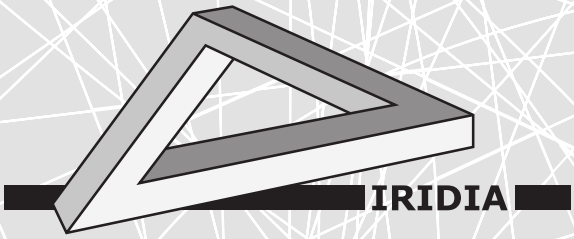


**Université Libre de Bruxelles**

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

# Comparison of Acceptance Criteria in Randomized Local Searches

A. Franzin and T. Stützle

**IRIDIA – Technical Report Series**

Technical Report No.
TR/IRIDIA/2017-010

August 2017

**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/2017-010

# Comparison of Acceptance Criteria in Randomized Local Searches

Alberto Franzin and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
alberto.franzin@ulb.ac.be, stuetzle@ulb.ac.be

**Abstract.** One key component of stochastic local search algorithms is the acceptance criterion that determines whether a solution is accepted as the new current solution or it is discarded. One of the most studied local search algorithms is simulated annealing. It often uses the Metropolis condition as acceptance criterion, which always accepts equal or better quality solutions and worse ones with a probability that depends on the amount of worsening and a parameter called temperature. After the introduction of simulated annealing several other acceptance criteria have been introduced to replace the Metropolis condition, some being claimed to be simpler and better performing. In this article, we evaluate various such acceptance criteria from an experimental perspective. We first tune the numerical parameters of the algorithms using automatic algorithm configuration techniques for two test problems, the quadratic assignment problem and a permutation flowshop problem. Our experimental results show that, while results may differ depending on the specific problem, the Metropolis condition and the late acceptance hill climbing rule are among the choices that obtain the best results.

## 1 Introduction

Stochastic local search (SLS) methods are generic procedures commonly used to tackle hard optimization problems [9]. They are composed of a set of general rules of how to design effective heuristics for specific optimization problems; hence, an alternative name for these methods is *meta*-heuristics. Often, the sometimes rather problem-specific heuristic algorithms derived from these rules are very effective in finding high quality solutions in short computation time, and for many problems such algorithms define the state of the art.

To achieve good solutions, SLS methods balance the intensification of the search in narrow regions, often needed to find the best solutions in promising search space areas, with the exploration of different areas of the search space. One mechanism that many trajectory-based SLS methods use to promote diversification is the acceptance of solutions that are worse than the current incumbent solution. In this article, we call *acceptance criterion* the function devoted to determining whether a newly proposed candidate solution should replace the current one. A first metaheuristic that proposed a probabilistic acceptance criterion for acceptinf a worse candidate solution is simulated annealing (SA) [10,23].

It uses the so called Metropolis condition from statistical mechanics, which relies on a parameter called temperature, as acceptance criterion [14]. The name of the parameter mimics the temperature of a physical system, which was used in a Monte-Carlo simulation of physical systems proposed by Metropolis *et al.* [14]. According to the Metropolis condition, an improving or equal quality candidate solution is always accepted, while a worsening candidate solution is accepted with a probability that depends both on the quality difference between the current solution and the newly proposed one and the temperature parameter. To create a transition from search diversification to intensification of the search, in a typical SA algorithm the temperature is initially set to some high value (corresponding to a rather likely acceptance of worsening candidate solutions) and then subsequently lowered to make the acceptance of worsening candidate solutions less likely.

Over the years, various new ideas have been conceived with the motivation to improve over this usual acceptance criterion of SA algorithms. These new acceptance criteria have been compared in individual papers often directly to basic SA algorithms and in various such papers potential improvements have been reported. These new ideas include refinements of the Metropolis condition, such as generalized SA [2] and the bounded Metropolis condition [6]; a criterion where the acceptance probability of worsening solutions decreases geometrically [17]; and deterministic criteria such as threshold acceptance [8,15], the great deluge and record-to-record travel algorithms [7], and the late acceptance hill climbing [5]. The latter four methods all accept a solution with probability one when it meets the specific, deterministic acceptance conditions. In our experiments, we also include a basic hill climbing acceptance criterion [1], which accepts a solution if and only if it improves over the incumbent, as a baseline the other criteria need to outperform.

The original articles proposing these acceptance criteria often report experimental results on few problem instances or on very small instance sizes. One reason is that many of these acceptance criteria were introduced when experimental conditions available were quite different from today. Hence, there is limited indication in the original works on how to apply the various methods to different problems. To just cite one example, in the original paper on threshold acceptance, the authors present a sequence of values for the "threshold" parameter, stating that *"We have the feeling (really only the feeling, not, for instance, the impression) that the sequence above is somewhat better* [than another sequence mentioned]" [8]. The comparisons in these papers are also usually performed against the Metropolis condition and a limited set of the other criteria.

In this work, we compare well-known acceptance criteria on common benchmark sets, derived from two classical, NP-hard problems, namely the quadratic assignment problem (QAP) and the permutation flow-shop problem with the total completion time objective (PFSP-TCT). To obtain unbiased results we tune the numerical parameters of the algorithms, using the automatic algorithm configuration tool irace [11]. We evaluate the impact of the nine different criteria we study in terms of the quality of the final solutions and the robustness

of the criterion. Our experiments show that the results may change according to different problems, instance classes, or experimental condition. Overall, the Metropolis condition, its generalized version and, in particular, the rather recent late acceptance hill climbing are the criteria that gave the best results.

## 2 Literature review

We first introduce the notation used in the remainder of this work. We consider NP-hard combinatorial optimization problems, in which for a given problem instance $\pi$ a globally optimal solution $s^* \in S$, where $S$ is the search space of candidate solutions, is to be found. The quality of solutions is evaluated according to an objective function $f : S \mapsto \mathbb{R}$ and $f(s)$ is the objective function value for a generic solution $s$. Without loss of generality, we consider minimization problems, that is, for a globally optimal solution it holds that $f(s^*) \leq f(s), \forall s \in S$. Each algorithm we consider uses an iteration counter of the search process, which is denoted by $i$, and $s_i$ is the new candidate solution evaluated in that iteration. The difference in terms of objective function value between two solutions $s_i$ and $s_j$ is denoted with $\Delta(i, j)$, or simply $\Delta$ when no confusion may arise. With $\hat{s}$ we indicate the incumbent solution. The neighbourhood of $\hat{s}$ is denoted by $\mathcal{N}(\hat{s})$ and comprises all candidate solutions that can be reached from $s$ by one application of the neighborhood operator.

**Algorithm 1:** Outline of a generic randomized search algorithm.

**Input**: problem instance $\Pi$, $\mathcal{N}$, initial solution $s_0$, control parameters
**Output**: best solution $s^*$

```
1  best solution s* = incumbent solution ŝ = s_0;
2  parameter initialization, i := 0;
3  while stopping criterion is not met do
4      while parameters settings fixed do
5          i := i + 1;
6          generate a random solution s_i ∈ N(ŝ);
7          ŝ := accept (ŝ, s_i);
8          s* := best (s*, ŝ);
9      end
10     update parameters;
11 end
12 return s*;
```

All SLS methods that we consider can be interpreted as instantiations of the generic algorithm outlined in Algorithm 1. It starts from a given initial solution as incumbent (line 1), and iteratively generates one new candidate solution in the neighbourhood of the incumbent uniformly at random (line 6); at iteration $i$ the new candidate solution $s_i$ can be chosen to replace the current incumbent $\hat{s}$

if it meets some criteria (e.g. it is an improving solution, line 7), otherwise it is discarded. Periodically, the parameter(s) that control the search may get updated (line 10). All the algorithms we examine here fit in this generic template. They only differ in the acceptance criterion. However, some of these algorithms may not use all the components of the algorithm; for example, the late acceptance hill climbing relies only on one parameter that is held constant during the run of the algorithm and, hence, does not need to be updated in the outer loop (lines 3 to 11). In the following, the counter $k$ refers to the number of times the outer loop has been invoked. Conversely, SA and others evaluate solutions using the same parameter values in the inner loop (controlled by the temperature length, lines 4 to 9), and update the parameters in the outer loop.

SA, proposed independently in [10] and [23], is inspired by work in statistical physics [14]. In the usual, basic variants, SA iteratively generates and evaluates one random solution $s \in \mathcal{N}(\hat{s})$; if the new solution is better or equal to the incumbent in terms of objective function value, it replaces the incumbent one; otherwise it gets accepted with a probability that depends on the relative difference in terms of objective function values, $\Delta(s, \hat{s})$, and on the temperature parameter, denoted as $T$. The acceptance criterion of SA can be written as

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\Delta(s, \hat{s})/T\right) & \text{otherwise.} \end{cases} \tag{1}$$

This probabilistic criterion is known as Metropolis acceptance criterion or Metropolis condition, and it is the distinctive feature of SA. We refer to this criterion simply as SA in the rest of this paper.

More recently, in [6] the authors argue that solutions that are worse with respect to the incumbent by a quantity that exceeds a certain threshold $\phi_{BM}$ are not worth considering at all. This *bounded* Metropolis criterion (BSA) accepts a solution $s$ with a probability

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\Delta(s, \hat{s})/T\right) & \text{if} \quad 0 < \Delta(s, \hat{s}) \leq \phi_{BM} \\ 0 & \text{if} \quad \Delta(s, \hat{s}) > \phi_{BM}, \end{cases} \tag{2}$$

where $\phi_{BM}$ is a parameter.

Soon after the introduction of SA, the Metropolis acceptance criterion has been generalized in [2], where a variant of SA called generalized simulated annealing (GSA) was introduced. The GSA acceptance criterion is defined as

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ \exp\left(-\beta f(\hat{s})^{\gamma} \Delta(s, \hat{s})\right) & \text{otherwise,} \end{cases} \tag{3}$$

where $\beta$ and $\gamma$ are control parameters. Even if the temperature parameter is not explicitly considered in GSA, it is possible to recreate the original Metropolis condition by defining $\beta = 1/T$.

In [17], the authors propose a criterion in what is the first occurrence of a SA variant that does not consider the temperature value in the acceptance of solutions. They propose to accept a solution with probability

$$p^k = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq 0 \\ p_0 \times \rho^{k-1} & \text{otherwise.} \end{cases} \tag{4}$$

where $p_0$ is the initial acceptance probability, $0 < \rho < 1$ is a reducing factor, and $k$ is the number of times the probability has been updated. In this *geometric* acceptance criterion, the temperature value is (possibly) related only to the initial acceptance probability; during the search, the updating process of the probability matters, rather than the actual value of a temperature.

The actual need of stochasticity in the Metropolis acceptance criterion is questioned independently in [8] and [15]. In both works, the authors propose a criterion that accepts any move that is either improving or worsening by at most a given threshold $\phi^k > 0$:

$$p = \begin{cases} 1 & \text{if} \quad \Delta(s, \hat{s}) \leq \phi^k \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\phi^k$ is the value at step $k$ of the threshold, which gets updated periodically. In [8], the authors consider a sequence of thresholds, without giving any indication on how to set its initial value or how to update it. Our implementation follows [15], maintaining the SA terminology: the initial value of $\phi$ is the initial temperature of SA, and the updating process of the threshold is called cooling. This threshold acceptance (TA) is a deterministic version of SA. At the time of its introduction, it was argued that using TA is faster than evaluating the Metropolis condition as it does not require the generation of a random number and the computation of an exponential. This advantage may be important when the computation of the objective function value of a neighboring candidate solution is very fast. However, for problems that benefit little from incremental update schemes or where the computation of the objective function value of neighboring candidate solutions is expensive (as is the case in the problems we study here), the advantage of a faster computation of the acceptance test diminishes.

Two acceptance criteria have been derived from TA and proposed in [7] as new algorithms. The first algorithm and criterion proposed in [7] is called record-to-record travel (RTR), and accepts solutions that do not deviate from the best solution found so far plus a given threshold $\phi$:

$$p_{RTR} = \begin{cases} 1 & \text{if} \quad f(s) \leq f(s^*) + \phi \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

RTR is therefore a stricter version of TA, which compares the newly proposed candidate solution with the current incumbent; moreover, in the RTR algorithm $\phi$ does not get updated.

The second algorithm proposed in [7] is called great deluge algorithm (GDA) and is a radical change in terms of solution evaluation, as it moves away from the idea of comparing solutions. The acceptance criterion of GDA accepts every move whose objective function value is lower than a certain threshold that gets progressively lowered during the search

$$p^k = \begin{cases} 1 & \text{if} \quad f(s) \leq \phi^k \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

with $\bar{\phi}^{k+1} = \phi^k - \lambda$, $\lambda$ being a fixed parameter. The consequence of a lowering bound is that GDA becomes increasingly strict for accepting solutions.

A more recent work proposes another simple deterministic acceptance criterion, called late acceptance hill climbing (LAHC) [4,5]. This algorithm makes no use of a temperature-like parameter, but maintains limited knowledge about the history of the search. It accepts every solution $s$ that is improving either with respect to the current incumbent $\hat{s}$ or with respect to a solution visited $\kappa$ iterations before, for a fixed $\kappa$:

$$p = \begin{cases} 1 & \text{if} \quad f(s) \leq \min\{f(\hat{s}), f(s_{i-\kappa})\} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Finally, we consider as baseline for the comparison a simple hill climbing (HC) algorithm [1] that accepts a solution if and only if it improves over the incumbent. Obviously, we expect the other criteria to obtain better results with respect to HC. While in practice one would implement HC using a systematic enumeration of the neighbourhood, we implemented it inside the framework of Algorithm 1 for convenience.

## 3 Experimental Setup

The nine acceptance criteria presented in Section 2 are evaluated as candidate acceptance criteria for a generic algorithm outlined in Algorithm 1. The common components of the nine implementations are: (i) a random exploration of the neighbourhood, (ii) no parameter restarting rule (e.g. temperature restart in SA), and (iii) a termination condition based on runtime. The runtime differs for each problem, so the actual value is given below.

For the criteria that need initial values for their parameters (such as the temperature for the SA family of algorithms, or the threshold $\phi$ in TA), we use a value proportional by a coefficient $\epsilon$ to the maximum gap between consecutive solutions observed during an initial random walk of length 10000 in the solution space. The parameters that need to be modified during the algorithm run time (e.g. temperature in SA or threshold in TA) are updated using a geometric decreasing; e.g., the temperature $T$ in SA is updated according to the formula $T_{k+1} = \alpha \times T_k$, where $\alpha$ is a parameter. The inner loop of Algorithm 1 evaluates a number of solutions that is given by $\tau \cdot |\mathcal{N}(s)|$, where $\tau$ is a parameter and $|\mathcal{N}(s)|$ is the size of the neighbourhood of a solution $s$.

**Table 1.** Parameter values for the algorithms.

| | Metro | BMetro | GSA | Geom | TA | GDA | RTR | LAHC |
|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | $[0, 10]$ | $[0, 10]$ | $[0, 10]$ | $[0, 10]$ | $[0, 10]$ | $[0, 10]$ | – | – |
| $\alpha$ | $[0, 1]$ | $[0, 1]$ | $[0, 1]$ | $[0, 1]$ | $[0, 1]$ | $[1, 100]$ | – | – |
| $\tau$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ | $[1, 100]$ | – | – |
| $\phi, \phi_{BM}$ | – | $[0, 1]$ | – | – | – | – | $[0, 1]$ | – |
| $\beta$ | – | – | $[10^{-4}, 10]$ | – | – | – | – | – |
| $\gamma$ | – | – | $[0, 10]$ | – | – | – | – | – |
| $\rho$ | – | – | – | $[0, 1]$ | – | – | – | – |
| $\kappa$ | – | – | – | – | – | – | – | $[1, 10^4]$ |

We choose to not use parameter restart schemes to better observe the impact of the main algorithm component that is studied, the acceptance criterion. The periodic reset or increase of parameters such as the temperature or the threshold is often beneficial to obtain better results, as it facilitates search space exploration, but it also has the side effect of smoothening the difference in terms of impact of the other components.

The parameter values, and their presence for each algorithm, are given in Table 1. Parameters equivalent in scope and values are grouped together. The only algorithm that does not use the components described above is GDA. During the experimental phase, we have observed very poor results when using the GDA acceptance criterion with the choices above, indicating a lack of flexibility of the method. We thus consider the GDA algorithm in its original settings, which are anyway valid components that fit in the template of Algorithm 1. The initial threshold value $\phi$ is computed proportional to the objective function value of the initial solution, using a coefficient $\epsilon \in [0, 10]$; $\phi$ is updated according to the formula $\phi_{k+1} = \phi_k - \alpha$, where $\alpha$ is an integer in the interval $[1, 100]$; we bound this decrease to 0. The other components are as described above.

Our setup considers as test problems the quadratic assignment problem (QAP) [3] and the permutation flow-shop problem with the total completion time objective (PFSP-TCT) [19,18]. The QAP models the location of a set of facilities, with the goal of minimizing the overall distance between facilities taking into account also the flow between them. PFSP instead is a scheduling problem where a set of jobs have to be ordered to be executed on a set of machines.

For the QAP we use a randomly generated initial candidate solution and the exchange neighbourhood, which is defined as

$$\mathcal{N}(s) = \{s' \mid s'(j) = s(h) \wedge s'(h) = s(j) \wedge \forall : l \notin \{j, h\}\ s'(l) = s(l)\}, \quad (9)$$

where $s(j)$ is the solution vector at position $j$. The neighbourhood size is $n(n-1)/2$, where $n$ is the instance size. The running time considered for termination is 10 seconds. We consider two different instance sets of size 100, one where all QAP instance data are generated uniformly at random, and one randomly generated in analogy to structured real-world like QAP instances. Each instance set is divided into a training set of 25 instances and a test set of 25 instances.

From here onwards, we refer to these two scenarios as random instances and structured instances, respectively. The two scenarios are not mixed, that is, the configurations obtained for the random instances are evaluated on the random instances and not on the structured ones, and viceversa.

For the PFSP-TCT we use the `NEH` heuristic [16] for the initial solution generation. For an instance of size $n \times m$, where $n$ is the number of jobs and $m$ the number of machines, the neighbourhood is the insert neighbourhood that randomly picks one element $s(j)$ in position $j$ of the permutation $s = [s(1), \ldots, s(n)]$ and inserts it in position $k \neq j$, obtaining

$$s' = [s(1), \ldots, s(j-1), s(j+1), \ldots, s(k), s(j), s(k+1), \ldots, s(n)] \tag{10}$$

if $j < k$ and

$$s' = [s(1), \ldots, s(k-1), s(j), s(k), s(k+1), \ldots, s(j-1), s(j+1), \ldots, s(n)] \tag{11}$$

if $j > k$. The neighbourhood size is $n(n-1)$. In this case, we use an instance-based maximum runtime of $n \times m \times 0.015$ seconds. The training set consists of 40 randomly generated instances of size ranging from 50 jobs and 20 machines to 250 jobs and 50 machines [13] and the test set is composed by the instances `Tai31-110` of the Taillard benchmark [21]. We will, however, discuss separately the instances whose size is smaller than those covered by the training set (those with $n = 20$), covered by the training set (`Tai31-110`), and larger ($n = 500$).

We tune the numerical parameters using irace [11] with a budget of 2000 experiments per tuning on an Intel Xeon E5-2680 v3 CPU, with a speed of 2.5GHz, 16MB cache and 2.4GB of RAM available for each job. For each algorithm we run nine tunings, evaluate the best configuration obtained from each tuning on the test set, and average the final solution quality obtained on each instance by the nine configurations. The real valued parameters have a precision of 4 decimal digits.

## 4 Experiments on the Quadratic Assignment Problem

In Figure 1 we show the results obtained by the nine algorithms after the tuning on the random instances and on the structured instances respectively. Each boxplot reports the results obtained on the test instances in terms of the average relative percentage deviation (ARPD) from the best known solutions. In Table 2, we report the results of the Friedman rank sum test, obtained for the nine algorithms on the two QAP instance classes. The algorithms are ordered according to the sum of their ranks, and the difference in terms of rank sum with the best ranked algorithm is computed along with a statistical significance threshold. Algorithms whose rank sum differs from the best ranked one by a value larger than the significance threshold are statistically significantly worse than the best one.

On the random instances, RTR obtains the best results, with a mean ARPD slightly lower than 1%. The criteria based on the Metropolis condition (SA,

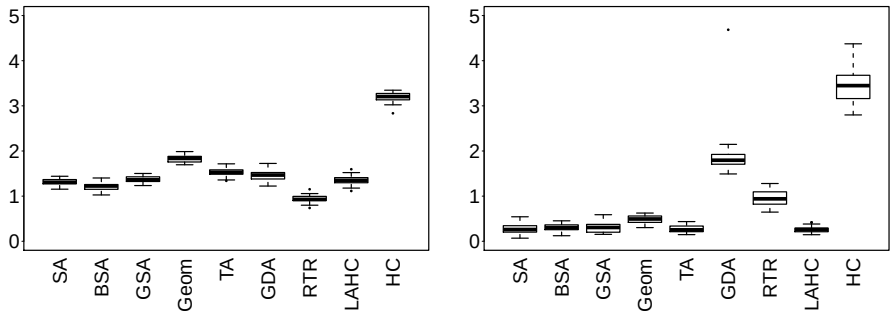


**Fig. 1.** Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on random (left plot) and structured instances (right plot).

BSA, GSA) and the LAHC algorithm obtain similar results, with mean ARPDs around 1.2 to 1.3%. Though the results are similar, BSA is consistently slightly better than the other ones. TA, GDA and the geometric criteria are worse, but still within the 2% average deviation, while HC stands around 3%. On the structured instances it is instead TA, LAHC and the family of the Metropolis criteria that obtain the best results, with average ARPDs all around 0.3%. The ARPDs among these five criteria are not statistically significantly different. The geometric criterion also obtains reasonably good results when considering the ARPD values, though from the rank-based analysis it is already clearly worse than the top-ranking group of acceptance criteria. RTR and GDA obtain solutions around 1% and 2% worse than the best known ones and, thus perform clearly worse than the other acceptance criteria. HC, as expected, is overall the worst, with ARPDs around 3 to 4%.

The difference of the results on the two scenarios can be explained by the different landscape of the instances [22,20]. The random instances present a relatively flat landscape, where it is easy to discover local optima and move through them, but difficult to converge to very good solutions. On the other hand, the landscape of the structured instances is less flat, with "deeper" local optima than in the random instances. The criteria that strengthen the intensification along the search process are the ones that apparently benefit from this landscape. RTR compares candidate solutions to the global best, making it therefore more difficult to accept a worsening solution; additionally, using a same parameter settings across all instances may make it less robust.

## 5 Experiments on the Permutation Flow-Shop Problem

In Figure 2 we report the results obtained on the PFSP-TCT on the 80 instances of the Taillard benchmark with 50 to 200 jobs. The results of the Friedman rank sum test for the nine algorithms are reported in Table 3, separated for the

**Table 2.** Results of the Friedman rank sum test for the nine algorithms on the QAP instances. Algorithms are ranked according to their results. $\Delta_R$ is the minimum rank-sum difference that indicates significant difference from the best one. Algorithms in boldface are significantly better than the following ones.

| Instance class | $\Delta_R$ | Acceptance criteria ranking |
|---|---|---|
| Random | 13.15 | **RTR** (0), BSA (33), SA (70), LAHC (80), GSA (88), GDA (115), TA (140), Geom (174), HC (200) |
| Structured | 16.08 | **TA** (0), **LAHC** (0), **SA** (11), **BSA** (16), GSA (21), Geom (81), RTR (109), GDA (135), HC (158) |

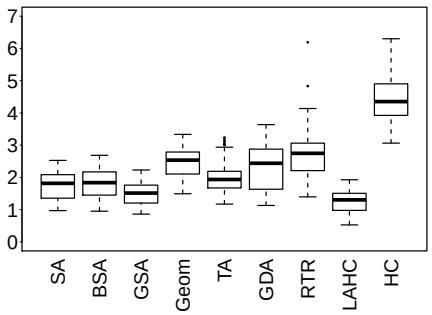


**Fig. 2.** Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on the instances `Ta031-110` of the Taillard benchmark.

three sets of instance subclasses considered (smaller than in the training set, size covered by the training set, and larger).

The results in Figure 2 for the PFSP-TCT exhibit higher variance than on the QAP, because they report results obtained on 8 subclasses of instances, with a different number of jobs and machines. Inside each instance subclass, the variance is much lower, indicating consistent results for each instance size.

Late acceptance hill climbing is the criterion that obtains clearly the best results, with an average ARPD of 1.2%. It is also more robust than the others: its worst results are below 2% of ARPD. GSA comes second best, with an average deviation of 1.5%, followed by SA and BSA (respectively 1.7% and 1.8% on average; a Wilcoxon test shows no statistically significant difference between them). TA obtains results comparable to BSA. The other criteria obtain results between 2% and 3% of ARDP, still significantly better than HC.

The different instance sizes in both the training and test sets favour the more robust solutions. GSA appears to be more robust than the original SA, thanks to the increased flexibility given by the additional parameters. Looking at the different instance subclasses, however, LAHC consistently outperforms all other acceptance criteria.

**Table 3.** Results of the Friedman rank sum test for the nine algorithms on the Taillard Benchmark. Algorithms are ranked according to their results. $\Delta_R$ is the minimum rank-sum difference that indicates significant difference from the best one. Algorithms in boldface are significantly better than the following ones.

| Instance class | $\Delta_R$ | Acceptance criteria ranking |
|---|---|---|
| `Ta001-030` | 13.57 | **LAHC** (0), GSA (30), TA (74), SA (82), BSA (87), Geom (142), RTR (193), GDA (206), HC (212) |
| `Ta031-110` | 27.58 | **LAHC** (0), GSA (94), SA (229), TA (267), BSA (279), GDA (412), Geom (455), RTR (490), HC (636) |
| `Ta111-120` | 2.93 | **LAHC** (0), GSA (11), RTR (19), GDA (31), HC (39), Geom (50), TA (63), SA (67), BSA (80) |

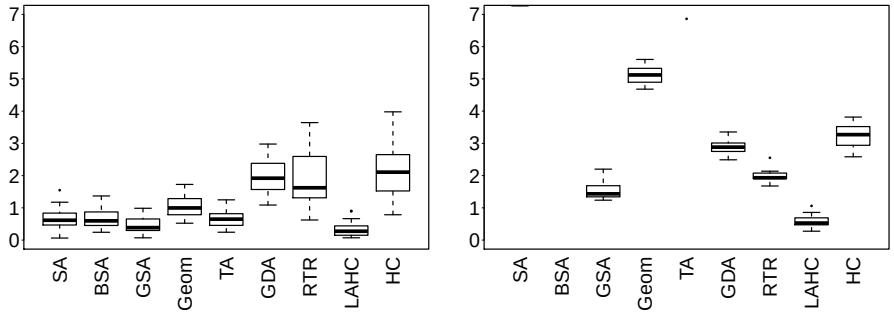


**Fig. 3.** Average Relative Percentage Deviation (ARPD) from the best known solutions obtained on the `Ta001-030` (left plot) and on the `Tai111-120` instances of the Taillard benchmark set (right plot, SA, BSA, and in part TA results are not shown as they were very poor).

We focus now on the instance subclasses not covered by the training set, either because they are too small (`Ta001-030`) or because they are too big `Tai111-120`. We can observe in Figure 3 and in Table 3 that LAHC is consistently the best performing one, followed by GSA. Overall, on the small instances all algorithms obtain results that are according to the ARPD values at least as good as on medium size instances of `Ta031-110`, with GDA and RTR being the only ones for which this is not true.

On the large instances, LAHC and GSA remain the top-performing algorithms with an average ARPD of 0.59% and 1.57%, respectively. SA and BSA instead obtain good results on the small instances, but perform very poorly on the larger ones, with ARPDs ranging around $9 - 10\%$, much worse than even HC. This effect is due to the parameters selected by the tuning phase, which are calibrated for instance sizes occurring in the training set and the given running time. The convergence behaviour of SA and BSA does not scale to large instance sizes for which the evaluation per solution is much more costly (the evaluation

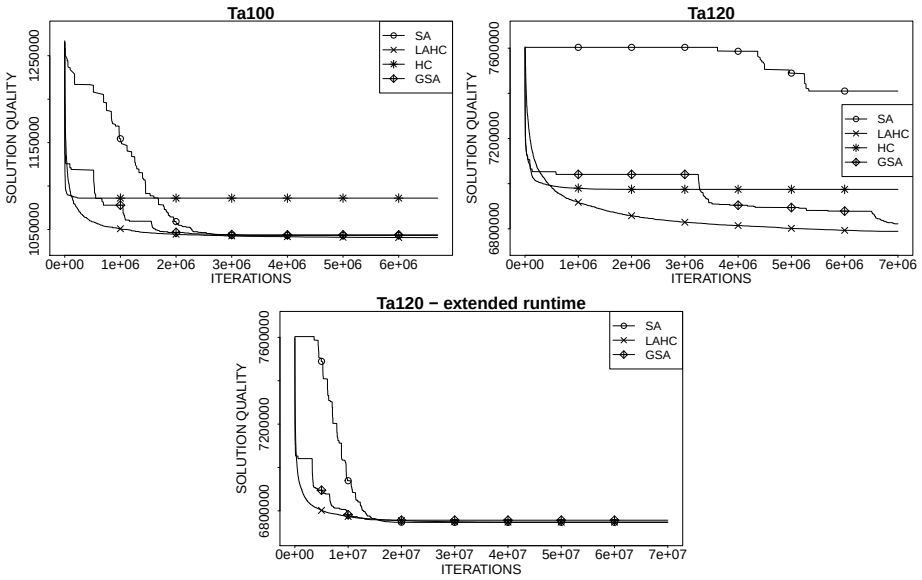


**Fig. 4.** Convergence behaviour of the SA, GSA, LAHC and HC algorithms for the `Ta100` (top left plot) and `Ta120` (top right plot) instances; in the bottom plot, the results for SA, GSA and LAHC on `Ta120` with 10× the original running time.

scales quadratically with instance size while the computation time only increases linearly). This is illustrated in Figure 4, where we compare the development of the solution quality over the number of iterations for SA, GSA, LAHC and HC on two instances: `Ta100`, whose size is $200 \times 10$ and is covered by the training set, and `Ta120`, whose size is $500 \times 20$. On `Ta100`, the four algorithms quickly discover good solutions; still, the convergence of SA is slower with respect to the other three. On `Ta120`, SA is clearly unable to converge within the originally allocated computation time. In the right plot of Figure 4 we observe the convergence of SA, GSA and LAHC on `Ta120` with a runtime ten times higher (1500s instead of 150s on that instance – HC not included in the plot): the convergence is more similar to the one observed for `Ta100`, with also SA discovering high quality solutions. In particular, GSA finds a solution very close to the best known one (6756860 vs 6755722), while SA and LAHC both find a solution of better quality than the currently best known one (6746818 and 6748131, respectively). It is interesting to note that SA has now found the best solution, while LAHC has continued improving until more than half the time available.

## 6 Conclusions

We have observed how a careful tuning of the numerical parameters is crucial to obtain good results, in terms of both solution quality and convergence. Across our

two benchmark problems, the algorithm that obtained the best results is LAHC. It exhibits a good convergence behaviour, quickly discovering good solutions in the beginning of the search, and continuously improving afterwards. It is also very robust, as it is the best performing algorithm across the whole Taillard benchmark for the PFSP-TCT, and it scales well also to instances of different sizes, unseen in the training set. Despite its simplicity (only one parameter to be tuned), LAHC makes a good use of the history of the search, as any solution it accepts is never worse than at least another one it has accepted in the past.

SA obtains overall good results, but it requires a proper tuning, as we have observed, in particular, for the large PFSP instances. It is able to obtain good results, but it might do so slowly; it is therefore advisable to tune SA for anytime behaviour [12] to obtain good results in a shorter time. BSA performs similarly to the standard SA. GSA, instead, has been shown to be flexible, outperforming SA also in terms of scalability and anytime behaviour. The geometric acceptance criterion is overall inferior to tthose derived from the original SA.

TA has obtained results overall not very different from SA. RTR has obtained good results on the random QAP instances, but was among the worse performers in the other scenarios, probably because of the fixed value of its threshold. GDA also showed a lack of flexibility, requiring a different setup and thus making its use within other algorithms more problematic.

As future work, we plan to extend the analysis to different conditions that might improve the performance of the various criteria. For example, a temperature restart, which is a common option in various SA algorithms, may change the conclusions of especially those criteria that rely on the temperature parameter. In addition, we plan to extend the set of acceptance criteria that are considered in this work and also extend the set of test problems to increase the experimental basis on which our conclusions rely. Finally, we intend to test the various acceptance criteria considering other aspects such as anytime behavior or robustness to other scenarios that differ in instance size and termination condition.

**Acknowledgments.**

We acknowledge support from the COMEX project (P7/36) within the IAP Programme of the BelSPO. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a senior research associate.

## References

1. Appleby, J., Blake, D., Newman, E.: Techniques for producing school timetables on a computer and their application to other scheduling problems. The Computer Journal 3(4), 237–245 (1961)
2. Bohachevsky, I.O., Johnson, M.E., Stein, M.L.: Generalized simulated annealing for function optimization. Technometrics 28(3), 209–217 (1986)
3. Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.S.: The quadratic assignment problem. In: Handbook of Combinatorial Optimization, vol. 2, pp. 241–338. Kluwer Academic Publishers (1998)

4. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. Tech. Rep. CSM-192, University of Stirling (2012)
5. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. European Journal of Operational Research 258(1), 70–78 (2017)
6. Chen, R.M., Hsieh, F.R.: An exchange local search heuristic based scheme for permutation flow shop problems. Applied Mathematics & Information Sciences 8(1), 209–215 (2014)
7. Dueck, G.: New optimization heuristics: the great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104(1), 86–92 (1993)
8. Dueck, G., Scheuer, T.: Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90(1), 161–175 (1990)
9. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA (2005)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
11. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58 (2016)
12. López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. European Journal of Operational Research 235(3), 569–582 (2014)
13. Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T.: From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In: Proc. LION 7, LNCS, vol. 7997, pp. 321–334. Springer, Heidelberg, Germany (2013)
14. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21, 1087–1092 (1953)
15. Moscato, P., Fontanari, J.F.: Stochastic versus deterministic update in simulated annealing. Physics Letters A 146(4), 204–208 (1990)
16. Nawaz, M., Enscore, Jr, E., Ham, I.: A heuristic algorithm for the $m$-machine, $n$-job flow-shop sequencing problem. Omega 11(1), 91–95 (1983)
17. Ogbu, F.A., Smith, D.K.: The application of the simulated annealing algorithm to the solution of the n/m/C max flowshop problem. Computers & Operations Research 17(3), 243–253 (1990)
18. Pan, Q.K., Ruiz, R.: Local search methods for the flowshop scheduling problem with flowtime minimization. European Journal of Operational Research 222(1), 31–43 (2012)
19. Pan, Q.K., Ruiz, R.: A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. Computers & Operations Research 40(1), 117–128 (2013)
20. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174(3), 1519–1539 (2006)
21. Taillard, É.D.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64(2), 278–285 (1993)
22. Taillard, É.D.: Comparison of iterative searches for the quadratic assignment problem. Location Science 3(2), 87–105 (1995)
23. Černý, V.: A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications 45(1), 41–51 (1985)