IRIDIA

Université Libre de Bruxelles

*Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle*

A Landscape-based Analysis of Fixed Temperature and Simulated Annealing

A. Franzin and T. Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2021-005

March 2021

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires*
et de Développements en Intelligence Artificielle
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2021-005

A Landscape-based Analysis of Fixed Temperature and Simulated Annealing

Alberto Franzin[a], Thomas Stützle[a]

[a]*IRIDIA, Université Libre de Bruxelles (ULB), Belgium*

Abstract

Since the introduction of Simulated Annealing (SA), researchers have considered variants that keep the same temperature value throughout the whole search and tried to determine whether this strategy can be more effective than the original cooling scheme. Several studied have tried to answer this question without a conclusive answer and without providing indications that could be useful for a practical implementation. In this work, we address this question following an experimental approach, relating the characteristics of the algorithms with the characteristics of the landscapes they encounter. We use problem-independent landscape features to study the algorithmic behaviour across different problems. We consider three different objective functions and various instance classes and determine the conditions under which the fixed-temperature variant of SA can outperform its original counterpart and when SA is instead a better choice.

Keywords: Discrete Optimization; Simulated Annealing; Stochastic Local Search; Automatic Algorithm Configuration; Exploratory Landscape Analysis.

1. Introduction

Stochastic Local Search (SLS) algorithms [20] are a popular class of non-exact methods to tackle optimization problems. They provide a set of guidelines to implement an effective heuristic algorithm for the problem under study. SLS algorithms are particularly useful in complex or black-box scenarios, where it is very difficult to develop a specialized algorithm, and large-scale scenarios that are beyond the feasibility of exact methods. While they usually are not able to find optimal solutions or to recognize one in case they find it, very often they are able to return high quality solution in short time. The reason for a SLS to perform on an instance boils down to its success in balancing its two contrasting capabilities, the exploration of the search space and the exploitation of the search in promising areas of the search space.

Over the years, plenty of methods have been proposed [3, 30]; some of the simplest of such methods remain among the most popular choices, due to their simplicity in the idea and effectiveness in practice. Yet, it is still difficult to understand their behaviour [7]. In a nutshell, simple single-solution SLSs are built on top of pure intensification local search algorithms such as first- or best-improvement. To obtain better results, they have to introduce a certain degree of diversification and one of the main ways of achieving this is to allow for the acceptance of worsening moves. One of the main examples of such metaheuristics is Simulated Annealing (SA). It uses a parameter called *temperature*, which is progressively lowered during the search to transition from a diversification behaviour to an intensification one [25, 50]. One alternative is to use the same temperature value

Email addresses: `afranzin@ulb.ac.be` (Alberto Franzin), `stuetzle@ulb.ac.be` (Thomas Stützle)

throughout the whole search [35]. This SA variant appears in the literature under different names [22, 39, 24, 5] and, for consistency, throughout this work we will refer to it as Fixed Temperature algorithm (FTA).

In this work we study how these two simple SLS methods perform on different problems and instances to understand the impact of their diversification strategies on different scenarios. A long-standing open question is to determine which of those two algorithms would obtain better results on a given instance, that is, whether it exists a temperature value for which FTA outperforms SA under any cooling scheme [22]. This question has been addressed in the literature mostly from a theoretical point of view, without a conclusive answer [22, 5, 37, 39, 52, 32]. We take instead a different approach, performing extensive experiments to compare these two algorithms on different problems and instance classes. For a meaningful comparison we obtain the best possible configuration for the algorithms in each scenario. Subsequently, we analyse the configurations obtained and determine the conditions for FTA to match or outperform SA. We focus in particular on some characteristics of the search space, as problem-independent features that we can study across the various scenarios, and to understand what characteristics a search space should exhibit for a FTA to perform as good as or better than SA. The specific research question addressed in this paper serves therefore also as an example of how to perform problem-independent analyses of algorithms.

We report experiments on three problems, the Quadratic Assignment Problem and the Makespan and Total Completion Time objectives of the Permutation Flowshop Problem. For each objective function we consider two instance classes of different characteristics. We use automatic algorithm configuration to generate the best possible FTA and SA algorithm to observe the potential of the algorithms in each scenario [28]. We analyze the results and configurations obtained, and relate them with information on the landscape of each instance, following an Exploratory Landscape Analysis approach [33]. We show that if the solution neighbourhoods for a given instance of a combinatorial optimization problem have the same structure in different areas of the solution space, then FTA works well. Conversely, if the neighbourhood structures vary for different areas of the search space, SA will find better solutions than FTA thanks to its higher adaptive capability.

In the next section, we review the literature comparing SA and FTA. In Section 3 we introduce the experimental settings uses for our experiments, reported in Sections 4 and 5. In Section 6 we discuss our results in relation with the solution landscape before concluding in Section 7.

2. Literature review

Simulated Annealing (SA) is a popular metaheuristic inspired by the annealing process in metallurgy [25, 50]. Starting from a given initial solution, SA iteratively evaluates its neighbourhood and accepts a move deterministically, if the solution either improves or is the same quality of the current incumbent, or with a probability that depends on the relative worsening of the solution quality with respect to the current incumbent and a parameter called temperature [34]. The temperature parameter is normally set at high values in the beginning of the search and slowly decreased as the search proceeds. This way, the algorithm slowly transitions from an initial exploratory behaviour, where many poor quality solutions are likely to be accepted, towards a final intensification one, where only good quality moves are likely to be accepted. The change of value of the temperature parameter is controlled by the so-called *cooling schedule*, a function that defines a usually non-increasing sequence of temperature values.

Soon after the introduction of SA, researchers began to study the behaviour of SA variants that keep the same temperature value throughout the entire search. This variant has been studied in seemingly independent lines of research, and appears in literature under several names such as Metropolis Algorithm [22, 23, 17], Static SA [39], Generalized Hill Climbing [24], or Fixed Temperature SA algorithm [5, 11] (FTA). For simplicity, we use FTA in the remainder of this paper.

The first work that mentioned a SA with a fixed temperature in the optimization literature is probably the one of Mitra *et al.* in 1985 [35]. In 1989, Hayek and Sasaki studied a SA for the polynomial-time matching problem and gave examples for which any monotone decreasing temperature sequence is not optimal [19]. They conjecture that no monotone decreasing temperature sequence is optimal for a broader set of cases. They also consider a (deterministic) threshold random search, prove that there is an optimal sequence of threshold values, and state that probably in many situations there is an optimal deterministic threshold sequence that outperforms any random threshold sequence; incidentally, this can be considered the first study of deterministic variants of SA, later also called Threshold Acceptance [36, 9]. However, they add that "the practical implication of these likelihoods is clouded, since it is unclear how to efficiently find an optimal temperature sequence or deterministic threshold sequence for a problem instance" [19], and thus SA is probably a good fallback solution. This is also an implicit statement about the necessity for an instance-based temperature schedule and about the adaptive capability of SA, something we are going to discuss more in detail in Section 6.

In 1996, Jerrum and Sinclair [22] study the Markov Chain Monte Carlo (MCMC) method and conclude with a comparison of SA and FTA (called Metropolis Algorithm in their work) as an application of MCMC. They also consider the matching problem, and observe how MA either solves or finds good approximations on all the instances with high probability in polynomial time. Their method can prove the optimality of a temperature value for a given instance, but no constructive procedure is given to compute it. However, the analogous analysis for SA is much more complex; the outcome is that optimal coolings do exist, but they are so slow that are not competitive with exhaustive search. Hence, it remains an open question whether SA can beat MA on a natural problem.

Mühlenbein and Zimmermann offer an alternative perspective, stating that choosing a proper neighborhood is more important than a cooling scheme, hence their answer to the question whether "to cool or not" is negative, and they suggest instead to use a Variable Neighborhood Search approach [37].

Orosz and Jacobson call the fixed-temperature variant of SA the Static Simulated Annealing (S^2A) and introduce an estimator of the upper bound on the expected number of moves to reach a solution of a certain target quality [39]. They support the theoretical derivation with some experiments on the TSP, but they do not compare S^2A with SA or other algorithms.

Wegener proves instead that SA beats MA for a certain class of Minimum Spanning Tree instances, in terms of expected time and probability of finding the optimal solution in a given bounded time [52]. Meer applies the same principle to the TSP and constructs some TSP instances for which MA is again outperformed by SA [32].

However, all these contributions are theoretical and focus on existence proofs, and in certain cases they address polynomially solvable problems. For a practicioner's perspective, they share the following drawbacks.

(i) There is no real indication on the right values for a fixed temperature scheme.

(ii) There is no guidance about which algorithm to choose for a certain given problem or problem instance.

(iii) The optimality of a value or a certain method is proven only a posteriori, that is, one should already be able to compute the optimal value in advance. This means, in practice, to solve the problem to optimality in order to be able to compute the optimal temperature value.

(iv) It is unclear how much these existence results generalize to other problems, how they apply to "natural" instances instead of artificially constructed ones, how they generalize when different components are used in the algorithm or, simply, how to consider them for a limited runtime instead of an infinite one.

However, as pointed out in [5]: "convergence is not relevant to the success of the algorithm". In fact, being all these analyses performed from a theoretical perspective, they deal with the "manageable" theory of Markov chains. From an algorithm design and implementation point of view this means that a vast amount of possible design choices that can make the algorithm perform well in practice are ignored. For example, only random moves in the neighbourhood are considered, while a different neighbourhood exploration scheme can be significantly better than a random one on certain problems [14]. Likewise, other effective components such as the temperature restart are ignored in the SA analyses.

Parallel to the theoretical works, some authors noticed that when a cooling strategy was reaching certain temperature values, good solutions were quickly obtained. Thus, the first practical indications on how to obtain these good temperature values began to appear in the literature. The first one we can find comes from Rothman, who according to [47] introduced SA in the Geophysics community in 1985. He noted that "the notion of a *critical temperature* is perhaps the most important, and less understood" open question in the development of a SA algorithm [45]. In a follow-up work, he presents both a theoretical and experimental analysis on a specific problem, stating that the "solution is obtained by dropping immediately from a high T[emperature] to a low T, and then maintaining this low T [. . .]"; initial and final temperature values are determined experimentally via trial and error [46].

Connolly tackles the Quadratic Assignment Problem (QAP) in a similar fashion, using SA with a Lundy-Mees cooling scheme [29] and records the temperature at which the best solution is found; after a certain amount of discarded moves, this "optimal temperature" is set and kept fixed [6].

Basu and Frazer also determine experimentally a good temperature value for a geophysics problem they study [1], evaluating several short runs at different temperature values, and choosing the temperature with the best results. It is interesting to note that they justify their approach noting how theoretically optimal cooling schemes are unacceptable in practice.

Cohn and Fielding study several cooling schemes for SA, and find that a fixed temperature works well for some TSP instances, probably depending on the structure of the instance [5]. They estimate the number of moves required to get to a specified target quality, and the relative optimal temperature value, but still deriving those value from preliminary runs of a SA until the best solution is found. Hence, this work runs in the same issues of its theoretical counterparts by still requiring to know or estimate the best solution. They also discuss an example instance from [18], and they show that in that case the optimal schedule is what they call *boiling*, an infinite temperature schedule that accepts every move with probability 1. In other words, according to their analysis the optimal strategy for that instance is a random walk. In a subsequent work, Fielding gives the first closed formula for three problems, TSP, QAP, and Graph Partitioning, again based on the estimate of the number of moves required for a given instance [11]. The formulas are extrapolated from a very limited set of instances, so they do not appear to be particularly robust. In particular, the formula proposed for QAP does not correspond to what we observe in our experiments reported in Section 4.

Except for this last work, all the papers comparing SA and FTA study one single problem. There is, however, not clear univocal conclusion that can be drawn from the existing body of work on this subject. The discrepancy in the results reported suggests that which algorithm between

Algorithm 1: Component-based formulation of a SA algorithm. In SMALLCAPS the components we choose via automated tuning, in **boldface** the ones we keep fixed. FTA is obtained by omitting lines $12 - 14$.

```
   Input: a problem instance $\Pi$, a neighbourhood $\mathcal{N}$ for the solutions, an initial
          solution $s_0$, control parameters
   Output: the best solution $s^*$ found during the search
1  best solution $s^* =$ incumbent solution $\hat{s} = s_0$;
2  $i = 1$;
3  $t_1 =$ initialize temperature;
4  while stopping criterion is not met do
5      choose a solution $s_{i+1}$ in the neighbourhood of $\hat{s}$ according to SEARCH SPACE
         EXPLORATION criterion;
6      if $s_{i+1}$ meets acceptance criterion then
7          $\hat{s} = s_{i+1}$;
8      end
9      if $\hat{s}$ improves over $s^*$ then
10         $s^* = \hat{s}$;
11     end
12     if TEMPERATURE LENGTH is reached then
13         update temperature according to cooling scheme and TEMPERATURE
             RESTART;
14     end
15     $i = i + 1$;
16 end
17 return $s^*$;
```

SA or FTA performs better is to be determined on a case-by-case basis. In particular, since the optimal (local) search algorithm configuration for a problem and instance depends on the landscape it traverses, in Section 6 we will relate the performance of the algorithms with the characteristics of different landscapes.

3. Materials and methods

3.1. Algorithms and implementation

We can implement FTA as a fixed temperature SA, and therefore the components that differentiate the two algorithms are those that control the temperature during the execution. More precisely, we consider the traditional geometric cooling scheme for SA and a fixed temperature scheme for FTA; consequently, SA also employs a temperature length and a temperature restart component. The outline of the SA algorithm is given in Algorithm 1. FTA is obtained by simply omitting the temperature update components (cooling scheme and temperature length, lines $12-14$ in the outline).

3.1.1. Algorithmic components

The set of algorithm-specific components used is a subset of the ones considered in our previous work [14]. Here we list the options. For a more detailed description and original references we refer the reader to [14]. The common components between the two algorithms are the acceptance criterion, for which we use only the traditional Metropolis criterion [34], the initial temperature scheme, the neighbourhood exploration, the termination condition, and the problem-specific components (initial solution and neighbourhood). Importantly, the key parameters we tune are the initial temperature scheme and the neighbourhood exploration, as they do not discriminate between FTA and SA and have a strong impact on the algorithm performance [14].

We implement the set of components for the two algorithms in the EMILI framework [40]. For every experiment reported in this work, the algorithms are instantiated at runtime by selecting the desired combination of components and numerical parameters. The selection is done automatically using the irace configurator [28]. In each experiment, irace finds the best configuration on a training set of instances, and the final configuration is then evaluated on a separated test set. The duration of each experiment is also problem-dependent.

The problem specific components and experimental settings are described in the following sections relative to each problem.

Initial temperature. We consider five options: a given initial temperature value, an initial value computed proportionally to a given initial solution value, and three schemes based on a preliminary random walk: (i) a value proportional to the highest gap between consecutive solutions observed, (ii) a value proportional to the average gap between consecutive solutions observed, and (iii) a value that yields a desired initial acceptance probability of worsening moves.

Neighbourhood exploration. We consider four options to select one move to evaluate in the neighbourhood: the traditional random scheme, the sequential scheme that evaluates the solutions in a given order until one is accepted, and two partial evaluations of the neighbourhood. These two latter schemes randomly select k solutions in the neighbourhood, and choose either (i) the overall best one in the subset, or (ii) the first improving move found when evaluating the subset in a random order. If no such solution exists in the subset, the least worsening one is selected.

Temperature length. The options selected for the temperature length are a constant value, a value proportional to the instance size, a value proportional to the neighbourhood size, and three schemes based on a maximum number of accepted moves: (i) a given constant value, (ii) either a given value or a maximum amount of moves evaluated, or (iii) either a given value or a maximum amount of moves evaluated proportional to the neighbourhood size.

Temperature restart. The three options to reset the temperature to its initial value are: (i) no restart, (ii) restart when a given minimum temperature is reached, or (iii) restart when the rate of accepted moves falls below a given threshold.

3.2. QAP setup

The Quadratic Assignment Problem (QAP) is an NP-hard problem that models the assignment of n facilities to n locations [4, 26]. Between each pair of facilities i and j there is an associated flow f_{ij}, and between each pair of locations k and l there is an associated distance d_{kl}. A solution of a QAP instance can be represented as a permutation π, where each element $\pi(i)$ contains the location of facility i. The objective function is

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)}. \tag{1}$$

We use a random permutation as initial solution, and the neighbourhood function is the exchange neighbourhood

$$\mathcal{N}(\pi) = \{\pi' \mid \pi'(j) = \pi(h) \wedge \pi'(h) = \pi(j) \wedge \forall l \notin \{j, h\} \ : \pi'(l) = \pi(l)\} \tag{2}$$

that swaps two elements in positions i and j in a solution π. The size of the exchange neighbourhood is $n(n-1)/2$. The cost of a solution $s\prime$ neighbour of s can be computed in linear time starting from s.

We use a set of 600 instances in total, equally divided in two classes, that we refer to as random and structured instances. Instances in the first class have data matrices generated uniformly at random [21], while instances in the second class have data matrices generated randomly following an Euclidean structure [49]. Each class has an equal number of instances of size 60, 80 and 100, which are beyond scale for current exact methods. As no certified optimal solution is available for our set of instances, we use as target value for our evaluation an upper bound on the optimal value previously computed. For each class and size, 50 instances are assigned to the training set for our tuning tasks, and 50 instances to the test set for the evaluation of the configurations obtained. The tuning and testing is done separately for each class and instance size, and the runtime of each algorithm on an instance is ten seconds.

3.3. PFSP setup

The Permutation Flow Shop Problem (PFSP) is another permutation problem that models several scheduling problems arising in real life [16, 12, 43]. It requires to sort n jobs to be executed on a set of m machines. The ordering of the jobs is the same on every machine, and each job i takes p_{ij} units of time to be completed on machine m. A solution is a permutation π of length n, where each element $\pi(i)$ contains the index of the i-th job to be executed. Several variants of the PFSP exist to model different situations arising in real production environments. These usually involve the completion time of the jobs; we denote with $C_{\pi(i),j}$ the completion time of the i-th job in the solution on machine j. Among the several possible objective functions to be optimized, we consider two of the most studied ones, the Makespan objective

$$\min C_{\max} = C_{\pi(n),m}, \tag{3}$$

where $C_{\max}$ is the completion time of the last job on the m-th machine (PFSP-MS, [10]), and the Total Completion Time objective

$$\min \sum_{i=1}^{n} C_{\pi(i),m}, \tag{4}$$

that minimizes the completion time of each job (PFSP-TCT, [42, 41]). In the variants we consider there is no concurrency nor pre-emption, and all the jobs are ready for execution at instant 0. The PFSP is NP-hard for both objectives.

We use the NEH heuristic to obtain an initial solution for SA and FTA [38]. We consider the insertion neighbourhood that moves a job from position i to position $j \neq i$, resulting in a new permutation

$$\pi' = [\pi(1), \ldots, \pi(j-1), \pi(j+1), \ldots, \pi(k), \pi(j), \pi(k+1), \ldots, \pi(n)] \tag{5}$$

if $j < k$ and

$$\pi' = [\pi(1), \ldots, \pi(k-1), \pi(j), \pi(k), \pi(k+1), \ldots, \pi(j-1), \pi(j+1), \ldots, \pi(n)] \tag{6}$$

if $j > k$. The size of the insert neighbourhood is $n(n-1)$, and the time to evaluate each solution is $O(n)$. Additional experiments with the exchange neighbourhood as defined in the previous section and with a random permutation as initial solution are reported in the Supplementary Material [15].

We use two different instance sets for both PFSP objectives. The first one is a set of randomly generated instances, where we use 40 instances with sizes from 50 jobs and 20 machines to 250 jobs and 50 machines [31]; the test set is instead the Taillard benchmark [48] with 120 instances equally divided in twelve groups with sizes from 20 jobs and 5 machines to 500 jobs and 20 machines. The second benchmark consists in job-correlated and instance-correlated instances from [51], each type of sizes 100 × 20 and 200 × 20. The training set is composed of 20 instances for each type and size (thus 80 in total), while for the test set we use 80 instances for each type and size (320 in total). The runtime limit for each algorithm execution is $(n \times m \times 0.015)/2$ seconds.

3.4. Experiments outline and computational environment

In Sections 4 and 5 we are presenting the results obtained on the QAP and on the PFSP (both objectives) benchmarks, respectively. For each objective function we have two classes of instances, each one containing instances of different size.

We configure a FTA and a SA for each instance class (for separate instance sizes for the QAP, considering all the instance sizes available for the two PFSP objectives). The choice of non-fixed components and numerical parameters is done automatically using the irace R package [28]. Each tuning is repeated 15 times, and each configuration obtained is tested 15 times. A fixed set of random seeds is used throughout all the experiments. For each tuning we use a budget of 5000 experiments, all run on Intel Xeon E5-2680 v3 CPUs running at 2.5GHz, with 16MB cache and 2.4GB of RAM available per algorithm execution. Each algorithm execution is single-thread. The complete list and range of parameters is reported in the Supplementary Material, where also the tuning setups, the instances used and some preliminary experiments are included [15].

For each problem/instance class scenario, we report the results obtained by the final configurations on the relative testbed in terms of percentage deviation from the best known or optimal solutions. We report also a summary of the configurations that we obtain, to understand what parameters are chosen by irace in each scenario.

In Section 6 we finally explain the results and configurations obtained by relating them with the landscape characteristics of each scenario.

4. Quadratic Assignment Problem

We first consider the Quadratic Assignment Problem (QAP). We configure the FTA and SA algorithms separately on our training sets of instances, and compare the results that we obtain on the respective test set. Unless indicated otherwise, we consider separately both the different classes and the different instance sizes, having 25 training instances and 25 test instances for each algorithm. The results in terms of average percentage deviation from the best known solutions are reported in Figure 1 for both random and structured instances. We begin by discussing the results and configurations obtained on the random instances.

4.1. Random instances

As we can see in the left plot of Figure 1, the FTA obtains sligthly better results than SA on the random instances. For sizes 60 up to 100, the solutions found by FTA are on average 0.715, 0.734 and 0.844 worse than the best known solutions, while SA obtains solutions on average 0.781, 0.811 and 0.898 worse than the best known solutions. There is therefore a statistically significant difference in favour of FTA, with p-values of 1.42×10^{-7}, 1.3×10^{-8} and 5.44×10^{-7} for, respectively, instance sizes 60, 80, and 100.

In Table 1 we report the occurrence of each component, among the ones that appear in the final FTA configurations, for the tunings on the different instance sizes and for additional tunings that consider all the instance sizes. The best final FTA configurations for the general random scenario that we obtain are composed by (i) an initial temperature value obtained with an initial temperature

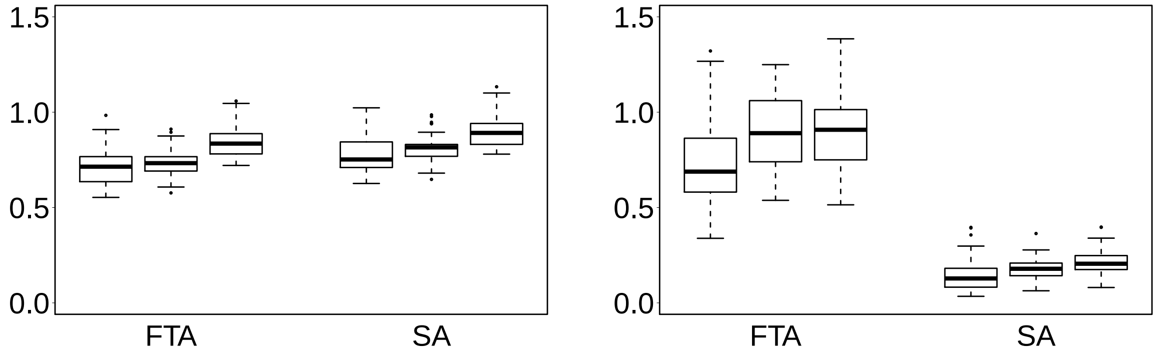

Figure 1: Results in terms of Average Relative Percentage Deviation (ARPD) obtained by the FTA and SA on the QAP random (left plot) and structured (right plot) instances. For each algorithm, the boxplots report the results obtained on instances of size 60, 80, 100.

Table 1: Occurrence of each component in the FTA configurations obtained out of 15 tunings on QAP random instance sizes 60, 80, 100, and on all the instance sizes.

Component	Size 60	Size 80	Size 100	All sizes
proportional to the average gap	8	10	4	11
proportional to the max gap	1	2	5	1
initial acceptance probability	3	2	6	2
fixed initial value	0	0	0	0
value based on initial solution	3	1	0	1
random exploration	0	0	0	0
sequential exploration	8	9	6	12
best of k	7	6	9	3
first best of k	0	0	0	0

scheme proportional by a factor of around 0.22 to the average gap between consecutive solutions in a random walk, and (ii) a sequential exploration. Thus, it looks like the initial temperature value is related to the instance size by means of the relative difference between solutions. With a scaling coefficient of 0.22, the acceptance probability of an average worsening move (that is, worse by the average gap found during the initial random walk) is around 1%. These configurations are observed more consistently in the tunings across all the instance sizes (last column of Table 1), because the homogeneity of the instances in each size makes it possible for other parameter combinations to obtain the same results. The initial temperature scaling coefficient is the key factor for the sequential exploration to make an impact. When this parameter combination is not found, the best alternative is a "best-of-k" exploration that selects the best solution in a randomly chosen subsample of the neighbourhood, with k consistently covering a circa 20% fraction of the neighbourhood. This configuration seems to be easy to find and overall quite robust. The reason for this is the "pure intensification" step of the exploration is able to balance the diversification entailed by a wide range of temperature values, even much higher than the best one.

We consider also additional experiments, reported in the Supplementary Material, that use the random exploration [15]. This is relevant because the random exploration is the common choice for

Table 2: Occurrence of each component in the SA configurations obtained out of 15 tunings on QAP random instance sizes 60, 80, 100.

Component	Size 60	Size 80	Size 100
proportional to the average gap	5	2	6
proportional to the max gap	3	3	2
initial acceptance probability	3	6	3
fixed initial value	3	3	1
value based on initial solution	1	1	3
random exploration	1	0	0
sequential exploration	0	2	2
best of k	4	8	6
first best of k	10	5	7
fixed temperature length	0	2	2
length proportional to instance size	2	4	4
length proportional to neighbourhood size	4	1	4
length based on maximum number of accepted moves	2	1	2
length based on maximum number of accepted moves, capped	2	4	2
max no. of accepted moves, cap proportional to neigh. size	3	1	0
no temperature length	2	2	1
temperature restart based on minimum temperature	5	3	4
restart based on low acceptance rate	3	5	5
no temperature restart	7	7	6

SA algorithms and their variants, and random instances are often used in the literature. With these experiments we can therefore compare the results and configurations we obtain with the existing literature. Our experiments indicate that using the random exploration the best temperature value is consistently around 0.18, slightly lower than with the sequential exploration (0.22). However, even with the lower temperature the results are not as good as the results obtained with the sequential exploration, and, in fact, it was never selected by irace in our primary set of experiments. Solutions found using the random exploration are around 16% worse than those found using the sequential exploration, and are also slightly worse than those found by SA.

The set of final configurations of the SA algorithms is reported in Table 2. As we can see it is very diverse, meaning that several cooling-based algorithms obtain the same results on our set of random instances, but none of them outperforms the best FTA configurations. We can, however, note how the "best-of-k" exploration (or its first improvement variant) with k covering roughly 1/4 of the neighbourhood size appears in half of the final configurations, strengthening the suitability of alternative neighbourhood exploration on this scenario.

4.2. Structured instances

As seen in the right plot of Figure 1, the results on the structured instances are very different than the previous case, with SA clearly outperforming FTA. In fact, a SA tuned with a budget of only 500 experiments obtains better results than a FTA tuned with a budget of 5000 and less parameters. More precisely, FTA found solutions on average 0.764, 0.891 and 0.901 worse than the best known solutions for instances of size respectively 60, 80 and 100. For the same instance sizes SA obtains relative percentage deviations of 0.148, 177 and 0.213.

Table 3: Occurrence of each component in the FTA configurations obtained out of 15 tunings on QAP structured instance sizes 60, 80, 100, and on all the instance sizes.

Component	Size 60	Size 80	Size 100	All sizes
proportional to the average gap	13	14	12	14
proportional to the max gap	2	0	0	0
initial acceptance probability	0	0	1	0
fixed initial value	0	1	1	0
value based on initial solution	0	0	1	1
random exploration	1	3	7	5
sequential exploration	12	10	6	9
best of k	1	1	1	0
first best of k	1	1	1	1

This is clearly a scenario where FTA is not a good choice. Nonetheless, there is a clear indication about what FTA configurations is better on this scenario, as we can see in Table 3. The initial temperature scheme based on the average gap between consecutive solutions in a random walk is chosen most consistently, with the other schemes appearing only sporadically. The difference with the random scenario is that the scaling coefficient is now dependent on the instance size. For instance size 60, the average coefficient is 0.095, for instance size 80 it is 0.078, for instance size 100 it is 0.067 when using the sequential exploration. This means an average acceptance probability of a random average solution of, respectively in these three cases, 0.0027%, 0.00028%, and 3.013×10^{-5}%. Essentially, as the instance size increases, the optimal temperature decreases. The reason for this are discussed in Section 6.

The sequential exploration is once again the most common choice. Conversely to the random instances case, however, the second most common option is the random exploration, while the remaining two schemes rarely appear.

The choice of the configurations, and, in particular, of the initial temperature values, is caused by the structure of the instances: if there is an "easy" valley in the fitness landscape, too much diversification will hamper the search convergence; however, the structure of the slope makes it impossible to find configurations that work fine for both the initial and the latter stages of the search.

The composition of the SA algorithms is again very diverse, as can be seen in Table 4, but in this case the results are consistently good. With respect to FTA, in fact, SA with its transition from "high" to "low" temperature values can better fit the fitness landscape. The temperature restart employed by most of the SA algorithms also makes it possible to cycle between exploration and exploitation several times, increasing the possibility of finding good temperature values for different sequences of neighbourhoods observed. This repeated transition is sufficient to adapt the behaviour of the algorithm, while the specific choice of the components is less relevant.

4.3. Discussion

On these two scenarios, we can observe very different results. On the random instances, no configuration for SA matches the results that are obtained by a proper temperature value of a FTA. This is caused by the shape of the landscape, which can be described as "uniformly smooth", in the sense that the proper temperature values for average-quality areas of the search space are almost the same around good quality solutions. The situation clearly differs on the structured

Table 4: Occurrence of each component in the SA configurations obtained out of 15 tunings on QAP structured instance sizes 60, 80, 100.

Component	Size 60	Size 80	Size 100
proportional to the average gap	4	5	4
proportional to the max gap	1	1	3
initial acceptance probability	3	2	3
fixed initial value	5	3	4
value based on initial solution	2	4	1
random exploration	12	10	5
sequential exploration	3	5	9
best of k	0	0	0
first best of k	0	0	1
fixed temperature length	1	5	3
length proportional to instance size	0	1	0
length proportional to neighbourhood size	3	1	3
length based on maximum number of accepted moves	6	0	3
length based on maximum number of accepted moves, capped	4	2	3
max no. of accepted moves, cap proportional to neigh. size	0	2	2
no temperature length	1	4	1
temperature restart based on minimum temperature	12	7	5
restart based on low acceptance rate	2	3	6
no temperature restart	1	5	4

instances, where SA can better adapt to the change of the landscape. In fact, the variety of SA configurations observed suggests that a progressive adaptation of the exploration/exploitation tradeoff matters rather than a well-defined SA structure. Section 6 is devoted to more detailed analysis of how the FTA and SA are affected by the landscape.

Additional experiments are included in the Supplementary Material [15]. We include experiments with various runtimes and tuning budgets. The best configurations we obtained do not change for different runtimes, where the solution quality increases monotonically with the runtime, nor when we rescale the instance values. This confirms that an initial temperature scheme based on the difference between neighbouring solutions is a proper and robust choice. We also observe how for increasing tuning budgets the results improve, but not as much as it can be expected; in fact, for the two instance classes, a tuning budget of 500 experiments is already sufficient to discriminate between good and bad performing algorithms. We also observe how starting with a good quality solution (computed with a hill climbing, which on our QAP instances can reach solutions between 4 and 5% of RPD) does not impact neither the final solution quality nor the configurations we obtain.

We can now comment on Fielding's temperature formula for the QAP $T = 1.5 \times f^\star/n^2$, where $f^\star$ is the optimal solution value and n the instance size. Fielding extrapolated his formula from results obtained on seven small QAPLIB random symmetric instances,[1] similar to those used in our experiments. The temperature values obtained are roughly inversely proportional to the neigh-

[1] Instance `sko100a` was left out of the regression as an outlier.

bourhood size ($n(n-1)/2$). In our experiments, instead, the (near-)optimal temperature values are related to the relative difference between solutions. On our test instances, the effect of Fielding's formula is a roughly constant temperatue value for increasing instance sizes, while the initial temperature scheme chosen by the tunings in our experiments yields increasing temperature values for increasing instance sizes. The same happens when using the random exploration, the same scheme used by Fielding. Hence, we see that Fielding's formula is a clear overfit over a restricted set of small instances, whose outcome does not generalize.

We note instead how one of the findings of [14] is confirmed in these experiments, namely that the optimal diversification is the minimal one needed to "smoothen" the landscape. irace is able to find the best temperature value for such behaviour, and no configuration among the final ones has values lower than the best one [13].

5. Permutation Flowshop Problem

In Figures 2 and 3 we show the results obtained by our set of algorithms on the Taillard benchmark, for the Makespan and Total Completion Time objectives, respectively, in terms of relative percentage deviation (RPD) with respect to the optimal or best known solutions. For each objective, we show separately the results obtained when testing the FTA and SA algorithms on (i) the 30 Taillard instance with a number of jobs and machines included in our training set (50×20, 100×20, 200×20), (ii) the 80 Taillard instances with a number of jobs included in our training set (with 50, 100 and 200 jobs), and (iii) the whole Taillard benchmark. For simplicity, we refer to these three sets of instances as TIJM, TIJ, and TIA. In Figure 4 we show the results obtained for the two objectives on the Watson benchmark.

All the algorithms use the insertion neighbourhood, and start from an initial solution computed using the NEH heuristic. The components selected by the tunings for the experiments reported in this section are listed in Tables 5 and 6 for FTA and SA respectively. In the Supplementary Material we report a full breakdown of the 12 instance classes of the Taillard benchmark, and additional experiments where the algorithms use the exchange neighbourhood and a random permutation as initial solution [15].

5.1. Makespan objective

5.1.1. Taillard benchmark

Given the variety of the instance sizes and characteristics of this benchmark, where the instances are divided into 12 different groups, overall the results show a high variance. We can anyway distinguish a certain trend. For the TIJM and TIJ instances, whose characteristics are covered in the training set, FTA and SA obtain very similar results (p-values respectively of 0.0221 and 0.02553). On the whole benchmark, instead, SA outperforms FTA (p-value of 1.237×10^{-7}). In terms of solution quality, the results can be very different between different subclasses of instances. We observe also a general higher consistence of the results in each subclass of instances, except for instances of size 50×10, where while the average solution quality of the two algorithms is similar the variance is high.

Instances in TIJM have a low jobs/machine ratio, a case notoriously difficult in practice. On these thirty instances, FTA and SA obtain very similar results. The TIJ subset includes several instances with higher jobs/machine ratios, easier to tackle, and, in fact, the performance improves for both algorithms, remaining similar overall. SA can instead significantly outperform FTA on the remaining instance groups, which include the smaller instances and the largest group, 500×20. Perhaps surprisingly, on the largest instances both algorithms again perform similarly well. Where SA outperforms FTA is instead on the smallest instances, which have a number of jobs lower

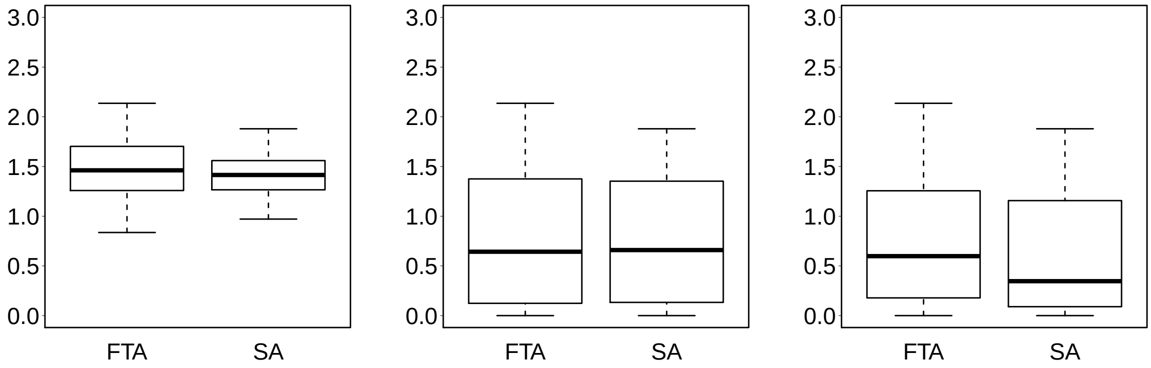

Figure 2: Results in terms of Average Relative Percentage Deviation (ARPD) obtained by the FTA and SA on the PFSP-MS Taillard benchmark. The three boxplots report the results for, left to right, (i) instances with a number of jobs and machines included in the training set (TIJM), (ii) the instances with a number of jobs included in the training set (TIJ), (iii) all instances (TIA).

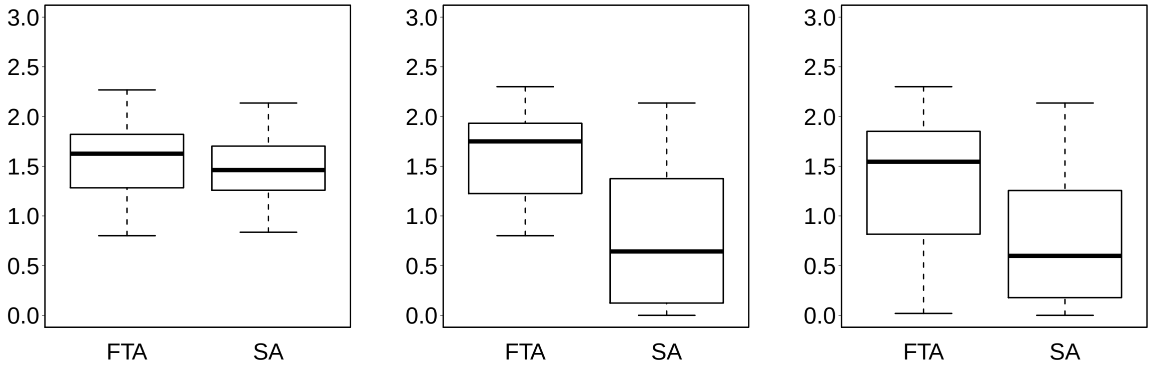

Figure 3: Results in terms of Average Relative Percentage Deviation (ARPD) obtained by the FTA and SA on the PFSP-TCT Taillard benchmark. The three boxplots report the results for, left to right, (i) instances with a number of jobs and machines included in the training set (TIJM), (ii) the instances with a number of jobs included in the training set (TIJ), (iii) all instances (TIA).

than any instance in our training set. On these instances, SA is often able to find the optimal solution, while FTA usually does not, and this explains the statistically significant difference in results observed in the third plot of Figure 2.

Regarding the FTA configurations, the majority of them use an initial temperature based on the average gap between consecutive solutions in a random walk, with a scaling factor for the initial temperature consistently around 0.066, and a random exploration. The remaining ones use a first or best improvement in a small random subsample of the neighbourhood as neighbourhood exploration. This scheme admits a greater variety of initial temperature schemes and values, which therefore appear albeit sporadically in the list of final FTA configurations.

Regarding the SA algorithms, instead, the set of options chosen for the initial temperature is much wider. Many of the algorithms feature instead a cooling coefficient in the range 0.6 to 0.9 range, no temperature restart, a temperature length based on a maximum number of accepted moves, and random exploration of the neighbourhood.

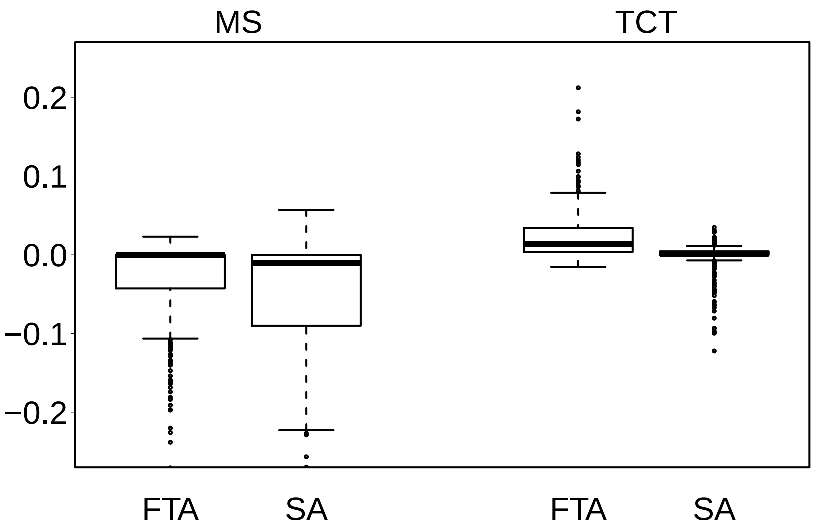

Figure 4: Results in terms of Average Relative Percentage Deviation (ARPD) obtained by the FTA and SA on the Watson benchmark under both the MS and TCT objectives.

5.1.2. Watson benchmark

The results on the Watson benchmark are very homogeneous, and the optimal value of many instances is met or the best known value improved. A paired Wilcoxon test indicates that there is no statistical difference between the FTA and SA results, with a p-value of 0.5865.

The set of FTA configurations in this case is the most homogeneous among our experiments. All the fifteen FTAs use again an initial temperature based on a preliminary random walk, with an average scaling factor of 0.093, and a random exploration. The SA algorithms, instead, regardless of neighbourhood and initial solution scheme, exhibit a variety of choices and combinations, in particular, for the initial temperature. The only steady non-fixed choice is the random exploration of the neighbourhood, but also the temperature length scheme and the absence of a restart option appear often.

Table 5: Occurrence of each component of FTA in the configurations obtained out of 15 tunings on the four PFSP scenarios (Makespan and Total Completion Time objectives, Taillard and Watson instances).

Component	MS Tai	MS Wat	TCT Tai	TCT Wat
proportional to the average gap	12	15	12	4
proportional to the max gap	1	0	0	3
initial acceptance probability	1	0	1	1
fixed initial value	0	0	1	2
value based on initial solution	1	0	1	5
random exploration	10	15	11	1
sequential exploration	0	0	0	0
best of k	2	0	0	10
first best of k	3	0	4	4

Table 6: Occurrence of each component of SA in the configurations obtained out of 15 tunings on the four PFSP scenarios (Makespan and Total Completion Time objectives, Taillard and Watson instances).

Component	MS Tai	MS Wat	TCT Tai	TCT Wat
proportional to the average gap	6	7	3	7
proportional to the max gap	1	2	4	3
initial acceptance probability	2	5	5	2
fixed initial value	2	0	1	2
value based on initial solution	4	1	2	1
random exploration	15	15	15	15
sequential exploration	0	0	0	0
best of k	0	0	0	0
first best of k	0	0	0	0
fixed temperature length	1	1	0	6
length proportional to instance size	0	0	0	0
length proportional to neighbourhood size	0	1	0	1
max number of accepted moves	13	10	15	2
max number of accepted moves, capped	1	3	0	6
max no. of accepted moves, cap $\propto$ neigh. size	0	0	0	0
no temperature length	0	0	0	0
temperature restart based on minimum temperature	0	0	0	0
restart based on low acceptance rate	1	4	0	5
no temperature restart	14	11	15	10

5.2. Total Completion Time objective

5.2.1. Taillard benchmark

When focusing on the TIJM sets of instances, FTA and SA obtain similar results: a pairwise Wilcoxon test gives a p-value of 0.08407. On the TIJ and TIA sets of instances, instead, FTA instead does not obtain results close to those obtained by SA.

The explanation we give is that the instances outside TIJM are easily tackled by the progressive transition in the exploration/exploitation tradeoff of SA algorithms regardless of the specific components used, similarly to what we observed in the structured QAP instances (see Table 5). This interpretation is supported by the variety of SA configurations obtained, and by the landscape analysis of Section 6.

Eleven out of our 15 FTA algorithms have random exploration and average gap-based initial temperature with a scaling factor of 0.071. The remaining ones have a partial local search in the neighbourhood coupled with different initial temperature schemes yielding a higher initial temperature. Simulated annealing algorithms also have several different initial temperature options. However, they all employ the random exploration of the neighbourhood, no temperature restart scheme, and a temperature length based on a maximum number of accepted moves.

5.2.2. Watson benchmark

The results on the Watson benchmark are very homogeneous, and the optimal value of many instances is met or the best known value improved. Nonetheless, SA is still significantly better than FTA with a p-value of 2.2×10^{-16}.

The FTA configurations in these cases are very different; the only shared feature is a partial local search in the neighbourhood as exploration criterion. This is probably the factor that controls the exploration/exploitation tradeoff precisely enough to reach the best results. As we have seen with the QAP experiments in Section 4, this exploration scheme admits a wide range of temperature values, and, in fact, very different fixed initial temperatures appears. For the SA algorithms, instead, we again observe several options chosen for the initial temperature scheme. Most SA configuration also have no temperature restart scheme, and a variety of temperature length schemes. All the configurations, however, share the random exploration of the neighbourhood.

5.3. Discussion

We observe how, on our benchmarks, FTA can obtain results close to those of SA. It is however crucial to choose the right subsidiary components (neighbourhood and neighbourhood exploration) and to tune the temperature value on an instance distribution that matches the test set. When the test instances have different characteristics than the ones used for the training, SA is instead a better choice.

Also in this case, we report in the Supplementary Material additional experiments, where we use the exchange neighbourhood, a different initial solution, and an additional perturbation scheme [15]. In general we observe a regularization effect in the configurations introduced by the perturbation. The values of the initial temperature of the intensification FTA are more consistent when introducing the perturbation, and the same applies to the choice of components. This observation is true across different neighbourhoods and initial solutions, as it can be seen from the additional experiments included in the Supplementary Material. We explain this by the fact that the separation of the intensification and diversification phases makes it possible to have each step tailored for its specific task. There is instead no noticeable impact by the use of the NEH heuristic for the initial solution, as the results are not significantly different from the results obtained starting from a random initial solution for neither FTA nor SA.

We can also explain why the neighbourhood of choice for the PFSP is the insertion one rather than the exchange one as in the QAP. The exchange neighbourhood, in fact, generates a more rugged landscape than the insertion neighbourhood; exchange-based algorithms usually get stuck in a poor local optimum in a few moves. The sets of final solutions found by the algorithms that employ the exchange neighbourhood also have a higher variance than those found using the insertion neighbourhood.

The landscape generated by the Watson instances is extremely smooth, and a randomly generated permutation is expected to be 4% away (on some instances, less than 0.5% away) from the optimal or best known solution; in some cases a simple iterated improvement is sufficient to find the optimal solution. The variety of the configurations obtained and the lack of temperature restart schemes in many SA configurations are further indications of the easiness of the benchmark. The test set also comes from the same instance distributions of the training set. It is therefore not surprising that all the algorithms are, for both objectives, around or close to the optimal or best known solutions. The count for optimal or best solutions found or improved is reported in the Supplementary Material [15].

6. Exploratory Analysis of Landscape Features

FTA and SA introduce a diversification mechanism over a pure intensification local search method. In this section we address the question of how the problem and instance characteristics impact over the effectiveness of this mechanism. Having tested three different objective functions on instances of various characteristics in Sections 4 and 5, in this section we report the results of an exploratory analysis to understand under which conditions the diversification mechanism is more effective. In other words, we are interesting in finding out the conditions under which the constant rate of diversification employed by FTA suffices to obtain good results, and when the flexibility introduced by SA is necessary to outperform FTA.

To be able to study different problems, we need to use problem independent features. In discrete optimization, problem specific features are often considered, for example, in algorithm selection. We use instead an approach that follows the exploratory landscape analyses for continuous optimization problems and tries to characterize the solution landscape [33]. This is a valid approach also for algorithm selection [2]. In discrete optimization, however, it is an open question which kind of features should be considered to represent the landscape. Some recent works considered fitness landscape features for algorithm selection on the QAP, using global features such as fitness distance correlation and average distance to optimal solutions [44, 8]. In this work, however, we have a slightly different albeit related perspective. We do not seek to select different algorithms, but rather to understand how variants of a generic algorithmic template behave on different landscapes. Thus, under our perspective the landscape is generated not only by the objective function and the instance data, but also by the problem-specific components of the algorithm used, in particular in our case the neighbourhood function. By choosing a set of features that capture the behaviour of the algorithm on the landscape, we are then able to understand the relationship between a landscape and the performance of a specific algorithm used to traverse it. We choose therefore to use a set of *algorithmic-dependent* features to represent the landscape that is seen by the algorithms we use.

Since FTA and SA introduce a diversification mechanism on top of an iterative improvement, we probe the search space with a first-improvement and a best-improvement algorithm. By using these algorithms we want to understand what conditions are necessary to escape the local optima, and to relate them to the efficacy of each diversification mechanism tested. The features we compute include measures about the convergence of the search and statistics of the neighbourhoods traversed and are oriented towards observing whether a certain property holds constant throughout the fitness landscape or it changes in different areas of the search space. The full list of features is given in Table 7, and here we describe the most relevant ones, as resulting from the analysis, along with their rationale.

6.1. Feature analysis

For each instance of our test sets, we have run 30 first-improvement (FI) and 30 best-improvement (BI) algorithms, with different random seeds, from which we extract the landscape features used in our analysis. This means that each feature is the aggregated value (average or standard deviation) of 30 values, and every feature is generated twice, one time from the first-improvement runs (indicated as FI), and another from the best-improvement ones (BI). Each algorithm starts from a random initial solution. For the QAP we use the exchange neighbourhood, while for the two PFSP objectives we use the insertion neighbourhood.

The diversification mechanism of FTA accepts two equally worsening moves with the same probability at each step of the search, while SA, as usually defined, has a larger tolerance in the beginning and is much stricter at the end. We have empirically observed that a SA works well with overall low acceptance probabilities (less than 1% in the early stages), otherwise the algorithm will fail to converge to good solution areas. This means that, for FTA to be effective, the conditions for escaping a local optimum without failing to converge at all needs to be roughly the same at any stage of the search. For this to happen, the neighbourhoods traversed have to have the same properties regardless of whether they are centered around good or bad quality solutions. Conversely, the more flexible SA can exploit different settings in different stages of the search to adapt to different landscape profiles. One way of observing this difference is to observe whether the sequence of best solution values found can be better approximated by a linear model, indicating a flatter landscape where FTA can work well, or an exponential model, where SA can exploit its flexibility. We consider the R^2 error of the two models as features, along with their difference. The same model fitting idea can be applied also to the sequence of possible improving moves that appear in the neighbourhoods traversed.

Similarly, we consider that the Metropolis criterion works by rescaling the difference between the incumbent and the candidate solution values. For FTA to escape a local optimum or to deviate from a path that converges to a local optimum, the temperature value needs to be suitable to neighbourhoods of both good and bad solutions. As a proxy for this, first we record the difference between the incumbent solution value and the average solution value in its neighbourhood and then we fit a linear model on the sequence of such differences. A flat slope means that the effort it takes to escape the current path is the same throughout the whole search, while an upward slope means that the local optima are situated in deep valleys, more difficult to escape than other regions of the search space.

Other features that are apparently relevant in our analysis are relative to the average length of the path from a random initial solution to a local optimum. It is also interesting to consider values rescaled by instance or neighbourhood size.

In preliminary analyses we also considered features based on initial and final solution values and relative percentage deviations from the best known solutions (RPD). Several of these features turned out to be the ones with highest predictive power. However, this can be explained with the relatively diverse set of values and RPDs obtained on our set of instances, such that final solution values observed on problems and instance classes where SA and FTA perform similarly well do not overlap with solution values of problems and instances where SA clearly outperforms FTA. Hence, these measures appear to be too dependent on our specific examples. Because of this fact, and because these features are mostly available only a posteriori, we prefer to not include them in the analyses reported in the present section.

We have a set of 38 features observed across the three different objective functions used. For each objective function we have one subset of test instances on which the two algorithms obtain similar results, and another subset on which there is a significant difference in the results. We define a prediction task where, given an instance, the goal is to predict the difference between FTA and SA in terms of relative percentage deviation from the best known or optimal solutions. By training a random forest model for this task, we can estimate the contribution of each feature to the task, and we can select the most relevant features according to the root mean-squared error (RMSE), the standard deviation of the prediction error, computed on the resampling of a 10-fold cross-validation. We use the *caret* R package [27] to run our analyses.

We first report the analysis on the whole set of results, and then we further explore the analysis for each objective function considered, reporting the specific differences arising under each objective function. Since replications of the analyses reported may differ slightly in the number or the order of features selected, for each objective we run ten analyses and identify the features selected most consistently.

6.2. Analysis on the whole dataset

The analyses across the three different objective functions are the ones that give the most consistent results; in fact, the most important features and their order is largely the same throughout the ten replications of the analysis.

The most important feature is the difference between the R^2 errors of a linear and an exponential model fit on the sequence of the normalized solution values, followed by the average number of

Table 7: Set of features used in the analysis. Each feature is computed with both a first-improvement and a best-improvement local search. The aggregated features are computed from 30 independent runs.

1	Average number of moves to local optimum
2	Standard Deviation of number of moves to local optimum
3	Average number of moves to local optimum, rescaled by instance size
4	Standard Deviation of number of moves to local optimum, rescaled by instance size
5	Average number of moves to local optimum, rescaled by neighbourhood size
6	Standard Deviation of number of moves to local optimum, rescaled by neighbourhood size
7	Average R^2 of a linear model fit on the solution values
8	Average R^2 of a linear model fit on the solution values normalized in $[0,1]$
9	Average R^2 of an exponential model fit on the solution values
10	Average R^2 of an exponential model fit on the solution values normalized in $[0,1]$
11	Average R^2 of a linear model fit on the number of improving moves in the sequence of neighbourhoods traversed
12	Average proportion of neutral moves in the neighbourhoods traversed
13	Number of neutral last moves
14	Difference of features 7 and 9
15	Difference of features 8 and 10
16	Average slope of the sequence of differences between best and average solution in a neighbourhood
17	Standard deviation of the slopes of the sequence of differences between best and average solution in a neighbourhood
18	Average of the standard deviation of the sequence of differences between best and average solution in a neighbourhood
19	Standard deviation of the standard deviation of the sequence of differences between best and average solution in a neighbourhood

moves needed to converge to a local optimum, rescaled by the neighbourhood size. The third most important feature is again the difference between the R^2 errors of the two models, but this time on the actual solution values and not on their rescaled values. The fourth feature selected as most relevant is the average ratio of neutral moves in the neighbourhood, followed by the average slope of a linear model fit on the sequence of the differences between the best and average solutions in the neighbourhood traversed, and the standard deviation of the number of moves to reach a local optimum, when rescaled by the neighbourhood size. Except for the average slope, which is computed using a FI, all the other features are computed after BI runs.

In general, on our set of problems and instances FTA works well (as good as SA, or even slightly better) on instances whose landscape converges to locally optimal solutions quickly (that is, in less moves than on the instances where SA outperforms FTA), smoothly and regularly (the solution values fit nicely a linear model).

In Figure 5 we show the main difference in the landscape that is seen by a best-improvement on two QAP instances. On the x-axis we have the moves accepted and on the y-axis we report the percentage ratio between average gap in the neighbourhood and the value of the solution around which the neighbourhood is centered. This ratio represents the average effort necessary to accept one random worsening move and deviate from the current BI path. Therefore a horizontal line would indicate that the effort is perfectly constant, and the proper temperature value one should

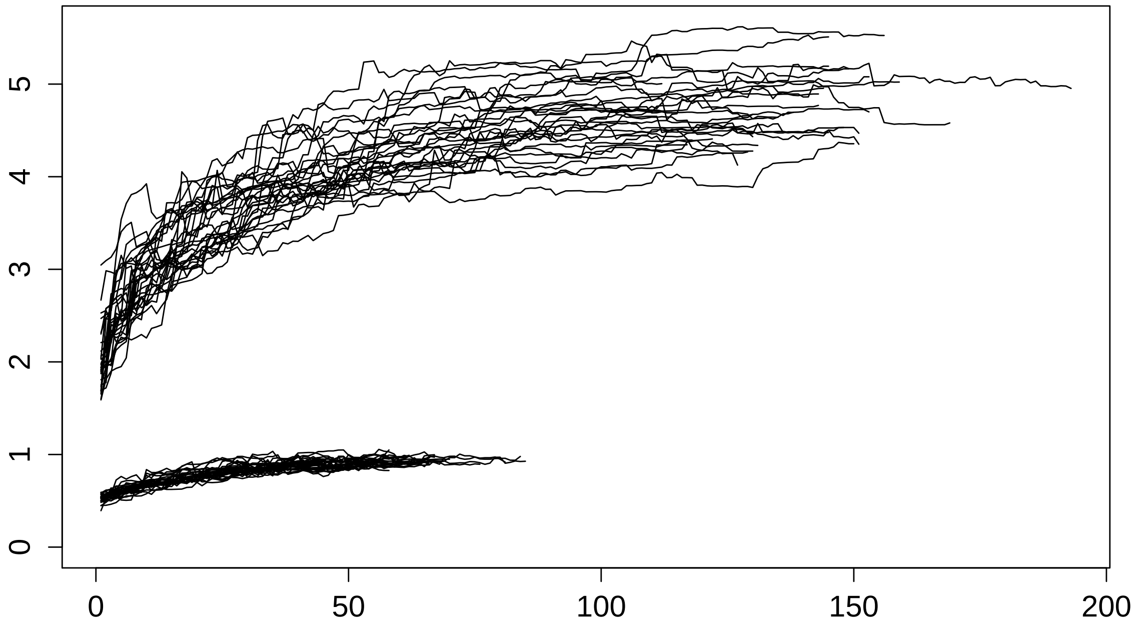

Figure 5: Convergence of thirty BI algorithms on a QAP structured instance (top) and thirty BI algorithms on a random instance (bottom). The x axis reports the moves accepted, and on the y axis we report the ratio (in percentage) between the average gap in the neighbourhood (of the corresponding move) and the value of the solution around which the neighbourhood is centered.

choose is the same for every neighbourhood traversed. In the plot we report ratios obtained along the search for thirty BI algorithms run on two size 100 instances, a structured one (on top) and a random one (on the bottom). The three main differences between the landscapes are immediately clear. First of all, the BIs on the random instance converge much faster than on the structured instance. The structured instance exhibits a high difference between the initial stages of the search and the final stages, where the search stops in a local optimum. Such an increase is not surprising, since for an average solution we can expect half of its neighbours to be better than it; as the search continues, the BI algorithm will end up in a path that is difficult to escape,[2] until it terminates in a local optimum. The third characteristic is a high variability of these paths, that indicate a rough landscape. The curves for the random instance, instead, have a flatter profile, which denotes a higher similarity in the neighbourhoods of both average and high quality solutions, and exhibits a lower variability, an indication of a smoother landscape.

6.3. Quadratic Assignment Problem

For the QAP we have one set of random instances where FTA and SA perform similarly, and a set of structured instances where SA clearly outperforms FTA. Each subset of instances includes

[2] More precisely, that it would be more difficult to escape, if the algorithm had this capability.

three different instance sizes, and for each instance size the results are rather homogeneous.

As in the general case, the average number of moves to converge to a local optimum (rescaled by instance size, feature 3 for BI and FI) appears, along with the difference between the R^2 error of the linear and exponential models on the original solution values (feature 14, for FI), and of the normalized ones (feature 15 for BI and FI). Feature 2, the standard deviation of the number of moves to reach a local optimum is selected nine times. Other related features appear seven times (the average R^2 errors for linear and exponential models, and the standard deviation of the number of moves to converge to a local optimum rescaled by instance and neighbourhood size), and other features less frequently.

On our QAP instances FTA works well on the random instances, whose landscape matches the description given in the general case. SA outperforms FTA on the structured instances, whose landscape changes shape as the solution quality increases (from steeper to flatter), and using different temperature values has an advantage over the fixed settings of FTA. On the structured instances it is very easy for any algorithm to improve over average solutions. However, at a certain point during the solution traversal towards good quality solutions, the landscape will change, and temperature-based diversification will allow to continue improving whereas a FI or BI stops. A good FTA for this sort of landscape has its temperature set to a value that allows to continue the progress beyond the possibility of a FI. However, as the change in the landscape continues, also this temperature value will eventually become ineffective. On the other hand, higher temperature values, which would be more suitable for the latter stages of the search, would also allow a greater diversification in the beginning, rendering the FTA unable to even converge to the solutions found by the FI.

6.4. PFSP, Makespan objective

For the PFSP problem we consider the instances for which we have a corresponding number of jobs in the training set. This ensures that the fixed temperature value chosen is properly chosen, given that the higher diversity of the instances in the Taillard benchmark makes with respect to the QAP scenario.[3] We also take the same number (80) of instances of the Watson benchmark, 20 for each size available.

The following features always appear: the standard deviation of the number of moves to reach a local optimum, rescaled by instance and neighbourhood size (features 2 and 4 for BI), the average number of moves to reach a local optimum and the average ratio of neutral moves in a neighbourhood (features 1 and 12, both FI). The average number of moves to reach a local optimum, rescaled by instance and neighbourhood size, appears nine times, alongside with the average ratio of neutral moves in a neighbourhood, and the average slope of gaps between the best and the average solution value in the neighbourhood (features 1, 2, 5, 12 and 16, all BI).

Our PFSP instances form a more fragmented set than the QAP instances, so in this case it is more difficult to pinpoint a clear distinction between instance groups. It appears, however, that the instances on which FTA is able to find solutions of quality similar to SA, are those where BI and FI are able to converge in a shorter amount of moves, and with a larger proportion of neutral moves in the neighbourhoods. On instances for which BI and FI take a longer time to converge to local optima, or where there is a lower number or neutral moves in the solution neighbourhoods, SA is still the better choice.

[3] However, we observed little difference in the analysis when considering the whole instance set.

6.5. PFSP, Total Completion Time objective

For the TCT objective we consider the same set of instances used in the Makespan case. In this case we have a set of eight features always selected with a consistent order of importance, with only a handful of other features appearing less often. The most important features are the difference between the R^2 errors of a linear and an exponential model (feature 14 for both FI and BI – again for the latter also with normalized solution values, feature 15), the average ratio of neutral moves in a neighbourood as seen by BI (feature 12), the average number of moves to converge to a local optimum (also when rescaled by instance or neighbourhood size, features 1 and 3, for BI), the average R^2 error of a linear model fit on the sequence of solution values (feature 7, BI), and the standard deviation of the standard deviations of the sequence of gaps between best and average solutions in the neighbourhood traversed (feature 19, BI).

In this case, FTA performs well on instances where BI or FI can converge regularly towards local optima, and in a shorter amount of moves. SA instead performs better than FTA when the landscape around average quality solutions differs from the landscape observed in high quality areas of the search space. From this perspective, the TCT objective function generates landscapes that are more similar to those generated by the QAP than to the Makespan objective function.

6.6. Discussion

In this section we have analyzed the landscapes generated by the instances in several different scenarios, and we have been able to relate the characteristics of the landscape to the results obtained by the algorithms we considered. We have therefore determined the conditions necessary for understanding which algorithm between FTA and SA can perform better on a given instance.

The diversification mechanism introduced by the Metropolis acceptance criterion works by allowing to accept worsening moves with a probability that is dependent on both the relative worsening of the move with respect to the incumbent solution and the temperature parameter. Therefore, the temperature enables the transition from the incumbent solution towards a portion of its neighbourhood that increases along with the temperature. In particular, the optimal fixed temperature value would be the one that allows to reach the optimal solution; in practice, we search for a temperature value that obtains the best possible solutions. For this condition to be verified, the same temperature value has to be optimal, or close to optimal, in every neighbourhood traversed. This can happen only if the neighbourhood structure is the same across all the solution space, in the sense that the distribution of differences between neighbouring solutions must be the same for all the solutions. This is indeed the "ideal" case considered in several theoretic works. The more the actual distributions of differences look the same, the better a FTA will perform. In this case, it is therefore possible to determine an optimal temperature value. Lower values for the temperature enforce stricter acceptance conditions, making the search stop in local optima, or suboptimal regions of the solution space. Higher values, instead, will accept too many worsening moves, making the algorithm unlikely to converge towards good solutions.

When, instead, different neighbourhoods will have different distributions of differences, there is no single temperature value that can allow the same optimal traversal throughout the entire search space: values good for certain neighbourhoods will be either too strict or too high for other neighbourhoods. In this case, the adaptive capability of SA will make it possible to eventually find good temperature values in different areas of the search space. This does not mean that SA is able to determine a good temperature value for the incumbent neighbourhood, but rather that there is a good chance that at a certain point the variation of temperature in SA will end up at a value that allows progress, until either the neighbourhood structures or the temperature change.

We have thus offered an alternative perspective to the theoretical studies reported in Section 2. We emphasize that, differently to many of the existing studies on this subject, our approach explicitely relates algorithmic characteristics with instance characteristics and the performance obtaines, on a plurality of problems and instance classes. Contrarily to those studies, moreover, we did not focus on the probability of finding an optimal solution, but on the effective performance under realistic conditions, thus providing an indication about which algorithm to use in practice.

7. Conclusions

Simulated annealing and its variants are a popular and effective class of stochastic local search algorithms. They are widely used in practice, but there is not yet a complete characterization of their behaviour. One of the long standing open questions in the theory of stochastic local search algorithms has been to determine whether it is possible to design a fixed-temperature variant of SA that can outperform its original counterpart. This question has been addressed in several works, but no definitive answer has been provided so far. Researchers have, in fact, for the most part focused on single problems, and on one or very few instances, trying to model the ideal convergence behaviour of the algorithm. However, no work so far has included specific characteristics of the instances in the modeling, essentially detaching the algorithm from the conditions it operates in, and this is the reason for the variety of outcomes of the different analyses.

In this work, we have instead followed an experimental approach, using configuration tools to automatically instantiate the best possible algorithms for a variety of scenarios, and relating them to the solution landscapes traversed. This way, we have been able to answer the question whether "to cool or not" by identifying the conditions that have to be present in order for the fixed temperature variant of SA to obtain solutions as good as or even better than SA. The answer to the question is therefore "it depends", being dependent on the landscape generated by the instance. FTA works well for instances whose distribution of differences between neighbouring solutions is approximately constant. If, instead, neighbourhoods have a different structure in different areas of the search space, for example they differ for average quality solutions and good quality ones, SA is able to find better solutions.

Beyond the particular research question addressed in this work, we have shown how to combine automatic algorithm configuration and design techniques with problem-independent landscape analysis to infer knowledge about the general behaviour of stochastic local search algorithms.

This work can be extended in several ways. One direction is to include additional problem-independent features in the analysis, and determine the minimal set necessary to make per-instance algorithm selection and configuration feasible at runtime. Likewise, including different problems and instance classes will make it possible to expand our analysis and observe the behaviour of FTA and SA under different conditions. Finally, the same approach can be applied to other classes of algorithms, both metaheuristics such as genetic algorithms or exact methods like MIP solvers, to gain further understanding about how they work. The final goal of this direction is to move away from predefined algoritmic structures to be able to automatically instantiate algorithms whose components and parameters are tailored to a specific problem and instance.

Acknowledgments

Alberto Franzin acknowledges support from the Innoviris project 2018-SHAPE-25a "PHILEAS: smart monitoring par détection de comportements anormaux appliquée aux objets connectés". Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Director.

References

[1] Basu, A. and Frazer, L. N. (1990). Rapid determination of the critical temperature in simulated annealing inversion. *Science*, 249(4975):1409–1412.

[2] Bischl, B., Mersmann, O., Trautmann, H., and Preuss, M. (2012). Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In Soule, T. and Moore, J. H., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 313–320. ACM Press, New York, NY.

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

[4] Burkard, R. E., Çela, E., Pardalos, P. M., and Pitsoulis, L. S. (1998). The quadratic assignment problem. In Pardalos, P. M. and Du, D.-Z., editors, *Handbook of Combinatorial Optimization*, volume 2, pages 241–338. Kluwer Academic Publishers.

[5] Cohn, H. and Fielding, M. J. (1999). Simulated annealing: Searching for an optimal temperature. *SIAM Journal on Optimization*, 9(3):779–802.

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

[7] Cook, W. J. (2019). Computing in combinatorial optimization. In Steffen, B. and Woeginger, G., editors, *Computing and Software Science: State of the Art and Perspectives*, volume 10000 of *LNCS*, pages 27–47. Springer, Cham.

[8] Dantas, A. and Pozo, A. (2020). On the use of fitness landscape features in meta-learning based algorithm selection for the quadratic assignment problem. *Theoretical Computer Science*, 805:62–75.

[9] Dueck, G. and Scheuer, T. (1990). Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175.

[10] Fernandez-Viagas, V., Ruiz, R., and Framiñán, J. M. (2017). A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. *European Journal of Operational Research*, 257(3):707–721.

[11] Fielding, M. J. (2000). Simulated annealing with an optimal fixed temperature. *SIAM Journal on Optimization*, 11(2):289–307.

[12] Framiñán, J. M., Leisten, R., and Ruiz, R. (2014). *Manufacturing Scheduling Systems: An Integrated View on Models, Methods, and Tools*. Springer, New York, NY.

[13] Franzin, A., Pérez Cáceres, L., and Stützle, T. (2018). Effect of transformations of numerical parameters in automatic algorithm configuration. *Optimization Letters*, 12(8):1741–1753.

[14] Franzin, A. and Stützle, T. (2019). Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research*, 104:191–206.

[15] Franzin, A. and Stützle, T. (2021). A landscape-based analysis of fixed temperature and simulated annealing: Supplementaty material. `http://iridia.ulb.ac.be/supp/IridiaSupp2021-002`.

[16] Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–129.

[17] Haario, H., Saksman, E., and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242.

[18] Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329.

[19] Hajek, B. and Sasaki, G. (1989). Simulated annealing–to cool or not. *System & Control Letters*, 12(5):443–447.

[20] Hoos, H. H. and Stützle, T. (2005). *Stochastic Local Search—Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA.

[21] Hussin, M. S. and Stützle, T. (2014). Tabu search vs. simulated annealing for solving large quadratic assignment instances. *Computers & Operations Research*, 43:286–291.

[22] Jerrum, M. and Sinclair, A. (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In Hochbaum, D. S., editor, *Approximation Algorithms For NP-hard Problems*, pages 482–520. PWS Publishing Co.

[23] Jerrum, M. and Sorkin, G. (1998). The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82(1):155–175.

[24] Johnson, A. W. and Jacobson, S. H. (2002). On the convergence of generalized hill climbing algorithms. *Discrete Applied Mathematics*, 119(1):37–57.

[25] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

[26] Koopmans, T. C. and Beckmann, M. J. (1957). Assignment problems and the location of economic activities. *Econometrica*, 25:53–76.

[27] Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(5):1–26.

[28] López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

[29] Lundy, M. and Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming*, 34(1):111–124.

[30] Martí, R., Pardalos, P. M., and Resende, M. G. C., editors (2018). *Handbook of Heuristics*. Springer International Publishing.

[31] Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., and Stützle, T. (2013). From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In Pardalos, P. M. and Nicosia, G., editors, *Learning and Intelligent Optimization, 7th International Conference, LION 7*, volume 7997 of *LNCS*, pages 321–334. Springer.

[32] Meer, K. (2007). Simulated annealing versus Metropolis for a TSP instance. *Information Processing Letters*, 104(6):216–219.

[33] Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In Krasnogor, N. and Lanzi, P. L., editors, *GECCO*, pages 829–836. ACM Press, New York, NY.

[34] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092.

[35] Mitra, D., Romeo, F., and Sangiovanni-Vincentelli, A. (1985). Convergence and finite-time behavior of simulated annealing. In *Decision and Control, 1985 24th IEEE Conference on*, pages 761–767. IEEE.

[36] Moscato, P. and Fontanari, J. F. (1990). Stochastic versus deterministic update in simulated annealing. *Physics Letters A*, 146(4):204–208.

[37] Mühlenbein, H. and Zimmermann, J. (2000). Size of neighborhood more important than temperature for stochastic local search. In *IEEE CEC*, pages 1017–1024, Piscataway, NJ. IEEE Press.

[38] Nawaz, M., Enscore, Jr, E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

[39] Orosz, J. E. and Jacobson, S. H. (2002). Analysis of static simulated annealing algorithms. *Journal of Optimization Theory and Applications*, 115(1):165–182.

[40] Pagnozzi, F. and Stützle, T. (2019). Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European Journal of Operational Research*, 276:409–421.

[41] Pan, Q.-K. and Ruiz, R. (2012). Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, 222(1):31–43.

[42] Pan, Q.-K. and Ruiz, R. (2013). A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. *Computers & Operations Research*, 40(1):117–128.

[43] Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems.* Springer, New York, NY, 4th edition.

[44] Pitzer, E., Beham, A., and Affenzeller, M. (2013). Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis. In Middendorf, M. and Blum, C., editors, *EvoCOP*, volume 7832 of *LNCS*, pages 109–120. Springer.

[45] Rothman, D. H. (1985). Nonlinear inversion, statistical mechanics, and residual statics estimation. *Geophysics*, 50(12):2784–2796.

[46] Rothman, D. H. (1986). Automatic estimation of large residual statics corrections. *Geophysics*, 51(2):332–346.

[47] Sambridge, M. (1999). Geophysical inversion with a neighbourhood algorithm–i. searching a parameter space. *Geophysical Journal International*, 138(2):479–494.

[48] Taillard, É. D. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

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

[50] Černý, V. (1985). A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

[51] Watson, J.-P., Barbulescu, L., Whitley, D., and Howe, A. E. (2002). Contrasting structured and random permutation flow-shop scheduling problems: Search space topology and algorithm performance. *INFORMS Journal on Computing*, 14(2):98–123.

[52] Wegener, I. (2005). Simulated annealing beats metropolis in combinatorial optimization. In Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C., and Yung, M., editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *LNCS*, pages 589–601. Springer.