Automatic Configuration of Bi-objective Optimisation Algorithms: Impact of Correlation between Objectives

Aymeric Blot
CRIStAL CNRS, UMR 9189
Université de Lille,
Lille, France
aymeric.blot@univ-lille.fr

Holger H. Hoos
LIACS,
Leiden University
Leiden, Netherlands
hh@liacs.nl

Marie-Éléonore Kessaci
CRIStAL CNRS, UMR 9189
Université de Lille,
Lille, France
mkessaci@univ-lille.fr

Laetitia Jourdan
CRIStAL CNRS, UMR 9189
Université de Lille,
Lille, France
laetitia.jourdan@univ-lille.fr

***Abstract*—Multi-objective optimisation algorithms expose various parameters that have to be tuned in order to be efficient. Moreover, in multi-objective optimisation, the correlation between objective functions is known to affect search space structure and algorithm performance. Considering the recent success of automatic algorithm configuration (AAC) techniques for the design of multi-objective optimisation algorithms, this raises two interesting questions: what is the impact of correlation between optimisation objectives on (1) the efficacy of different AAC approaches and (2) on the optimised algorithm designs obtained from these automated approaches? In this work, we study these questions for multi-objective local search algorithms (MOLS) for three well-known bi-objective permutation problems, using two single-objective AAC approaches and one multi-objective approach. Our empirical results clearly show that overall, multi-objective AAC is the most effective approach for the automatic configuration of the highly parametric MOLS framework, and that there is no systematic impact of the degree of correlation on the relative performance of the three AAC approaches. We also find that the best-performing configurations differ, depending on the correlation between objectives and the size of the problem instances to be solved, providing further evidence for the usefulness of automatic configuration of multi-objective optimisation algorithms.**

***Index Terms*—Automatic algorithm configuration, Multi-objective optimisation, Combinatorial optimisation, Heuristic algorithms**

I. Introduction

Multi-objective optimisation involves two or more objective functions to be optimised simultaneously. The degree to which these objectives conflict with each other can vary and is known to have an impact on the difficulty of multi-objective optimisation problems. Indeed, Verel et al. [23] have demonstrated the impact of the correlation between objectives on the Pareto optimal set for optimising a tunable problem-independent multi-objective model. Metaheuristics, including local search and bio-inspired algorithms, are widely used for solving challenging multi-objective optimisation problems. In this context, Kessaci-Marmion et al. [14] showed that the distribution of neighbours of given solutions for classical multi-objective permutation problems differs, depending on the degree of correlation between objectives.

Over the last decade, it has become increasingly common to leverage automatic algorithm configuration (AAC) techniques in the design and application of high-performance heuristic algorithms, and in particular, of metaheuristic optimisation procedures. More recently, AAC approaches have been successfully applied to multi-objective optimisation algorithms, and it has been shown that multi-objective AAC techniques, which simultaneously optimise multiple performance metrics, are particularly well-suited in this context [4]. However, it is unclear how the correlation between optimisation objectives affects AAC for multi-objective optimisation algorithms. In this work, we investigate this issue, considering multi-objective local search algorithms [8], [9], [17] for three classical bi-objective permutation problems: the travelling salesman problem (TSP), the quadratic assignment problem (QAP) and the permutation flowshop scheduling problem (PFSP) with different degrees of correlation between objectives. Specifically, we address three currently open questions: (1) Does the performance of the AAC approaches change according to the degree of correlation between objectives? (2) How are the optimised configurations produced by AAC affected by correlation? (3) Do these effects vary with problem instance size?

The remainder of this paper is organised as follows. Section II provides some basic definitions and introduces the flexible multi-objective local search framework used in our study. Section III describes the three classical bi-objective permutation problems we considered. Section IV details three automated algorithm configuration protocols we used in our experiments. Results from our experiments are presented in Section V, and Section VI provides some conclusions and an outlook on future work.

II. Multi-objective Local Search Algorithms

In this section, we briefly outline basic notions from multi-objective combinatorial optimisation, before introducing the MOLS algorithm framework and its component strategies used in this work.

A. Multi-objective Combinatorial Optimisation

Multi-objective optimisation problems arise when multiple criteria have to be optimised simultaneously. To compare two solutions s_1 and s_2 of a given problem, the notion of *Pareto dominance* is used: s_1 is said to *dominate* s_2 if, and only if, s_1 is better or equal to s_2 according to all criteria, and there is at least one criterion according to which s_1 is strictly better than s_2. If neither s_1 dominates s_2 nor s_2 dominates s_1, both solutions are called *incomparable*. Assessing and optimising solutions based on Pareto dominance has advantages over alternatives, such as weighted aggregation of the given objectives into a single (scalar) measure of solution quality or ordering objectives lexicographically.

Solving multi-objective combinatorial optimisation problems based on Pareto dominance involves finding *Pareto sets*, i.e., sets of incomparable solutions. Such a set is called a *Pareto optimum* with respect to a set of permissible changes to each of its elements if no such change results in any element of the set to be dominated.

In order to assess the quality of Pareto sets of solutions, various indicators have been proposed [15], [19], [25]. These indicators are used to characterise the performance of dominance-based multi-objective algorithms in terms of convergence, distribution or cardinality. Since no single quality indicator is able to capture all of these properties, it is recommended to consider multiple indicators, preferably ones that complement each other, in order to assess the performance of multi-objective algorithms [24].

In this work, we use a combination of two well-known indicators: hypervolume [25] and a complementary spread measure. These were chosen in light of their common usage, their complementarity, and the additional requirements for unary indicators that do not require reference sets, which makes them suitable for 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 optimisation [21]. Assuming normalised objective values in $[0, 1]$, unary hypervolume measures the volume between a given Pareto set of solutions and the point $(1, 1)$. While HV is primarily a convergence indicator, it also captures information about the diversity of the set of solutions. In our experiments, we minimise $1 - HV$ by means of automated algorithm configuration (described later).

Our second indicator is a variant of Δ spread [5], used to capture the distributional properties of a 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 indicator is to be minimised; 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.

Algorithm 1: Iterated Multi-Objective Local Search

Output: A Pareto set of solutions

```
archive, current ← init();
until termination criterion is met do
    until inner termination criterion is met do
        /* Selection */
        selected ← select(current);
        /* Exploration */
        candidates ← ∅;
        for solution ∈ selected do
            ref ← reference(solution,
             current);
            accepted ← explore(solution, ref);
            candidates ← candidates ⋃
             accepted;
        /* Archive */
        current ← current ⋃ candidates;
        current ← pareto(current);
        current ← bound(current);
    archive ← archive ⋃ current;
    archive ← pareto(archive);
    /* Perturbation */
    current ← perturb(archive);
return archive;
```

B. Multi-objective Local Search

To solve combinatorial problems, metaheuristics are often preferred to exact methods, since they tend to quickly find very good solutions. Commonly used metaheuristics include bio-inspired methods –such as evolutionary, ant colony, and particle swarms optimisation algorithms– and other types of stochastic local search methods.

Multi-objective local search (MOLS) algorithms are efficient metaheuristics obtained by extending relatively simple, yet powerful stochastic local search (SLS) algorithms to deal with multi-objective optimisation problems [10]. Many MOLS algorithms have been proposed, most of them based on the Pareto local search (PLS) [20]; these include iterated PLS [6], stochastic PLS [7], and anytime PLS [9].

In our experiments, we use the generalised MOLS framework [3], outlined in Algorithm 1. The high-level MOLS algorithm iterates over four distinct phases: *selection*, *exploration*, *archive* and *perturbation*. All four phases can be instantiated in many different ways, controlled by categorical parameters, which specify key aspects of the strategies being used, and numerical parameters, which are used to calibrate these strategies. In the following, we give an overview of these parameters and their permissible values (see Table I) that lead to $10\,920$ configurations.

Selection. The first step of every iteration is the selection of a subset of solutions from the current archive (i.e., Pareto set) to be further explored, controlled by parameter `select-strat`. From the current archive, either all solutions are selected, or only a specific number (specified by parameter `select-size`); in the latter case, solutions can be selected either uniformly at random, or according to the order

TABLE I
MOLS PARAMETERS AND CONFIGURATION SPACE

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`	$\{3, 5, 10\}$

in which they have been added to the archive (from oldest to newest). In all cases, a solution whose neighbourhood has been completely explored is never selected again.

Exploration. Parameter `explor-strat` determines how the neighbourhood of each solution is explored and which neighbours will be included into the archive. If the exploration is exhaustive (values `all`, `all_imp`), all improving and non-dominated neighbours, or only all improving neighbours, are added as candidates, respectively. Otherwise, exploration ends after a given number (parameter `explor-size`) of either improving (values `imp`, `imp_ndom`) or non-dominated (value `ndom`) neighbours have been found and added as candidates. Non-dominated neighbours evaluated during an `imp_ndom` exploration are also added as candidates. The parameter `explor-ref` specifies if the "improving" and "non-dominating" criteria are computed using only the current solution being explored (value `sol`) or the current set of all selected solutions (value `arch`).

Archive. After the exploration phase, all accepted candidate solutions are added to the archive, and Pareto-dominated solutions are filtered out. If the size of the archive exceeds the value of parameter `bound-size`, non-dominated solutions are only added to the archive if they replace at least one of the current solutions.

Perturbation. When the inner termination criterion is met, neighbourhood exploration is stopped and perturbation is applied to the current archive in order to diversify the search. Depending on parameter `perturb-strat`, either new solutions, generated uniformly at random (value `restart`) are considered, or a kick move (values `kick`, `kick_all`) is applied to a given number of solutions (parameter `perturb-size`) or to all solutions of the archive, respectively. A solution selected for a kick move is replaced by a solution reached by given number of search steps (parameter `perturb-strength`), performed sequentially and uniformly at random within the given neighbourhood.

III. PERMUTATION PROBLEMS

A. 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 of size N. Several objectives are commonly used for the classical multi-objective PFSP, including makespan (total completion time of the schedule), total flowtime (sum of the individual completion times) and total tardiness (when a due date is associated to each job). In this work, we need to control the correlation between the objectives. Therefore, we consider a bi-objective PFSP minimising 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 these matrices as $f_k(\pi) = \sum_{i=1}^{N} C^k_{\pi_i}$, where π is the sequence of jobs and $C^k_{\pi_i}$ the completion time of job i regarding to matrix P^k.

For this work, we generated our own instances following the idea of uniform random generation underlying the commonly used Taillard instances [22] and using a protocol introduced in [14]. 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 neighbourhoods include the exchange neighbourhood, where the positions of two jobs are exchanged, and the insertion neighbourhood, where one job is reinserted at another position in the permutation. In this study, we consider a hybrid neighbourhood defined as the union of the exchange and insertion neighbourhoods, which is known to lead to better performance than considering each neighbourhood independently [9].

B. The Bi-objective Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the most widely studied combinatorial optimisation 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 travelled is minimised, i.e., a minimum-weight Hamiltonian cycle in G. This cycle corresponds to a permutation of the cities.

In this work, we consider the bi-objective symmetric 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 covered by a given tour according to each of the two distance matrices D^1 and D^2, respectively.

Once again, we follow the protocol presented in [14]: 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. Following Kessaci-Marmion et al. [14], we obtained TSP instances with medium and high correlation using $\rho = 600$ and $\rho = 150$, respectively.

In the following, we consider the so-called 2-opt neighbourhood, where two tours are neighbours 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.

C. 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 $\pi = \{\pi_1, \ldots, \pi_N\}$, 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 [16]. In this work, we consider 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: $f_k(\pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} w^k_{i,j} d_{\pi_i,\pi_j}$.

As for the PFSP and TSP, we follow the same protocol [14] 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 $[0; 99]$.

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 neighbour of a solution is obtained by swapping two facilities between their two locations.

D. Benchmark Sets

For each of these three bi-objective permutation problems, we considered six benchmark sets, using two instances sizes and three degrees of correlation between the given objectives. For the PFSP, we considered 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 considered a set of 50-city instances and a set with 100-city instances. Similarly, for the QAP, we considered 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.

IV. Automatic Algorithm Configuration

A. Background

Given an algorithm A with parameters that affect its performance, e.g. running time on a given input, *automatic algorithm configuration (AAC)* aims to automatically determine performance-optimising settings of these parameters for a given set or distribution of inputs of A. In this context, A is called the *target algorithm*, each combination of settings of A's performance parameter is called a *configuration of* A, and the procedure used to find performance-optimising configurations is known as a *configuration procedure* or *configurator*. Furthermore, we refer to the set of all valid configurations of A as the *configuration space of* A. We note that automatic algorithm configuration can be seen as a supervised learning problem, with the set or distribution of inputs of the given target algorithm as *training data*.

AAC involves the optimisation of a single performance objective, such as running time or solution quality as the optimisation objective. However, the problem can easily be extended to deal with multiple performance measures of a given target algorithm, which leads to the *multi-objective automated algorithm configuration (MO-AAC) problem* we consider in the following. Formally, in MO-AAC, the performance of a given configuration is characterised by a vector, and the goal is to find a Pareto-optimal set of configurations.

There has been a substantial amount of research on single-objective algorithm configuration (SO-AAC) procedures, resulting in several widely used high-performance, general-purpose configurators, including ParamILS [12], [13], SMAC [11] and irace [1], [18]. Recently, ParamILS has been extended to multi-objective algorithm configuration, and the resulting MO-ParamILS procedure has been demonstrated to be effective in a number of MO-AAC scenario [2]. In light of their performance, versatility, and ready availability, we chose to use ParamILS as SO-AAC and MO-ParamILS as MO-AAC in this study.

B. Experimental Protocol

We consider three AAC approaches for optimising the performance of a multi-objective local search algorithms in

TABLE II
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 III
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

terms of hypervolume and spread indicators [4]. HV is a SO-AAC approach that optimises the hypervolume indicator only; it gives a baseline for the other two approaches that take into account both performance metrics of interest. HV**+**Δ' is a SO-AAC approach that optimises a weighted sum of hypervolume and spread. Finally, $HV||\Delta'$ is a MO-AAC approach that simultaneously optimises hypervolume and spread with Pareto consideration.

Experiments are conducted for a given scenario following three phases [2] 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+Δ' approaches. Both approaches using ParamILS followed the recommendation of using adaptive and aggressive

capping [12]. All SO and MO approaches used the Focused variant of (MO-)ParamILS.

During the *validation* phase, each configuration returned by the configurator is run 1 time on each of the 30 training instances, totalling 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, totalling 100 runs, and Pareto-dominated configurations are filtered to obtain the final set of optimal configurations, presented and discussed in the next Section.

V. Experimental Results

In this section, we present the results of our experiments on the three problems we studied, followed by a general discussion of the impact of objective correlation on AAC.

A. Optimised Configuration

PFSP. Tables II and III show the final non-dominated configurations found by the three AAC approaches HV, HV+Δ' 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 favoured 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 objective does not seem to impact the set of optimised configurations.

TSP. Tables IV and V show the final non-dominated configurations found by our three AAC approaches, HV, HV+Δ' 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 MOLS 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 VI and VII 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 for the PFSP and TSP, and vary with instance size as well as correlation between the objectives. The `restart` perturbation strategy is favoured for small,

TABLE IV
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 V
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 VI
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		

TABLE VII
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

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.

TABLE VIII
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)			
HV	3	**0.494**–0.495	1.230–1.385	3	**0.354**–0.375	1.499–1.884
HV+Δ'	5	0.495–0.519	0.402–1.523	3	0.355–0.375	1.485–1.930
$HV\|\|\Delta'$	13	0.495–0.519	0.090–1.454	15	0.355–0.384	0.071–1.930
			(medium correlation)			
HV	1	**0.503**	1.186	3	**0.369**–0.370	1.551–1.748
HV+Δ'	4	**0.503**–0.508	1.056–1.209	8	**0.369**–0.382	1.015–1.746
$HV\|\|\Delta'$	19	**0.503**–0.527	0.025–1.187	14	0.370–0.393	0.006–1.548
			(no correlation)			
HV	1	**0.521**	0.970	1	0.387	1.167
HV+Δ'	1	**0.521**	0.971	5	0.387–0.390	0.822–1.170
$HV\|\|\Delta'$	17	**0.521**–0.550	0.129–0.975	20	**0.386**–0.411	0.074–1.215
	TSP 50 cities			**TSP 100 cities**		
			(high correlation)			
HV	3	**0.157**–0.158	0.669–0.804	2	**0.115**	0.623–0.629
HV+Δ'	4	0.158–0.160	0.591–0.670	2	**0.115**	0.627–0.629
$HV\|\|\Delta'$	6	**0.157**–0.164	0.452–0.732	2	**0.115**	0.623–0.630
			(medium correlation)			
HV	1	**0.167**	0.657	2	**0.123**	0.662–0.816
HV+Δ'	2	**0.167**	0.658–0.662	3	**0.123**	0.660–0.662
$HV\|\|\Delta'$	2	**0.167**	0.658–0.663	5	**0.123**–0.126	0.644–0.815
			(no correlation)			
HV	1	**0.185**	0.676	4	**0.139**–0.140	0.884–0.956
HV+Δ'	1	**0.185**	0.676	5	**0.139**–0.141	0.654–0.957
$HV\|\|\Delta'$	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)			
HV	2	**0.319**	0.881–0.882	1	**0.319**	0.839
HV+Δ'	2	**0.319**	0.872–0.893	1	**0.319**	0.821
$HV\|\|\Delta'$	8	**0.319**–0.321	0.169–0.891	6	0.320–0.321	0.080–0.815
			(medium correlation)			
HV	3	**0.321**	0.849–0.884	1	**0.320**	0.868
HV+Δ'	3	**0.321**	0.848–0.878	5	**0.320**–0.321	0.000–0.886
$HV\|\|\Delta'$	3	**0.321**–0.322	0.172–0.861	8	**0.320**–0.321	0.000–0.907
			(no correlation)			
HV	2	**0.322**	0.787–0.794	4	**0.321**	0.702–1.219
HV+Δ'	2	**0.322**	0.781–0.796	3	**0.321**	0.710–1.249
$HV\|\|\Delta'$	5	**0.322**	0.710–0.7973	4	**0.321**	0.000–0.708

B. Configurator Performance

Table VIII summarises the performance of our three AAC approaches, HV, HV+Δ', 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+Δ', in particular for the PFSP. While HV+Δ' 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+Δ' approach, when the 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 the work of Blot et al. [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.

VI. Conclusion

In our experiments, we automatically configured multi-objective local search (Algorithm 1) on 18 different scenarios, spanning three bi-objective permutation problems, two instance sizes per problem, and three degrees of correlation between optimisation objectives.

We found that of the three configuration approaches we considered, the multi-objective AAC procedure $HV||\Delta'$ tends to work best; for most of our scenarios, it was able to find larger sets of non-dominated configurations that typically span a larger range of Δ' values. At the same time, $HV||\Delta'$ still achieves hypervolume values that compare favourably against those obtained from HV, which only optimises hypervolume. The $HV{+}\Delta'$ approach shows performance similar to that of HV, while only sometimes achieving better spread of configurations, when its aggregation direction closely matches the shape of the optimal front of solutions.

Interestingly, although the automatically optimised configurations of MOLS found by our three AAC approaches are similar within each scenario, they vary markedly between scenarios, and we found that for each parameter we considered, every possible value was present in at least one of the optimised, non-dominated configurations. Looking closer, we noticed differences induced by all three factors, problem, instance size, and objective correlation, as well as preferred values for some parameters. While the optimal configurations on PFSP instances does not seem to be affected by correlation, larger instances slightly favour more aggressive strategies. Performance on TSP instances is similarly affected by low correlation between objectives and large instances, which both make the problem considerably harder.

Finally, correlation level and instance size both impact different parameters of MOLS algorithms on QAP instances.

Overall, our results show 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. Further insights might be obtained based on an exhaustive analysis of a reduced configuration space for each scenario, in which the effects of all parameters can be determined more precisely.

References

[1] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-Race and iterated F-Race: An overview," in *Experimental methods for the analysis of optimization algorithms*. Springer, 2010, pp. 311–336.

[2] A. Blot, H. H. Hoos, L. Jourdan, M.-É. Marmion, and H. Trautmann, "MO-ParamILS: A multi-objective automatic algorithm configuration framework," in *LION 10*, 2016, pp. 32–47.

[3] A. Blot, L. Jourdan, and M.-É. Kessaci-Marmion, "Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem," in *GECCO 2017*, 2017, pp. 227–234.

[4] A. Blot, A. Pernet, L. Jourdan, M.-É. Kessaci-Marmion, and H. H. Hoos, "Automatically configuring multi-objective local search using multi-objective optimisation," in *EMO 2017*, 2017, pp. 61–76.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE TEC*, vol. 6, no. 2, pp. 182–197, 2002.

[6] M. M. Drugan and D. Thierens, "Path-guided mutation for stochastic Pareto local search algorithms," in *PPSN XI*. Springer, 2010, pp. 485–495.

[7] M. M. Drugan and D. Thierens, "Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies," *JOH*, vol. 18, no. 5, pp. 727–766, 2012.

[8] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle, "A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems," *COR*, vol. 38, no. 8, pp. 1219–1236, 2011.

[9] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle,, "Anytime pareto local search," *EJOR*, vol. 243, no. 2, pp. 369–385, 2015.

[10] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

[11] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *LION 5*, 2011, pp. 507–523.

[12] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *JAIR*, vol. 36, pp. 267–306, 2009.

[13] F. Hutter, H. H. Hoos, and T. Stützle, "Automatic algorithm configuration based on local search," in *Twenty-Second AAAI Conference* , 2007, pp. 1152–1157.

[14] M.-E. Kessaci-Marmion, C. Dhaenens, and J. Humeau, "Neutral neighbors in bi-objective optimization: Distribution of the most promising for permutation problems," in *EMO 2017*. Springer, 2017, pp. 344–358.

[15] J. Knowles and D. Corne, "On metrics for comparing nondominated sets," in *IEEE CEC*, vol. 1, 2002, pp. 711–716.

[16] J. Knowles and D. Corne, "Instance generators and test suites for the multiobjective quadratic assignment problem," in *EMO 2003*, LNCS, no. 2632, 2003, pp. 295–310.

[17] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E. Talbi, "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems," *JOH*, vol. 18, no. 2, pp. 317–352, 2012.

[18] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *OR Perspectives*, vol. 3, pp. 43–58, 2016.

[19] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," in *IEEE CEC*, vol. 2, 2003, pp. 878–885.

[20] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for Multiobjective Optimisation*. Springer, 2004, pp. 177–199.

[21] N. Riquelme, C. von Lücken, and B. Barán, "Performance metrics in multi-objective optimization," in *Proceedings of the 2015 Latin American Computing Conference*, 2015, pp. 1–11.

[22] E. Taillard, "Benchmarks for basic scheduling problems," *EJOR*, vol. 64, no. 2, pp. 278–285, 1993.

[23] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens, "On the structure of multiobjective combinatorial search space: Mnk-landscapes with correlated objectives," *EJOR*, vol. 227, no. 2, pp. 331–342, 2013.

[24] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE TEC*, vol. 7, no. 2, pp. 117–132, 2003.

[25] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE TEC*, vol. 3, no. 4, pp. 257–271, 1999.