1 Blot, Aymeric, Marie-Eléonore Kessaci, Laetitia Jourdan, and Holger H. Hoos (2019). "Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems". In: *Evolutionary Computation* 27.1, pp. 147–171.

2 Blot, Aymeric, Marie-Eléonore Kessaci, and Laetitia Jourdan (2018). "Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation". In: *Journal of Heuristics* 24.6, pp. 853–877.

3 Dhaenens, Clarisse, Julie Jacques, Vincent Vandewalle, Maxence Vandromme, Emmanuel Chazard, Cristian Preda, Alexandru Amarioarei, Porpimol Chaiwuttisak, Cristina Cozma, Grégoire Ficheur, Marie-Eléonore Kessaci, Renaud Perichon, Julien Taillard, Régis Bordet, Amélie Lansiaux, Laetitia Jourdan, David Delerue, and Arnaud Hansske (2018). "ClinMine: Optimizing the Management of Patients in Hospital". In: *IRBM* 39.2, pp. 83–92.

Book editor

1 Dhaenens, Clarisse, Laetitia Jourdan, and Marie-Eléonore Marmion, eds. (2015). *9th International Conference on Learning and Intelligent Optimization, LION, Lille, France, January 12-15, 2015*. Vol. 8994. Lecture Notes in Computer Science. Springer.

International conferences with committee and proceedings

1 Pageau, Camille, Aymeric Blot, Holger H. Hoos, Marie-Eléonore Kessaci, and Laetitia Jourdan (2019). "Configuration of a Dynamic MOLS Algorithm for Bi-objective Flowshop Scheduling". In: *Proceedings of the 10th International Conference on Evolutionary Multi-Criterion Optimization, EMO, East Lansing, USA*. Vol. 11411. Lecture Notes in Computer Science. Springer, pp. 565–577.

2 Pavelski, Lucas M., Marie-Eléonore Kessaci, and Myriam R. Delgado (2019). "Meta-learning on Flowshop using Fitness Landscape Analysis". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Praha, Czech Republic*. ACM, pp. –.

3 Blot, Aymeric, Holger H. Hoos, Marie-Eléonore Kessaci, and Laetitia Jourdan (2018). "Automatic Configuration of Bi-Objective Optimisation Algorithms: Impact of Correlation Between Objectives". In: *Proceedings of the IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI, Volos, Greece*, pp. 571–578.

4 Blot, Aymeric, Marie-Eléonore Kessaci, Laetitia Jourdan, and Patrick De Causmaecker (2018). "Adaptive Multi-objective Local Search Algorithms for the Permutation Flowshop Scheduling Problem". In: *Proceedings of the 12th International Conference Learning and Intelligent Optimization, LION, Kalamata, Greece*. Vol. 11353. Lecture Notes in Computer Science. Springer, pp. 241–256.

5 Blot, Aymeric, Manuel López-Ibáñez, Marie-Eléonore Kessaci, and Laetitia Jourdan (2018). "New Initialisation Techniques for Multi-objective Local Search - Application to the Bi-objective Permutation Flowshop". In: *Proceedings of the 15th International Conference Parallel Problem Solving from Nature, PPSN, Coimbra, Portugal*. Vol. 11101. Lecture Notes in Computer Science. Springer, pp. 323–334.

6 Pavelski, Lucas M., Marie-Eléonore Kessaci, and Myriam R. Delgado (2018). "Meta-Learning for Optimization: A Case Study on the Flowshop Problem Using Decision Trees". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Rio de Janeiro, Brazil*, pp. 1–8.

7 Pavelski, Lucas M., Marie-Eléonore Kessaci, and Myriam R. Delgado (2018). "Recommending Meta-Heuristics and Configurations for the Flowshop Problem via Meta-Learning: Analysis and Design". In: *Proceedings of the 7th Brazilian Conference on Intelligent Systems, BRACIS, São Paulo, Brazil*. IEEE Computer Society, pp. 163–168.

8 Blot, Aymeric, Laetitia Jourdan, and Marie-Eléonore Kessaci (2017). "Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Berlin, Germany*. ACM, pp. 227–234.

9 Blot, Aymeric, Alexis Pernet, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Holger H. Hoos (2017). "Automatically Configuring Multi-objective Local Search Using Multi-objective Optimisation". In: *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization, EMO, Münster, Germany*. Vol. 10173. Lecture Notes in Computer Science. Springer, pp. 61–76.

10 Kessaci-Marmion, Marie-Eléonore, Clarisse Dhaenens, and Jérémie Humeau (2017). "Neutral Neighbors in Bi-objective Optimization: Distribution of the Most Promising for Permutation Problems". In: *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization, EMO 2017, Münster, Germany*. Vol. 10173. Lecture Notes in Computer Science. Springer, pp. 344–358.

11 Mousin, Lucien, Marie-Eléonore Kessaci, and Clarisse Dhaenens (2017). "A New Constructive Heuristic for the No-Wait Flowshop Scheduling Problem". In: *Proceedings of the 11th International Conference Learning and Intelligent Optimization, LION, Nizhny Novgorod, Russia*. Vol. 10556. Lecture Notes in Computer Science. Springer, pp. 196–209.

12 Blot, Aymeric, Holger H. Hoos, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Heike Trautmann (2016). "MO-ParamILS: A Multi-objective Automatic Algorithm Configuration Framework". In: *Proceedings of the 10th International Con-*

ference on Learning and Intelligent Optimization, LION, Ischia, Italy. Vol. 10079. Lecture Notes in Computer Science. Springer, pp. 32–47.

13 Marmion, Marie-Eléonore, Hernán E. Aguirre, Clarisse Dhaenens, Laetitia Jourdan, and Kiyoshi Tanaka (2016). "Multi-objective Neutral Neighbors': What could be the definition(s)?" In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Denver, USA*. ACM, pp. 349–356.

14 Mousin, Lucien, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Clarisse Dhaenens (2016). "Feature Selection Using Tabu Search with Learning Memory: Learning Tabu Search". In: *Proceedings of the 10th International Conference on Learning and Intelligent Optimization, LION, Ischia, Italy*. Vol. 10079. Lecture Notes in Computer Science. Springer, pp. 141–156.

15 Blot, Aymeric, Hernán E. Aguirre, Clarisse Dhaenens, Laetitia Jourdan, Marie-Eléonore Marmion, and Kiyoshi Tanaka (2015). "Neutral but a Winner! How Neutrality Helps Multiobjective Local Search Algorithms". In: *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, EMO, Guimarães, Portugal*. Vol. 9018. Lecture Notes in Computer Science. Springer, pp. 34–47.

16 Marmion, Marie-Eléonore and Olivier Regnier-Coudert (2015). "Fitness Landscape of the Factoradic Representation on the Permutation Flowshop Scheduling Problem". In: *Proceedings of the 9th International Conference on Learning and Intelligent Optimization, LION, Lille, France*. Vol. 8994. Lecture Notes in Computer Science. Springer, pp. 151–164.

17 López-Ibáñez, Manuel, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle (2014). "A template for designing single-solution hybrid metaheuristics". In: *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Vancouver, BC, Canada*, pp. 1423–1426.

18 Mascia, Franco, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Marie-Eléonore Marmion, and Thomas Stützle (2014). "Algorithm Comparison by Automatically Configurable Stochastic Local Search Frameworks: A Case Study Using Flow-Shop Scheduling Problems". In: *Proceedings of the 9th International Workshop Hybrid Metaheuristics, HM, Hamburg, Germany*. Vol. 8457. Lecture Notes in Computer Science. Springer, pp. 30–44.

19 López-Ibáñez, Manuel, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle (2013). "Automatic Design of a Hybrid Iterated Local Search for the Multi-Mode Resource-Constrained Multi-Project Scheduling Problem". In: *Proceedings of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications, MISTA, Ghent, Belgium*, pp. 820–825.

20 Marmion, Marie-Eléonore, Aymeric Blot, Laetitia Jourdan, and Clarisse Dhaenens (2013). "Neutrality in the Graph Coloring Problem". In: *Proceedings of the 7th International Conference on Learning and Intelligent Optimization, LION, Catania, Italy*. Vol. 7997. Lecture Notes in Computer Science. Springer, pp. 125–130.

21 Marmion, Marie-Eléonore, Franco Mascia, Manuel López-Ibáñez, and Thomas Stützle (2013). "Automatic Design of Hybrid Stochastic Local Search Algorithms". In: *Proceedings of the 8th International Workshop Hybrid Metaheuristics, HM, Ischia, Italy*. Vol. 7919. Lecture Notes in Computer Science. Springer, pp. 144–158.

International and national conferences with committee without proceedings

1 Pageau, Camille, Aymeric Blot, Holger H. Hoos, Marie-Eléonore Kessaci, and Laetitia Jourdan (2019). "A Dynamic Algorithm Framework to Automatically Design a Multi-Objective Local Search". In: *20ème Conférence ROADEF de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF, le Havre, France.*

2 Blot, Aymeric, Marie-Eléonore Kessaci-Marmion, and Laetitia Jourdan (2017). "AMH: a new Framework to Design Adaptive Metaheuristics". In: *12th Metaheuristics International Conference, MIC, Barcelona, Spain.*

3 Blot, Aymeric, Marie-Eléonore Kessaci-Marmion, and Laetitia Jourdan (2017). "AMH: une plate-forme pour le design et le contrôle automatique de métaheuristiques multi-objectif". In: *18ème Conférence ROADEF de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF, Metz, France.*

4 Mousin, Lucien, Marie-Eléonore Kessaci-Marmion, and Clarisse Dhaenens (2017). "An Iterated Greedy-based Approach Exploiting Promising Sub-Sequences of Jobs to solve the No-Wait Flowshop Scheduling Problem". In: *12th Metaheuristics International Conference, MIC, Barcelona, Spain.*

5 Mousin, Lucien, Marie-Eléonore Kessaci-Marmion, and Clarisse Dhaenens (2017). "De nouvelles meilleures solutions pour le problème d'ordonnancement No-Wait Flowshop". In: *18ème Conférence ROADEF de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF, Metz, France.*

6 Marmion, Marie-Eléonore, Franco Mascia, Manuel López-Ibáñez, and Thomas Stützle (2013). "Towards the Automatic Design of Metaheuristics". In: *10th Metaheuristics International Conference, MIC, Singapore.*

Bibliography

Abbasi, Maryam, Luıs Paquete, and Francisco B. Pereira (2015). "Local Search for Multiobjective Multiple Sequence Alignment". In: *Bioinformatics and Biomedical Engineering - Third International Conference, IWBBIO 2015, Granada, Spain, April 15-17, 2015. Proceedings, Part II*. Vol. 9044. Lecture Notes in Computer Science. Springer, pp. 175–182

Cited on page 28.

Aguirre, Hernán and Kiyoshi Tanaka (2005). "Random bit climbers on multiobjective MNK-landscapes: effects of memory and population climbing". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 88.1, pp. 334–345

Cited on pages 31, 34, 42 and 43.

Aldowaisan, Tariq and Ali Allahverdi (July 2003). "New heuristics for no-wait flowshops to minimize makespan". In: *Computers & Operations Research* 30.8, pp. 1219–1231

Cited on page 115.

Angel, Eric, Evripidis Bampis, and Laurent Gourvés (2004). "Approximating the Pareto curve with local search for the bicriteria TSP (1, 2) problem". In: *Theoretical Computer Science* 310.1-3, pp. 135–146

Cited on pages 31, 34, 42 and 43.

Ansótegui, Carlos, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney (2015). "Model-Based Genetic Algorithms for Algorithm Configuration". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. AAAI Press, pp. 733–739

Cited on page 4.

Ansótegui, Carlos, Meinolf Sellmann, and Kevin Tierney (2009). "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms". In: *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*. Vol. 5732. Lecture Notes in Computer Science. Springer, pp. 142–157

Cited on page 4.

Arnold, Florian and Kenneth Sörensen (2019a). "Knowledge-guided local search for the vehicle routing problem". In: *Computers & OR* 105, pp. 32–46

Cited on page 141.

Arnold, Florian and Kenneth Sörensen (2019b). "What makes a VRP solution good? The generation of problem-specific knowledge for heuristics". In: *Computers & OR* 106, pp. 280–288

Cited on page 141.

Arroyo, José Elias Claudio, Rafael dos Santos Ottoni, and Alcione de Paiva Oliveira (2011). "Multi-objective Variable Neighborhood Search Algorithms for a Single Machine Scheduling Problem with Distinct due Windows". In: *Electr. Notes Theor. Com-*

put. Sci. 281, pp. 5–19

Cited on page 29.

Bandyopadhyay, Sanghamitra, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb (2008). "A simulated annealing-based multiobjective optimization algorithm: AMOSA". In: *IEEE Transactions on Evolutionary Computation* 12.3, pp. 269–283

Cited on page 29.

Barba-González, Cristóbal, José Garcıa-Nieto, Antonio J. Nebro, José A. Cordero, Juan José Durillo, Ismael Navas Delgado, and José Francisco Aldana Montes (2018). "jMetalSP: A framework for dynamic multi-objective big data optimization". In: *Appl. Soft Comput.* 69, pp. 737–748

Cited on page 47.

Basseur, Matthieu and Edmund K. Burke (2007). "Indicator-based multi-objective local search". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore*. IEEE, pp. 3100–3107

Cited on pages 28, 31, 34, 42 and 43.

Basseur, Matthieu and Adrien Goëffon (2015). "Climbing combinatorial fitness landscapes". In: *Appl. Soft Comput.* 30, pp. 688–704

Cited on page 7.

Basseur, Matthieu, Rong-Qiang Zeng, and Jin-Kao Hao (2012). "Hypervolume-based multi-objective local search". In: *Neural Computing and Applications* 21.8, pp. 1917–1929

Cited on page 31.

Baykasoglu, Adil, Stephen Owen, and Nabil Gindy (1999). "A taboo search based approach to find the Pareto optimal set in multiple objective optimization". In: *Engineering Optimization* 31.6, pp. 731–748

Cited on page 29.

Beausoleil, Ricardo P. (2001). "Multiple Criteria Scatter Search". In: *Metaheuristics International Conference 4th*, pp. 534–539

Cited on page 29.

Bertolissi, Edy (2000). "Heuristic algorithm for scheduling in the no-wait flow-shop". In: *Journal of Materials Processing Technology* 107.1-3, pp. 459–465

Cited on page 115.

Bezerra, Leonardo C. T., Manuel López-Ibáñez, and Thomas Stützle (2016). "Automatic Component-Wise Design of Multiobjective Evolutionary Algorithms". In: *IEEE Trans. Evolutionary Computation* 20.3, pp. 403–417

Cited on page 53.

Bianco, Lucio, Paolo Dell'Olmo, and Stefano Giordani (1999). "Flow Shop No-Wait Scheduling With Sequence Dependent Setup Times And Release Dates". In: *INFOR: Information Systems and Operational Research* 37.1, pp. 3–19

Cited on page 115.

Blot, Aymeric, Hernán E. Aguirre, Clarisse Dhaenens, Laetitia Jourdan, Marie-Eléonore Marmion, and Kiyoshi Tanaka (2015). "Neutral but a Winner! How Neutrality Helps Multiobjective Local Search Algorithms". In: *Proceedings of the 8th International Conference on Evolutionary Multi-Criterion Optimization, EMO, Guimarães, Portugal*. Vol. 9018. Lecture Notes in Computer Science. Springer, pp. 34–47

Cited on pages 35, 37, 87, 88 and 89.

Blot, Aymeric, Holger H. Hoos, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Heike Trautmann (2016). "MO-ParamILS: A Multi-objective Automatic Algorithm Configuration Framework". In: *Proceedings of the 10th International Conference on Learning and Intelligent Optimization, LION, Ischia, Italy*. Vol. 10079. Lecture Notes in Computer Science. Springer, pp. 32–47

Cited on pages 50 and 54.

Blot, Aymeric, Holger H. Hoos, Marie-Eléonore Kessaci, and Laetitia Jourdan (2018a). "Automatic Configuration of Bi-Objective Optimisation Algorithms: Impact of Correlation Between Objectives". In: *Proceedings of the IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI, Volos, Greece*, pp. 571–578

Cited on pages 27, 41 and 49.

Blot, Aymeric, Laetitia Jourdan, and Marie-Eléonore Kessaci (2017a). "Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Berlin, Germany*. ACM, pp. 227–234

Cited on pages 27, 37, 40, 41 and 49.

Blot, Aymeric, Marie-Eléonore Kessaci-Marmion, and Laetitia Jourdan (2017b). "AMH: a new Framework to Design Adaptive Metaheuristics". In: *12th Metaheuristics International Conference, MIC, Barcelona, Spain*

Cited on page 47.

Blot, Aymeric, Marie-Eléonore Kessaci, and Laetitia Jourdan (2018b). "Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation". In: *Journal of Heuristics* 24.6, pp. 853–877

Cited on page 27.

Blot, Aymeric, Marie-Eléonore Kessaci, Laetitia Jourdan, and Patrick De Causmaecker (2018c). "Adaptive Multi-objective Local Search Algorithms for the Permutation Flowshop Scheduling Problem". In: *Proceedings of the 12th International Conference Learning and Intelligent Optimization, LION, Kalamata, Greece*. Vol. 11353. Lecture Notes in Computer Science. Springer, pp. 241–256

Cited on pages 27, 46 and 144.

Blot, Aymeric, Marie-Eléonore Kessaci, Laetitia Jourdan, and Holger H. Hoos (2019). "Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems". In: *Evolutionary Computation* 27.1, pp. 147–171

Cited on pages 27, 41 and 49.

Blot, Aymeric, Alexis Pernet, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Holger H. Hoos (2017c). "Automatically Configuring Multi-objective Local Search Using Multi-objective Optimisation". In: *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization, EMO, Münster, Germany*. Vol. 10173. Lecture Notes in Computer Science. Springer, pp. 61–76

Cited on pages 27, 40, 41, 49 and 53.

Branke, Jürgen and Jawad Asem Elomari (2012). "Meta-optimization for parameter tuning with a flexible computing budget". In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. ACM, pp. 1245–1252

Cited on pages 5 and 50.

Brockhoff, Dimo (2015). "A Bug in the Multiobjective Optimizer IBEA: Salutary Lessons for Code Release and a Performance Re-Assessment". In: *Evolutionary Multi-Criterion Optimization - EMO 2015, Guimarães, Portugal*, pp. 187–201

Cited on page 96.

Burke, Edmund K., Michel Gendreau, Matthew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu (2013). "Hyper-heuristics: a survey of the state of the art". In: *JORS* 64.12, pp. 1695–1724

Cited on page 5.

Burke, Edmund K., Matthew R. Hyde, and Graham Kendall (2012). "Grammatical Evolution of Local Search Heuristics". In: *IEEE Trans. Evolutionary Computation* 16.3, pp. 406–417

Cited on page 17.

Cáceres, Leslie Pérez and Thomas Stützle (2017). "Exploring variable neighborhood search for automatic algorithm configuration". In: *Electronic Notes in Discrete Mathematics* 58, pp. 167–174

Cited on page 82.

Cahon, Sébastien, Nordine Melab, and El-Ghazali Talbi (2004). "ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics". In: *J. Heuristics* 10.3, pp. 357–380

Cited on page 46.

Cai, Deng, Chiyuan Zhang, and Xiaofei He (2010). "Unsupervised feature selection for multi-cluster data". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 333–342

Cited on page 130.

Calhoun, Vince (2014). *MLSP 2014 Schizophrenia Classification Challenge*. `https://www.kaggle.com/c/mlsp-2014-mri`

Cited on page 136.

Cervante, Liam, Bing Xue, Mengjie Zhang, and Lin Shang (2012). "Binary particle swarm optimisation for feature selection: A filter based approach". In: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, pp. 1–8

Cited on page 130.

Chicano, Francisco, Fabio Daolio, Gabriela Ochoa, Sébastien Vérel, Marco Tomassini, and Enrique Alba (2012). "Local Optima Networks, Landscape Autocorrelation and Heuristic Search Performance". In: *Parallel Problem Solving from Nature - PPSN XII - 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part II*. Vol. 7492. Lecture Notes in Computer Science. Springer, pp. 337–347

Cited on page 7.

Conover, William Jay (1999). *Practical Nonparametric Statistics*. Wiley & Sons

Cited on page 22.

Consoli, Pietro A., Yi Mei, Leandro L. Minku, and Xin Yao (2016). "Dynamic selection of evolutionary operators based on online learning and fitness landscape analysis". In: *Soft Comput.* 20.10, pp. 3889–3914

Cited on page 7.

Czyzak, Piotr and Andrezej Jaszkiewicz (1996). "A multiobjective metaheuristic approach to the location of petrol stations by the capital budgeting model". In: *Control and*

Cybernetics 25.1, pp. 177–187

Cited on pages 28, 34, 37, 38, 42 and 43.

Czyzak, Piotr and Andrezej Jaszkiewicz (1998). "Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization". In: *Journal of Multi-Criteria Decision Analysis* 7.1, pp. 34–47

Cited on pages 28 and 36.

Daolio, Fabio, Arnaud Liefooghe, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka (2017). "Problem Features versus Algorithm Performance on Rugged Multiobjective Combinatorial Fitness Landscapes". In: *Evolutionary Computation* 25.4

Cited on page 8.

Daolio, Fabio, Sébastien Vérel, Gabriela Ochoa, and Marco Tomassini (2012). "Local optima networks and the performance of iterated local search". In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. ACM, pp. 369–376

Cited on page 7.

Deb, Kalyanmoy (2001). *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons

Cited on page 37.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Trans. Evolutionary Computation* 6.2, pp. 182–197

Cited on page 53.

Desport, Pierre, Matthieu Basseur, Adrien Goëffon, Frédéric Lardeux, and Frédéric Saubion (2015). "Empirical Analysis of Operators for Permutation Based Problems". In: *Learning and Intelligent Optimization - 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers*. Vol. 8994. Lecture Notes in Computer Science. Springer, pp. 137–150

Cited on page 5.

Ding, Jian-Ya, Shiji Song, Jatinder N.D. Gupta, Rui Zhang, Raymond Chiong, and Cheng Wu (2015). "An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem". In: *Applied Soft Computing* 30, pp. 604–613

Cited on pages 115, 118, 120, 121 and 126.

Doerr, Benjamin and Carola Doerr (2018). "Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices". In: *CoRR* abs/1804.05650

Cited on pages 5, 46, 74, 111, 113 and 144.

Dorigo, Marco and Mauro Birattari (2010). "Ant Colony Optimization". In: *Encyclopedia of Machine Learning*. Springer, pp. 36–39

Cited on page 132.

Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni (1996). "Ant system: optimization by a colony of cooperating agents". In: *IEEE Trans. Systems, Man, and Cybernetics, Part B* 26.1, pp. 29–41

Cited on page 2.

Dréo, Johann (2009). "Using performance fronts for parameter setting of stochastic metaheuristics". In: *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, Companion Material*. ACM, pp. 2197–2200

Cited on pages 5 and 50.

Drugan, Madalina M. and Dirk Thierens (2010). "Path-Guided Mutation for Stochastic Pareto Local Search Algorithms". In: *Parallel Problem Solving from Nature - PPSN XI, 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*. Vol. 6238. Lecture Notes in Computer Science. Springer, pp. 485–495

Cited on page 40.

Drugan, Madalina M. and Dirk Thierens (2012). "Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies". In: *J. Heuristics* 18.5, pp. 727–766

Cited on pages 32, 34, 38, 40, 42 and 43.

Du, Jianzhong and Joseph Y.-T. Leung (1990). "Minimizing Total Tardiness on One Machine is NP-Hard". In: *Math. Oper. Res.* 15.3, pp. 483–495

Cited on page 18.

Dubois-Lacoste, Jérémie, Manuel López-Ibáñez, and Thomas Stützle (2009). "Effective Hybrid Stochastic Local Search Algorithms for Biobjective Permutation Flowshop Scheduling". In: *Hybrid Metaheuristics, 6th International Workshop, HM 2009, Udine, Italy, October 16-17, 2009. Proceedings*. Vol. 5818. Lecture Notes in Computer Science. Springer, pp. 100–114

Cited on pages 18 and 19.

Dubois-Lacoste, Jérémie, Manuel López-Ibáñez, and Thomas Stützle (2011). "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems". In: *Computers & Operations Research* 38.8, pp. 1219–1236

Cited on page 31.

Dubois-Lacoste, Jérémie, Manuel López-Ibáñez, and Thomas Stützle (2012). "Pareto local search algorithms for anytime bi-objective optimization". In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, pp. 206–217

Cited on pages 32 and 37.

Dubois-Lacoste, Jérémie, Manuel López-Ibáñez, and Thomas Stützle (2015). "Anytime pareto local search". In: *European Journal of Operational Research* 243.2, pp. 369–385

Cited on pages 28, 32, 35 and 55.

Durillo, Juan José and Antonio J. Nebro (2011). "jMetal: A Java framework for multi-objective optimization". In: *Advances in Engineering Software* 42.10, pp. 760–771

Cited on page 47.

Duval, Béatrice, Jin-Kao Hao, and Jose Crispin Hernandez Hernandez (2009). "A memetic algorithm for gene selection and molecular classification of cancer". In: *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009*, pp. 201–208

Cited on page 130.

Eiben, Ágoston E., Robert Hinterding, and Zbigniew Michalewicz (1999). "Parameter control in evolutionary algorithms". In: *IEEE Transactions on Evolutionary Computation*

3.2, pp. 124–141

Cited on pages 5, 74 and 144.

Emmanouilidis, Christos, Andrew Hunter, and John MacIntyre (2000). "A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator". In: *Evolutionary Computation, 2000.* Vol. 1. IEEE, pp. 309–316

Cited on page 130.

Engrand, Philippe (1998). *A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management.* Tech. rep. Electricite de France

Cited on page 28.

Feo, Thomas A. and Mauricio G. C. Resende (1995). "Greedy Randomized Adaptive Search Procedures". In: *J. Global Optimization* 6.2, pp. 109–133

Cited on page 25.

Feo, Thomas A., Mauricio G. C. Resende, and Stuart H. Smith (1994). "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set". In: *Operations Research* 42.5, pp. 860–878

Cited on page 29.

Fialho, Álvaro, Luis Da Costa, Marc Schoenauer, and Michèle Sebag (2009). "Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms". In: *International Conference on Learning and Intelligent Optimization.* Springer, pp. 176–190

Cited on pages 5 and 74.

Fortemps, Philippe, Jacques Teghem, and Berthold Ulungu (1994). "Heuristics for multi-objective combinatorial optimization by simulated annealing". In: *XIth International Conference on MCDM*, pp. 1–6

Cited on pages 28, 34, 37, 38, 42 and 43.

Franzin, Alberto, Leslie Pérez Cáceres, and Thomas Stützle (2018). "Effect of transformations of numerical parameters in automatic algorithm configuration". In: *Optimization Letters* 12.8, pp. 1741–1753

Cited on page 83.

Franzin, Alberto and Thomas Stützle (2019). "Revisiting simulated annealing: A component-based analysis". In: *Computers & OR* 104, pp. 191–206

Cited on page 25.

Galván-López, Edgar, Riccardo Poli, Ahmed Kattan, Michael O'Neill, and Anthony Brabazon (2011). "Neutrality in evolutionary algorithms... What do we know?" In: *Evolving Systems* 2.3, pp. 145–163

Cited on page 101.

Gangadharan, Rajesh and Chandrasekharan Rajendran (1993). "Heuristic algorithms for scheduling in the no-wait flowshop". In: *International Journal of Production Economics* 32.3, pp. 285–290

Cited on page 115.

Garrett, Deon (2009). "Plateau Connection Structure and multiobjective metaheuristic performance". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, 18-21 May, 2009.* IEEE, pp. 1281–1288

Cited on page 7.

Garrett, Deon and Dipankar Dasgupta (2007). "Multiobjective Landscape Analysis and the Generalized Assignment Problem". In: *Learning and Intelligent Optimization, Second International Conference, LION 2007, Trento, Italy, December 8-12, 2007. Selected Papers*. Vol. 5313. Lecture Notes in Computer Science. Springer, pp. 110–124
Cited on page 7.

Geiger, Martin Josef (2008). "Randomised Variable Neighbourhood Search for Multi Objective Optimisation". In: *CoRR* abs/0809.0271
Cited on pages 29, 34, 42 and 43.

Gendreau, Michel and Jean-Yves Potvin (2010). *Handbook of metaheuristics*. Vol. 2. Springer
Cited on page 2.

Gheyas, Iffat A. and Leslie S. Smith (2010). "Feature subset selection in large dimensionality domains". In: *Pattern Recognition* 43.1, pp. 5–13
Cited on page 130.

Gilmore, Paul C. and Ralph E. Gomory (1964). "Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem". In: *Operations Research* 12.5, pp. 655–679
Cited on page 114.

Glover, Fred, Manuel Laguna, E Taillard, and Dominique de Werra (1993). *Tabu search*. Baltzer Basel
Cited on pages 2, 25 and 29.

Grabowski, Józef and Jarosław Pempera (2005). "Some local search algorithms for no-wait flow-shop problem with makespan criterion". In: *Computers & Operations Research* 32.8, pp. 2197–2212
Cited on page 115.

Guerra-Salcedo, Cesar and Darrell Whitley (1999). "Genetic Approach to Feature Selection for Ensemble Creation". In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), 13-17 July 1999, Orlando, Florida, USA*, pp. 236–243
Cited on page 136.

Guizzo, Giovani, Silvia R. Vergilio, Aurora T. R. Pozo, and Gian Mauricio Fritsche (2017). "A multi-objective and evolutionary hyper-heuristic applied to the Integration and Test Order Problem". In: *Appl. Soft Comput.* 56, pp. 331–344
Cited on page 5.

Guyon, Isabelle, Steve R. Gunn, Asa Ben-Hur, and Gideon Dror (2004). "Result Analysis of the NIPS 2003 Feature Selection Challenge". In: *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pp. 545–552
Cited on page 136.

Guyon, Isabelle, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh (2008). *Feature extraction: foundations and applications*. Vol. 207. Springer
Cited on page 136.

Hamdani, Tarek M., Jin-Myung Won, Adel M. Alimi, and Fakhri Karray (2007). "Multi-objective Feature Selection with NSGA II". In: *Adaptive and Natural Computing Algorithms, 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I*, pp. 240–247
Cited on page 130.

Hansen, Michael Pilegaard (1997). "Tabu search for multiobjective optimization: MOTS". In: *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, pp. 574–586

Cited on pages 28, 29, 34, 36, 37, 38, 42 and 43.

Hernando, Leticia, Fabio Daolio, Nadarajen Veerapen, and Gabriela Ochoa (2017). "Local Optima Networks of the Permutation Flowshop Scheduling Problem: Makespan vs. total flow time". In: *Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. IEEE, pp. 1964–1971

Cited on page 7.

Herrmann, Sebastian, Gabriela Ochoa, and Franz Rothlauf (2016). "Communities of Local Optima as Funnels in Fitness Landscapes". In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*. ACM, pp. 325–331

Cited on page 7.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press

Cited on page 2.

Hoos, Holger H. and Thomas Stützle (2004). *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann

Cited on pages 2, 14, 15 and 50.

Horn, Daniel, Karin Schork, and Tobias Wagner (2016). "Multi-objective Selection of Algorithm Portfolios: Experimental Validation". In: *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. Vol. 9921. Lecture Notes in Computer Science. Springer, pp. 421–430

Cited on page 5.

Humeau, Jérémie, Arnaud Liefooghe, El-Ghazali Talbi, and Sébastien Vérel (2013). "ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms". In: *J. Heuristics* 19.6, pp. 881–915

Cited on pages 17 and 46.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*. Vol. 6683. Springer, pp. 507–523

Cited on pages 4 and 50.

Hutter, Frank, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle (2009). "ParamILS: An Automatic Algorithm Configuration Framework". In: *J. Artif. Intell. Res.* 36, pp. 267–306

Cited on pages 4, 50, 54 and 69.

Inja, Maarten, Chiel Kooijman, Maarten de Waard, Diederik M. Roijers, and Shimon Whiteson (2014). "Queued Pareto Local Search for Multi-Objective Optimization". In: *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*. Vol. 8672. Lecture Notes in Computer Science. Springer, pp. 589–599

Cited on page 32.

Ishibuchi, Hisao and Tadahiko Murata (1996). "Multi-Objective Genetic Local Search Algorithm". In: *Proceedings of 1996 IEEE International Conference on Evolutionary Computation, Nayoya University, Japan, May 20-22, 1996*. IEEE, pp. 119–124

Cited on pages 28, 30, 34, 42 and 43.

Ishibuchi, Hisao, Noritaka Tsukamoto, and Yusuke Nojima (2008). "Evolutionary many-objective optimization: A short review". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008, June 1-6, 2008, Hong Kong, China*. IEEE, pp. 2419–2426

Cited on page 28.

Jaeggi, Daniel, Chris Asselin-Miller, Geoffrey T. Parks, Timoleon Kipouros, Theo Bell, and P. John Clarkson (2004). "Multi-objective Parallel Tabu Search". In: *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*. Vol. 3242. Lecture Notes in Computer Science. Springer, pp. 732–741

Cited on page 29.

Jaeggi, Daniel, Geoffrey T Parks, Timoleon Kipouros, and P John Clarkson (2008). "The development of a multi-objective Tabu Search algorithm for continuous optimisation problems". In: *European Journal of Operational Research* 185.3, pp. 1192–1212

Cited on page 29.

Jarboui, Bassem, Mansour Eddaly, and Patrick Siarry (2010). "A hybrid genetic algorithm for solving no-wait flowshop scheduling problems". In: *The International Journal of Advanced Manufacturing Technology* 54.9-12, pp. 1129–1143

Cited on page 115.

Jaszkiewicz, Andrezej (2002). "Genetic local search for multi-objective combinatorial optimization". In: *European Journal of Operational Research* 137.1, pp. 50–71

Cited on pages 28 and 30.

Jones, Terry (1995). "Evolutionary algorithms, fitness landscapes and search". PhD thesis. University of New Mexico

Cited on page 6.

Jones, Terry and Stephanie Forrest (1995). "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms". In: *Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 15-19, 1995*. Morgan Kaufmann, pp. 184–192

Cited on page 6.

Kadioglu, Serdar, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney (2010). "ISAC - Instance-Specific Algorithm Configuration". In: *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*. Vol. 215. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 751–756

Cited on page 5.

Karafotias, Giorgos, Mark Hoogendoorn, and Ágoston E. Eiben (2015). "Parameter Control in Evolutionary Algorithms: Trends and Challenges". In: *IEEE Transactions on Evolutionary Computation* 19.2, pp. 167–187

Cited on pages 5, 46 and 74.

Kauffman, Stuart (1993). *The Origins of Order : Self-Organization and Selection in Evolution*. Oxford University Press

Cited on page 7.

Kerschke, Pascal, Holger H. Hoos, Frank Neumann, and Heike Trautmann (2019). "Automated Algorithm Selection: Survey and Perspectives". In: *Evolutionary Computation* 27.1, pp. 3–45

Cited on page 5.

Kerschke, Pascal and Heike Trautmann (2019). "Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning". In: *Evolutionary Computation* 27.1, pp. 99–127

Cited on pages 112 and 145.

Kessaci-Marmion, Marie-Eléonore, Clarisse Dhaenens, and Jérémie Humeau (2017). "Neutral Neighbors in Bi-objective Optimization: Distribution of the Most Promising for Permutation Problems". In: *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization, EMO 2017, Münster, Germany*. Vol. 10173. Lecture Notes in Computer Science. Springer, pp. 344–358

Cited on page 87.

Kirkpatrick, Scott, C Daniel Gelatt, Mario P Vecchi, et al. (1983). "Optimization by simulated annealing". In: *Science* 220.4598, pp. 671–680

Cited on pages 2, 14, 15 and 28.

Knowles, Joshua and David Corne (1999). "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation". In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 1. IEEE, pp. 98–105

Cited on pages 28, 30, 31, 34, 36, 42 and 43.

Knowles, Joshua and David Corne (2000a). "Approximating the nondominated front using the Pareto archived evolution strategy". In: *Evolutionary Computation* 8.2, pp. 149–172

Cited on pages 30, 34, 42 and 43.

Knowles, Joshua and David Corne (2000b). "M-PAES: A memetic algorithm for multiobjective optimization". In: *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. Vol. 1. IEEE, pp. 325–332

Cited on page 30.

Knowles, Joshua and David Corne (2002). "On metrics for comparing nondominated sets". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*. Vol. 1, pp. 711–716

Cited on page 52.

Knowles, Joshua and David Corne (2003). "Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem". In: *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 2003, Proceedings*, ed. by Carlos Fonseca et al. LNCS 2632, pp. 295–310

Cited on pages 7 and 56.

Kochenberger, Gary A., Fred W. Glover, Bahram Alidaee, and César Rego (2004). "A unified modeling and solution framework for combinatorial optimization problems". In: *OR Spectrum* 26.2, pp. 237–250

Cited on page 25.

Kohavi, Ron and George H. John (1997). "Wrappers for Feature Subset Selection". In: *Artif. Intell.* 97.1-2, pp. 273–324

Cited on page 129.

Kotthoff, Lars (2014). "Algorithm Selection for Combinatorial Search Problems: A Survey". In: *AI Magazine* 35.3, pp. 48–60

Cited on page 5.

Laha, Dipak and Uday K. Chakraborty (2008). "A constructive heuristic for minimizing makespan in no-wait flow shop scheduling". In: *The International Journal of Advanced Manufacturing Technology* 41.1-2, pp. 97–109

Cited on page 115.

Li, Wenwen, Ender Ozcan, and Robert John (2019). "A Learning Automata-Based Multiobjective Hyper-Heuristic". In: *IEEE Trans. Evolutionary Computation* 23.1, pp. 59–73

Cited on page 5.

Liefooghe, Arnaud, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka (2017a). "A Fitness Landscape Analysis of Pareto Local Search on Bi-objective Permutation Flowshop Scheduling Problems". In: *Evolutionary Multi-Criterion Optimization - 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*. Vol. 10173. Lecture Notes in Computer Science. Springer, pp. 422–437

Cited on page 8.

Liefooghe, Arnaud, Bilel Derbel, Sébastien Vérel, Hernán E. Aguirre, and Kiyoshi Tanaka (2017b). "Towards Landscape-Aware Automatic Algorithm Configuration: Preliminary Experiments on Neutral and Rugged Landscapes". In: *Evolutionary Computation in Combinatorial Optimization - 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings*. Vol. 10197. Lecture Notes in Computer Science, pp. 215–232

Cited on page 7.

Liefooghe, Arnaud, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and El-Ghazali Talbi (2012). "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems". In: *Journal of Heuristics* 18.2, pp. 317–352

Cited on pages 28, 32, 34, 35, 36, 41, 42 and 43.

Liefooghe, Arnaud, Laetitia Jourdan, and El-Ghazali Talbi (2011). "A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO". In: *European Journal of Operational Research* 209.2, pp. 104–112

Cited on page 46.

Lin, Shih-Wei and Kuo-Ching Ying (Oct. 2016). "Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics". In: *Omega* 64, pp. 115–125

Cited on page 115.

Lindauer, Marius Thomas, Holger H. Hoos, Frank Hutter, and Torsten Schaub (2015). "AutoFolio: An Automatically Configured Algorithm Selector". In: *J. Artif. Intell. Res.* 53, pp. 745–778

Cited on page 144.

Lindauer, Marius, Rolf-David Bergdoll, and Frank Hutter (2016). "An Empirical Study of Per-instance Algorithm Scheduling". In: *Learning and Intelligent Optimization - 10th*

International Conference, LION 10, Ischia, Italy, May 29 - June 1, 2016, Revised Selected Papers. Vol. 10079. Lecture Notes in Computer Science. Springer, pp. 253–259

Cited on pages 5 and 77.

Long, Nanye, Daniel Gianola, and Guilherme J. M. Rosa (2007). "Machine learning classification procedure for selecting SNPs in genomic selection: application to early mortality in broilers". In: *Journal of animal breeding and genetics* 124.6, pp. 377–389

Cited on page 130.

López-Ibáñez, Manuel and Christian Blum (2010). "Beam-ACO for the travelling salesman problem with time windows". In: *Computers & OR* 37.9, pp. 1570–1583

Cited on page 25.

López-Ibáñez, Manuel, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari (2016). "The irace package: Iterated Racing for Automatic Algorithm Configuration". In: *Operations Research Perspectives* 3, pp. 43–58

Cited on pages 4 and 50.

López-Ibáñez, Manuel, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari (2011). *The irace package, Iterated Race for Automatic Algorithm Configuration*. Tech. rep. TR/IRIDIA/2011-004. IRIDIA, Université Libre de Bruxelles, Belgium

Cited on page 18.

López-Ibáñez, Manuel, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle (2013). "Automatic Design of a Hybrid Iterated Local Search for the Multi-Mode Resource-Constrained Multi-Project Scheduling Problem". In: *Proceedings of the 6th Multidisciplinary International Conference on Scheduling : Theory and Applications, MISTA, Ghent, Belgium*, pp. 820–825

Cited on page 13.

López-Ibáñez, Manuel, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle (2014). "A template for designing single-solution hybrid metaheuristics". In: *Companion Material Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Vancouver, BC, Canada*, pp. 1423–1426

Cited on page 13.

Lourenço, Helena, Olivier Martin, and Thomas Stützle (2010). "Iterated Local Search: Framework and Applications". In: *Handbook of Metaheuristics*. Vol. 2. Springer, pp. 363–397

Cited on pages 2, 14, 15, 40 and 50.

Lust, Thibaut and Jacques Teghem (2010). "Two-phase Pareto local search for the biobjective traveling salesman problem". In: *Journal of Heuristics* 16.3, pp. 475–510

Cited on page 31.

Manderick, Bernard, Mark K. de Weger, and Piet Spiessens (1991). "The Genetic Algorithm and the Structure of the Fitness Landscape". In: *Proceedings of the 4th International Conference on Genetic Algorithms, San Diego, CA, USA, July 1991*. Morgan Kaufmann, pp. 143–150

Cited on page 6.

Marmion, Marie-Eléonore, Hernán E. Aguirre, Clarisse Dhaenens, Laetitia Jourdan, and Kiyoshi Tanaka (2016). "Multi-objective Neutral Neighbors': What could be the definition(s)?" In: *Proceedings of the Genetic and Evolutionary Computation Conference,*

GECCO, Denver, USA. ACM, pp. 349–356

Cited on page 87.

Marmion, Marie-Eléonore, Aymeric Blot, Laetitia Jourdan, and Clarisse Dhaenens (2013a). "Neutrality in the Graph Coloring Problem". In: *Proceedings of the 7th International Conference on Learning and Intelligent Optimization, LION, Catania, Italy*. Vol. 7997. Lecture Notes in Computer Science. Springer, pp. 125–130

Cited on page 6.

Marmion, Marie-Eléonore, Clarisse Dhaenens, Laetitia Jourdan, Arnaud Liefooghe, and Sébastien Verel (2011a). "NILS: A Neutrality-Based Iterated Local Search and Its Application to Flowshop Scheduling". In: *Proceedings of the 11th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP, Torino, Italy*. Vol. 6622. Lecture Notes in Computer Science. Springer, pp. 191–202

Cited on pages 7, 88, 101 and 111.

Marmion, Marie-Eléonore, Clarisse Dhaenens, Laetitia Jourdan, Arnaud Liefooghe, and Sébastien Verel (2011b). "On the Neutrality of Flowshop Scheduling Fitness Landscapes". In: *Proceedings of the 5th International Conference on Learning and Intelligent Optimization, LION, Rome, Italy*. Vol. 6683. Lecture Notes in Computer Science. Springer, pp. 238–252

Cited on pages 6 and 97.

Marmion, Marie-Eléonore, Jérémie Humeau, Laetitia Jourdan, and Clarisse Dhaenens (2013b). "Fitness Landscape Analysis and Metaheuristics Efficiency". In: *J. Math. Model. Algorithms* 12.1, pp. 3–26

Cited on page 7.

Marmion, Marie-Eléonore, Franco Mascia, Manuel López-Ibáñez, and Thomas Stützle (2013c). "Automatic Design of Hybrid Stochastic Local Search Algorithms". In: *Proceedings of the 8th International Workshop Hybrid Metaheuristics, HM, Ischia, Italy*. Vol. 7919. Lecture Notes in Computer Science. Springer, pp. 144–158

Cited on page 13.

Martí, Rafael, Vicente Campos, Mauricio G. C. Resende, and Abraham Duarte (2015). "Multiobjective GRASP with Path Relinking". In: *European Journal of Operational Research* 240.1, pp. 54–71

Cited on page 29.

Mascia, Franco, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Marie-Eléonore Marmion, and Thomas Stützle (2014). "Algorithm Comparison by Automatically Configurable Stochastic Local Search Frameworks: A Case Study Using Flow-Shop Scheduling Problems". In: *Proceedings of the 9th International Workshop Hybrid Metaheuristics, HM, Hamburg, Germany*. Vol. 8457. Lecture Notes in Computer Science. Springer, pp. 30–44

Cited on page 13.

Mascia, Franco, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle (2013). "From Grammars to Parameters: Automatic Iterated Greedy Design for the Permutation Flow-Shop Problem with Weighted Tardiness". In: *Learning and Intelligent Optimization - 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers*. Vol. 7997. Lecture Notes in Computer Science. Springer,

pp. 321–334

Cited on page 18.

McKay, Robert I., Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'Neill (2010). "Grammar-based Genetic Programming: a survey". In: *Genetic Programming and Evolvable Machines* 11.3-4, pp. 365–396

Cited on pages 16 and 17.

Merz, Peter (2004). "Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms". In: *Evolutionary Computation* 12.3, pp. 303–325

Cited on page 7.

Minella, Gerardo, Rubén Ruiz, and Michele Ciavotta (2008). "A Review and Evaluation of Multiobjective Algorithms for the Flowshop Scheduling Problem". In: *INFORMS Journal on Computing* 20.3, pp. 451–471

Cited on page 20.

Mladenović, Nenad and Pierre Hansen (1997). "Variable neighborhood search". In: *Computers & operations research* 24.11, pp. 1097–1100

Cited on pages 2, 14, 15, 29 and 115.

Moalic, Laurent, Alexandre Caminada, and Sid Lamrous (2013). "A fast local search approach for multiobjective problems". In: *International Conference on Learning and Intelligent Optimization*. Springer, pp. 294–298

Cited on pages 32, 34, 42 and 43.

Molina, Julián, Manuel Laguna, Rafael Martı, and Rafael Caballero (2005). "SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization". In: *INFORMS Journal on Computing* 19.1, pp. 91–100

Cited on page 29.

Moslehi, Ghasem and Mehdi Mahnam (2011). "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search". In: *International Journal of Production Economics* 129.1, pp. 14–22

Cited on page 31.

Mousin, Lucien, Laetitia Jourdan, Marie-Eléonore Kessaci-Marmion, and Clarisse Dhaenens (2016). "Feature Selection Using Tabu Search with Learning Memory: Learning Tabu Search". In: *Proceedings of the 10th International Conference on Learning and Intelligent Optimization, LION, Ischia, Italy*. Vol. 10079. Lecture Notes in Computer Science. Springer, pp. 141–156

Cited on pages 7 and 113.

Mousin, Lucien, Marie-Eléonore Kessaci, and Clarisse Dhaenens (2017). "A New Constructive Heuristic for the No-Wait Flowshop Scheduling Problem". In: *Proceedings of the 11th International Conference Learning and Intelligent Optimization, LION, Nizhny Novgorod, Russia*. Vol. 10556. Lecture Notes in Computer Science. Springer, pp. 196–209

Cited on pages 115 and 120.

Mousin, Lucien, Marie-Eléonore Kessaci, and Clarisse Dhaenens (2019). "Exploiting Promising Sub-Sequences of Jobs to solve the No-Wait Flowshop Scheduling Problem". In: *CoRR* abs/1903.09035

Cited on pages 7 and 113.

Murata, Tadahiko, Hisao Ishibuchi, and Mitsuo Gen (2000). "Cellular genetic local search for multi-objective optimization". In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., pp. 307–314

Cited on page 30.

Nawaz, Muhammad, E Emory Enscore, and Inyong Ham (1983). "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem". In: *Omega* 11.1, pp. 91–95

Cited on pages 19 and 115.

Nebro, Antonio J., Juan José Durillo, and Matthieu Vergne (2015). "Redesigning the jMetal Multi-Objective Optimization Framework". In: *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*. ACM, pp. 1093–1100

Cited on page 47.

Ochoa, Gabriela, Marco Tomassini, Sébastien Vérel, and Christian Darabos (2008). "A study of NK landscapes' basins and local optima networks". In: *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*. ACM, pp. 555–562

Cited on page 6.

Ochoa, Gabriela and Nadarajen Veerapen (2016). "Deconstructing the Big Valley Search Space Hypothesis". In: *Evolutionary Computation in Combinatorial Optimization - 16th European Conference, EvoCOP 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings*, pp. 58–73

Cited on pages 6 and 7.

Ochoa, Gabriela, Nadarajen Veerapen, Darrell Whitley, and Edmund K. Burke (2015). "The Multi-Funnel Structure of TSP Fitness Landscapes: A Visual Exploration". In: *Artificial Evolution - 12th International Conference, Evolution Artificielle, EA 2015, Lyon, France, October 26-28, 2015. Revised Selected Papers*. Vol. 9554. Lecture Notes in Computer Science. Springer, pp. 1–13

Cited on pages 7 and 141.

Okabe, Tatsuya, Yaochu Jin, and Bernhard Sendhoff (2003). "A critical survey of performance indices for multi-objective optimisation". In: *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.* Vol. 2, pp. 878–885

Cited on page 52.

Oliveira, Luiz S, Marisa Morita, and Robert Sabourin (2006). "Feature selection for ensembles using the multi-objective optimization approach". In: *Multi-Objective Machine Learning*. Springer, pp. 49–74

Cited on page 130.

Pageau, Camille, Aymeric Blot, Holger H. Hoos, Marie-Eléonore Kessaci, and Laetitia Jourdan (2019). "Configuration of a Dynamic MOLS Algorithm for Bi-objective Flowshop Scheduling". In: *Proceedings of the 10th International Conference on Evolutionary Multi-Criterion Optimization, EMO, East Lansing, USA*. Vol. 11411. Lecture Notes in Computer Science. Springer, pp. 565–577

Cited on pages 27, 41 and 49.

Pagnozzi, Federico and Thomas Stützle (2019). "Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems". In: *European Journal of*

Operational Research 276.2, pp. 409–421

Cited on page 25.

Pan, Quan-Ke, M. Fatih Tasgetiren, and Yun-Chia Liang (2008). "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem". In: *Computers & Operations Research* 35.9, pp. 2807–2839

Cited on page 115.

Papadimitriou, Christos H. and Kenneth Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall

Cited on page 14.

Paquete, Luis, Marco Chiarandini, and Thomas Stützle (2004). "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study". In: *Metaheuristics for Multiobjective Optimisation*. Springer, pp. 177–199

Cited on pages 31, 34, 36, 42 and 43.

Paquete, Luis and Thomas Stützle (2003). "A two-phase local search for the biobjective traveling salesman problem". In: *Evolutionary Multi-Criterion Optimization*. Springer, pp. 69–69

Cited on page 31.

Pavelski, Lucas M., Marie-Eléonore Kessaci, and Myriam R. Delgado (2019). "Meta-learning on Flowshop using Fitness Landscape Analysis". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, Praha, Czech Republic*. ACM, pp. –

Cited on pages 112 and 145.

Pushak, Yasha and Holger H. Hoos (2018). "Algorithm Configuration Landscapes: - More Benign Than Expected?" In: *Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II*. Vol. 11102. Lecture Notes in Computer Science. Springer, pp. 271–283

Cited on page 143.

Qian, B., L. Wang, R. Hu, D.X. Huang, and X. Wang (2009). "A DE-based approach to no-wait flow-shop scheduling". In: *Computers & Industrial Engineering* 57.3, pp. 787–805

Cited on page 115.

Rajendran, Chandrasekharan (1994). "A No-Wait Flowshop Scheduling Heuristic to Minimize Makespan". In: *Journal of the Operational Research Society* 45.4, pp. 472–478

Cited on page 115.

Rice, John R. (1976). "The Algorithm Selection Problem". In: *Advances in Computers* 15, pp. 65–118

Cited on page 5.

Riquelme, Nery, Christian von Lücken, and Benjamın Barán (2015). "Performance metrics in multi-objective optimization". In: *Proceedings of the 2015 Latin American Computing Conference*, pp. 1–11

Cited on page 53.

Röck, Hans (1984). "The Three-Machine No-Wait Flow Shop is NP-Complete". In: *Journal of the ACM* 31.2, pp. 336–345

Cited on page 114.

Ruiz, Rubén and Thomas Stützle (2007). "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem". In: *European Journal of Operational Research* 177.3, pp. 2033–2049

Cited on pages 14, 15, 115 and 118.

Samarghandi, Hamed and Tarek Y. ElMekkawy (2012). "A meta-heuristic approach for solving the no-wait flow-shop problem". In: *International Journal of Production Research* 50.24, pp. 7313–7326

Cited on page 115.

Schindl, David and Nicolas Zufferey (2013). "Solution Methods for Fuel Supply of Trains". In: *INFOR: Information Systems and Operational Research* 51.1, pp. 23–30

Cited on pages 131 and 132.

Schölkopf, Bernhard and Alex Smola (1998). "Support Vector Machines". In: *Encyclopedia of Biostatistics*

Cited on pages 130 and 135.

Serafini, Paolo (1994). "Simulated annealing for multi objective optimization problems". In: *Multiple Criteria Decision Making*. Springer, pp. 283–292

Cited on pages 28, 34, 37, 38, 42 and 43.

Suman, Balram (2003). "Simulated annealing-based multiobjective algorithms and their application for system reliability". In: *Engineering Optimization* 35.4, pp. 391–416

Cited on page 29.

Suman, Balram and Prabhat Kumar (2006). "A survey of simulated annealing as a tool for single and multiobjective optimization". In: *Journal of the Operational Research Society* 57.10, pp. 1143–1160

Cited on page 29.

Suppapitnarm, Apichart and Geoff Parks (1999). "Simulated annealing: an alternative approach to true multiobjective optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*. Morgan Kaufmann Publishers, pp. 406–407

Cited on page 28.

Suresh, RK and KM Mohanasundaram (2004). "Pareto archived simulated annealing for permutation flow shop scheduling with multiple objectives". In: *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*. Vol. 2. IEEE, pp. 712–717

Cited on page 29.

Taillard, Eric (1993). "Benchmarks for basic scheduling problems". In: *European Journal of Operational Research* 64.2, pp. 278–285

Cited on pages 20 and 55.

Talbi, El-Ghazali, Malek Rahoual, Mohamed Hakim Mabed, and Clarisse Dhaenens (2001). "A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop". In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, pp. 416–428

Cited on pages 28, 30, 31, 34, 42 and 43.

Tan, Choo Jun, Chee Peng Lim, and Yu–N Cheah (2014). "A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models". In: *Neurocomputing* 125, pp. 217–228

Cited on page 130.

Tari, Sara, Matthieu Basseur, and Adrien Goëffon (2017). "Sampled Walk and Binary Fitness Landscapes Exploration". In: *Artificial Evolution - 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25-27, 2017, Revised Selected Papers*. Vol. 10764. Lecture Notes in Computer Science. Springer, pp. 47–57

Cited on page 7.

Tari, Sara, Matthieu Basseur, and Adrien Goëffon (2018). "Worst Improvement Based Iterated Local Search". In: *Evolutionary Computation in Combinatorial Optimization - 18th European Conference, EvoCOP 2018, Parma, Italy, April 4-6, 2018, Proceedings*. Vol. 10782. Lecture Notes in Computer Science. Springer, pp. 50–66

Cited on page 7.

Thierens, Dirk (2005). "An adaptive pursuit strategy for allocating operator probabilities". In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, pp. 1539–1546

Cited on pages 5 and 74.

Tollo, Giacomo di, Frédéric Lardeux, Jorge Maturana, and Frédéric Saubion (2015). "An experimental study of adaptive control for evolutionary algorithms". In: *Appl. Soft Comput.* 35, pp. 359–372

Cited on page 5.

Tricoire, Fabien (2012). "Multi-directional local search". In: *Computers & Operations Research* 39.12, pp. 3089–3101

Cited on page 32.

Ulungu, Berthold, Jacques Teghem, and Philippe Fortemps (1995). "Heuristic for multi-objective combinatorial optimization problems by simulated annealing". In: *MCDM: Theory and Applications*

Cited on page 28.

Ulungu, Berthold, Jacques Teghem, Philippe Fortemps, and Daniel Tuyttens (1999). "MOSA method: a tool for solving multiobjective combinatorial optimization problems". In: *Journal of Multi-Criteria Decision Analysis* 8.4, p. 221

Cited on pages 28 and 36.

Vassilev, Vesselin K., Terence C. Fogarty, and Julian F. Miller (2000). "Information Characteristics and the Structure of Landscapes". In: *Evolutionary Computation* 8.1, pp. 31–60

Cited on page 6.

Veerapen, Nadarajen, Jorge Maturana, and Frédéric Saubion (2012). "An exploration-exploitation compromise-based adaptive operator selection for local search". In: *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*. ACM, pp. 1277–1284

Cited on page 5.

Veerapen, Nadarajen and Gabriela Ochoa (2018). "Visualising the global structure of search landscapes: genetic improvement as a case study". In: *Genetic Programming and Evolvable Machines* 19.3, pp. 317–349

Cited on pages 7 and 141.

Vérel, Sébastien, Philippe Collard, and Manuel Clergue (2004). "Scuba search: when selection meets innovation". In: *Proceedings of the IEEE Congress on Evolutionary Com-*

putation, CEC 2004, 19-23 June 2004, Portland, OR, USA. IEEE, pp. 924–931
Cited on pages 88 and 111.

Vérel, Sébastien, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini (2018). "Sampling Local Optima Networks of Large Combinatorial Search Spaces: The QAP Case". In: *Parallel Problem Solving from Nature - PPSN XV - 15th International Conference, Coimbra, Portugal, September 8-12, 2018, Proceedings, Part II*. Vol. 11102. Lecture Notes in Computer Science. Springer, pp. 257–268
Cited on page 7.

Vérel, Sébastien, Arnaud Liefooghe, Laetitia Jourdan, and Clarisse Dhaenens (2013). "On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives". In: *European Journal of Operational Research* 227.2, pp. 331–342
Cited on page 8.

Vianna, Dalessandro Soares and José Elias Claudio Arroyo (2004). "A GRASP Algorithm for the Multi-Objective Knapsack Problem". In: *XXIV International Conference of the Chilean Computer Science Society (SCCC 2004), 11-12 November 2004, Arica, Chile*. IEEE Computer Society, pp. 69–75
Cited on page 29.

Weinberger, Edward D. (1990). "Correlated and uncorrelated fitness landscapes and how to tell the difference". In: *Biological Cybernetics* 63.5, pp. 325–336
Cited on page 6.

Wismer, David A. (1972). "Solution of the Flowshop-Scheduling Problem with No Intermediate Queues". In: *Operations Research* 20.3, pp. 689–697
Cited on page 114.

Wright, Sewall (1932). "The roles of mutation, inbreeding, crossbreeding and selection in evolution". In: *Proceedings of the Sixth International Congress on Genetics*. Vol. 1
Cited on page 6.

Xu, Lin, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown (2008). "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *J. Artif. Intell. Res.* 32, pp. 565–606
Cited on page 5.

Xue, Bing, Mengjie Zhang, and Will N Browne (2013). "Particle swarm optimization for feature selection in classification: A multi-objective approach". In: *Cybernetics, IEEE Transactions on* 43.6, pp. 1656–1671
Cited on page 130.

Yang, Jihoon and Vasant Honavar (1998). "Feature subset selection using a genetic algorithm". In: *Feature extraction, construction and selection*. Springer, pp. 117–136
Cited on page 130.

Zhang, Tiantian, Michael Georgiopoulos, and Georgios C. Anagnostopoulos (2013). "S-Race: a multi-objective racing algorithm". In: *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*. ACM, pp. 1565–1572
Cited on page 5.

Zhang, Tiantian, Michael Georgiopoulos, and Georgios C. Anagnostopoulos (2015). "SPRINT Multi-Objective Model Racing". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*. ACM, pp. 1383–

1390

Cited on page 50.

Zhu, Zexuan, Yew-Soon Ong, and Manoranjan Dash (2007). "Markov blanket-embedded genetic algorithm for gene selection". In: *Pattern Recognition* 40.11, pp. 3236–3248

Cited on page 136.

Zitzler, Eckart and Lothar Thiele (1999). "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach". In: *IEEE Trans. Evolutionary Computation* 3.4, pp. 257–271

Cited on pages 32 and 52.

Zitzler, Eckart, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca (2003). "Performance assessment of multiobjective optimizers: an analysis and review". In: *IEEE Trans. Evolutionary Computation* 7.2, pp. 117–132

Cited on pages 3 and 52.

Résumé

Les métaheuristiques sont des algorithmes génériques et flexibles capables de s'adapter à tout type de problème d'optimisation grâce à la variété des stratégies algorithmiques et leurs propres valeurs de paramètres. Premièrement, la généricité de la conception peut aussi être complétée par des mécanismes ou des heuristiques dépendant du problème. Deuxièmement, bien qu'il soit largement admis qu'aucun algorithme ne domine tous les autres sur toutes les instances du problème, une métaheuristique doit être finement paramétrée pour bien fonctionner. Par conséquent, un processus d'apprentissage peut être utilisé pour concevoir une métaheuristique adaptée au problème à traiter.
Dans nos travaux, nous nous intéressons à deux façons différentes d'aborder la conception basée sur la connaissance : la configuration automatique d'algorithmes et l'analyse de paysage. La première partie traite de la configuration automatique des algorithmes de recherche locale mono-objectif et multi-objectif et la seconde traite de la caractérisation des paysages multi-objectifs et de l'exploitation des caractéristiques du problème pour concevoir des algorithmes de recherche locale.
Ces deux sujets, apparemment indépendants, ont, très récemment, commencé à se rejoindre et diverses perspectives seront données dans ce sens.

Abstract

Metaheuristics are generic and flexible algorithms able to adapt to any kind of optimization problems. Both genericity and flexibility are given by the available algorithmic strategies and their own parameters values. First, the generic design can be completed with problem-dependent mechanisms or heuristics. Second, while it is widely admitted that no single algorithm dominates all others on all problem instances, the metaheuristics have to be finely parameterized to perform well. Therefore learning process should be used to design metaheuristics adapted to the given problem instances.
In this work, we will be interested in two different ways to address knowledge-based design of metaheuristics: automatic algorithm configuration and fitness landscape analysis. The first part deals with the automatic configuration of single-objective and multi-objective stochastic local search algorithms and, the second one deals with the characterization of multi-objective landscapes and the exploitation of features into stochastic local search algorithms.
These two, apparently independent topics, have, very recently, started to meet each other and various perspectives will be given in that sense.