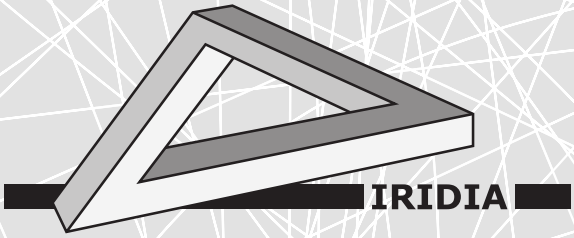
On the sensitivity of reactive tabu search to its meta-parameters

Paola Pellegrini, Franco Mascia, Thomas Stutzle, Mauro Birattari



Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

On the Sensitivity of Reactive Tabu Search to its Meta-parameters

Paola Pellegrini, Franco Mascia,
Thomas Stützle, and Mauro Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2011-025

December 2011

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/2011-025

On the Sensitivity of Reactive Tabu Search to its Meta-parameters

Paola Pellegrini[1⋆], Franco Mascia[2], Thomas Stützle[2], and Mauro Birattari[2]

[1] IFSTTAR – ESTAS, Villeneuve d'Ascq, Lille, France
[2]IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
paola.pellegrinin@ifsttar.fr; fmascia@ulb.ac.be; stuetzle@ulb.ac.be; mbiro@ulb.ac.be

Abstract. In this paper, we assess the sensitivity of reactive tabu search to its meta-parameters. With a thorough experimental analysis, based on the quadratic assignment and the maximum clique problem, we show that the performance of reactive tabu search is relatively insensitive to its meta-parameters. This is particularly evident when compared to the sensitivity of tabu search to its parameters: tabu search is rather penalized if used with sub-optimal parameter settings. Reactive tabu search does not strongly pay its high parameter robustness in terms of performance, although it does not improve the peak performance of tabu search.

1 Introduction

Tabu search (TS) is a metaheuristic that exploits the search history to direct an underlying local search. The essential idea behind TS is to forbid revisiting previously seen solutions. In practice, TS rather forbids components of the past T local search moves. In tabu search, T is a parameter called tabu list length or tabu tenure, and it is known to have a strong impact on performance. Reactive tabu search (RTS) [1–4, 6, 8, 11, 15, 16, 19] is a technique that adapts the value of T at run-time. The adaptation of the parameter T is managed by a mechanism that sits on top of the underlying search method and whose behavior in turn depends on the values of other parameters to which we refer as "meta-parameters". In a sense, RTS eliminates some parameters from TS but it introduces new meta-parameters, thus, possibly increasing the number of parameters of the underlying tabu search algorithm. This is done with the hope that it becomes easier to set the meta-parameters and that the algorithm achieves high performance regardless of the characteristics of the instances to be tackled.

An obvious question is how the meta-parameters impact on the performance of the algorithm. Often, it is tacitly assumed that parameter adaptation methods help and that their meta-parameters have a negligible impact on performance. In fact, only few articles investigate the impact of meta-parameters on performance. For what concerns RTS, the first paper proposing this method [3] devoted

⋆ Paola Pellegrini has carried our part of the study reported in this paper while working at IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium.

some experiments to the study of the impact of the meta-parameter settings on performance. The authors concluded that a 10% variation of these settings does not have any relevant effect on the results. Some work has been devoted to parameter adaptation methods applied to evolutionary algorithms [7, 10], but these methods differ very strongly from the mechanism used in RTS.

In this paper, we study the sensitivity of RTS to its meta-parameters, and we compare it with the sensitivity of TS to its parameters. Starting from the RTS algorithms proposed in the literature for tackling the quadratic assignment and the maximum clique problem, we eliminate the modules related to parameter adaptation, fix the relevant parameters, and, thus, obtain the TS algorithms for tackling the two problems. By doing this, differences in the behavior of RTS and TS are due to whether the parameter adaptation method is used or not. We show that the performance of RTS is rather insensitive to the meta-parameters. The opposite holds for TS: in some cases TS suffers a major performance degradation if inappropriate yet reasonable parameter settings are used. Moreover, the instance-based optimal parameter settings of TS vary strongly as a function of the characteristics of the instance tackled, and the adoption of sub-optimal parameter settings worsens significantly performance.

Our results also indicate that RTS with optimal meta-parameter settings does not outperform TS with optimal, instance-specific parameter settings. However, if the optimal parameter and meta-parameter settings are unknown, RTS is a safe choice for achieving high performance.

We base these conclusions on an experimental analysis on the quadratic assignment problem (QAP) for which RTS was originally proposed [3]. We tackle multiple instances with different sizes and characteristics, and we give the results of a full factorial analysis that tests several parameter and meta-parameter settings for TS and RTS, respectively. In addition, we study the main effects of meta-parameters and parameters through an ANOVA analysis. This analysis shows that, even if some interactions exist among RTS meta-parameters, they do not strongly impact on our conclusions. In TS, instead, the main effect of parameters strongly varies as a function of the instances being tackled. We replicate these analyses on the maximum clique problem (MCP), for which RTS is currently a state-of-the-art algorithm [1]. The results on the MCP support the conclusions we draw on the QAP.

The rest of the paper is organized as follows. Section 2 shortly describes the algorithms we consider, their parameters and meta-parameters. Sections 3 and 4 report the setup and the results of the experimental analyses on the QAP and the MCP, respectively. Section 5 summarizes the conclusions drawn from these analyses.

2 TS and RTS

The main goal of RTS is to adapt the tabu list length during the search process by exploiting the feedback provided by the search process itself. In particular, if the search revisits already seen solutions, this is taken as an indication of an

insufficient diversification, and, thus, the tabu list length is increased. If for a large number of local search steps no solutions are revisited, this is taken as an indication of the need of a greater intensification, and, thus, the tabu list length is reduced. If the number of visits to a same solution exceeds a predefined threshold, the algorithm is recognized to stagnate. In this case, RTS escapes the basin of attraction of the current local optimum by focusing the search on another region of the search space, either through a restart or through a perturbation. Restart means randomly selecting a new initial solution for the local search, as done by Battiti and Mascia in the RTS algorithm for the MCP [1]. Perturbation means introducing a large, random modification to the current solution, as done by Battiti and Tecchiolli in the RTS algorithm for the QAP [3].

The TS and RTS algorithms that we use in this paper differ only in the management of the parameters. As mentioned in Section 1, the parameters that are adapted in the RTS algorithms are clamped to some fixed values in the TS algorithms.

2.1 RTS_QAP and TS_QAP

RTS has been first proposed applying it to the QAP. The QAP consists in finding the minimum-cost assignment of n facilities to n locations. Each pair of locations (i, j) is separated by a distance d_{ij}. A flow f_{kl} exists between each pair of facilities (k, l). Let π_i be the facility assigned to location i, then the cost of an allocation is given by

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

The RTS_QAP algorithm that we consider in this paper is the one proposed by Battiti and Tecchiolli [3]. It relies on 2-opt moves, where a move exchanges the facilities assigned to two locations. The tabu status is associated to the assignment of facilities to locations: a move is tabu if, after the exchange, both facilities involved occupy locations that they had already occupied in the last T steps. All non-tabu moves are allowed. On the other hand, tabu moves are forbidden, unless an aspiration criterion is met: if the solution obtained by applying a tabu move has a better objective function value than the best found so far, then the move is allowed despite its tabu status. At each step, RTS_QAP selects the best allowed or aspired move and applies it. The setting of T changes as a function of the visits of already seen solutions during the search: RTS_QAP increases T by a factor $Tincr$, $Tincr > 1$, when it visits an already seen solution; RTS_QAP decreases T by a factor $Tdecr$, $0 < Tdecr < 1$, if no already seen solution is visited for a fixed number of moves. As a further means for escaping from local optima, RTS_QAP uses perturbations. A perturbation occurs as soon as the number of visits of already seen solutions is greater than a meta-parameter *chaos*. A perturbation is a sequence of randomly selected 2-opt moves, whose number is a function of the number MA of steps that are made between successive visits of already seen solutions: the larger MA, the larger the number of random 2-opt moves, that is, the perturbation size. Hence, the

perturbation size varies as a function of the evolution of the search, and, thus, it is a further parameter adapted by RTS_QAP. Together with *Tincr*, *Tdecr* and *chaos*, RTS_QAP introduces four additional meta-parameters that are used, for example, for setting the perturbation size as a function of *MA*. In the analysis presented in this paper, we focus on the two meta-parameters that we consider the most important ones due to their immediate influence on the adaptation of T (*Tincr* and *Tdecr*). In addition, we study meta-parameter *chaos* due to the strong impact of the perturbations on the performance. In summary, the parameters adapted by RTS_QAP are the tabu list length (T) and the perturbation size (p_size).

We obtain from RTS_QAP the TS_QAP algorithm by imposing static values to T and p_size. Moreover, by eliminating all the modules related to the adaptation, we eliminate also the trigger that decides when to perform a perturbation. TS_QAP performs a perturbation after each sequence of n_imp consecutive non-improving moves.

2.2 RTS_MCP and TS_MCP

RTS is currently one of the best performing approaches available for tackling the maximum clique problem (MCP). The MCP consists in finding a clique of maximum cardinality in a given graph. Let $G = (V, E)$ be a graph, with V being the set of nodes and E being the set of edges. Let $G(S) = (S, E \cap S \times S)$ be the subgraph induced by $S \subseteq V$. A clique is a set S such that $G(S)$ is complete, that is, all nodes in S are pairwise adjacent.

The RTS_MCP algorithm that we use for the MCP is described in Battiti and Mascia [1]. It relies on a local search, in which a basic move corresponds to either the addition or the removal of one node from the current clique. The tabu status is associated to nodes: a node that has been either inserted in or removed from one of the last T cliques can be neither inserted in nor removed from the current one. At each step, the algorithm evaluates all solutions in the neighborhood of the current clique, and it moves to the best non-tabu one. No aspiration criterion is applied. Furthermore, the algorithm escapes from local optima through restarts. The number of steps made before a restart is a parameter of the algorithm. This parameter is expressed as a constant, *restart*, multiplied by the size of the maximum clique found. RTS_MCP adapts the parameter T as a function of the number of visits of already seen solutions during the search. When an already seen solution is visited, RTS_MCP rises T to $\max\{T \cdot Tincr, T+1\}$. As RTS_QAP, RTS_MCP decreases T by a factor *Tdecr* if no already seen solution is visited for a fixed number of steps. Differently from RTS_QAP, the parameter *restart* is not adapted by RTS_MCP, thus, it remains constant.

We obtain the TS_MCP algorithm by suppressing the adaptation of T and by eliminating the hash-table used for recording already seen solutions.

Table 1. Sets of QAP instances tackled.

set	size	type	set	size	type
qap1	60	structured	qap7	100	structured
qap2	60	unstructured	qap8	100	unstructured
qap3	60	structured and unstructured	qap9	100	structured and unstructured
qap4	80	structured	qap10	60, 80 and 100	structured
qap5	80	unstructured	qap11	60, 80 and 100	unstructured
qap6	80	structured and unstructured	qap12	60, 80 and 100	structured and unstructured

3 Analysis on the QAP

As a first step, we analyze the sensitivity of RTS_QAP to the meta-parameters. We do this analysis using multiple sets of instances to assess the impact of both the characteristics and the heterogeneity of the set of instances. We tackle instances of three sizes, namely $n \in \{60, 80, 100\}$, and of two types, namely structured and unstructured. We created 100 instances of each size and type using the generator and the parameters described in Pellegrini et al. [18] and Hussin and Stützle [12]. In unstructured instances, the entries of both distance and flow matrices are random numbers uniformly distributed in the interval $[0, 99]$. In structured instances, the entries of the distance matrix are the Euclidean distances of points positioned in a 100×100 square according to a uniform distribution, rounded to the nearest integer. The entries of the flow matrix are assigned so that the resulting values follow the characteristics pf real-life instances, that is, the flow matrix entries have an asymmetric distribution, a significant fraction of the entries are zero (0.22) and for the non-zero entries there is a high frequency of low values and a low frequency of high values. From these instances we obtain twelve sets of instances that are shown in Table 1.

We test multiple parameter and meta-parameter settings for TS_QAP and RTS_QAP, respectively. The settings are reported in Table 2. The range of values we chose for TS_QAP's parameter settings include the values of the analogous parameter adopted in RTS_QAP while running with the default meta-parameter settings. While the frequency with which RTS_QAP uses each setting varies from instance to instance, the range of the values adopted for the parameters T and *p_size* during a run of RTS_QAP remains rather constant. The range of RTS_QAP meta-parameter settings is naturally bounded for what concerns *Tdecr*: being this a multiplicative factor used for decreasing T, it must be positive and smaller than one. Conversely, *Tincr* must be greater than one. In this case, it is not possible to identify a natural upper bound; we fixed as the upper bound 2.5. The interval we consider is large enough to contain all the values that we expect to be the best ones for RTS_QAP. Our expectation is based on the results reported by Battiti and Tecchiolli [3] and on our own previous experience. Please notice that the interval we consider includes the default value of 1.1 suggested

Table 2. Parameter and meta-parameter settings tested for the QAP.

parameter settings: TS_QAP		meta-parameter settings: RTS_QAP	
T	$1, 3, 5, 7, ..., 79$	$Tincr$	$1.1, 1.2, 1.3, ..., 2.5$
p_size	$1, 5, 10, 15, 20, 25, 30, 40, 50$	$Tdecr$	$0.1, 0.2, 0.3, ..., 0.9$
n_imp	$1, 5, 10, 20, 40, 80$	$chaos$	$1, 2, 3, 4, 5, 6$

by Battiti and Tecchiolli [3]. Moreover, we consider the granularity of 0.1 as fine enough to highlight the differences in the performance of RTS_QAP that are due to the settings of meta-parameters $Tincr$ and $Tdecr$. This granularity is the same used by Battiti and Tecchiolli in the short experimental analysis that led them to the default settings of 1.1 and 0.9 [3]. For setting the upper bound of meta-parameter $chaos$ we made a similar reasoning: its default value is three [3] and its upper bound here is set to six.

We run the algorithms on Xeon E5410 quad core 2.33GHz processors with 2x6 MB L2-Cache and 8 GB RAM, under the Linux cluster Rocks distribution CentOS version 5.3. We use as a stopping criterion a computation time of 7 seconds for instances of size 60, 15 seconds for instances of size 80 and 30 seconds for instances of size 100. This allows RTS with the default meta-parameter settings ($Tincr = 1.1$, $Tdecr = 0.9$, $chaos = 3$ [3]) to perform about $1300n$ iterations, where n is the size of the instance. We evaluate the results of RTS_QAP and TS_QAP in terms of the relative error with respect to the best-known solution on a single run per instance [5]. Moreover, we verify whether the conclusions drawn on the mean results on multiple instances are equivalent to the ones drawn on a single instance basis. For studying the results on single instances, we perform 100 runs on two randomly drawn instances from each set.

For each instance, we obtain the best-known solution by selecting the best result among the ones achieved in the following experiments: first, we perform ten runs of RTS_QAP with the default meta-parameter settings, considering runs 40 times as long as the previous mentioned times. Second, we perform ten runs of the same duration using an ILS algorithm [20] with the default parameter setting. Note that ILS [20] typically performs better than RTS_QAP on structured QAP instances. Third, we consider all the shorter runs we performed for evaluating the algorithms. All the instances used in the experimental analysis are available from Pellegrini et al. [17], together with their best-known solution value.

In this paper, we report only the results on the set of all structured and unstructured instances. The conclusions that can be drawn from these results are confirmed by the results of the whole experimental analysis, where we consider different levels of aggregation of the twelve sets. The full experimental results are available in Pellegrini et al. [17].

3.1 Sensitivity to parameters and meta-parameters on the QAP

As a first step, we conducted a landscape analysis of the meta-parameter and the parameter space. In particular, for RTS_QAP and TS_QAP, respectively, we plot

for each combination of meta-parameter and parameter values the response as measured by the mean relative error across a given set of instances w.r.t. to the best-known solutions. In particular, Figures 1 and 2 give the mean relative error incurred by RTS_QAP and TS_QAP on unstructured (Figure 1) and structured (Figure 2) instances of all sizes (sets qap10 and qap11 in Table 1) for each combination of RTS_QAP meta-parameter or TS_QAP parameter settings. We show here only the results achieved with $chaos = 3$ for RTS_QAP, and $n\_imp = 20$ for TS_QAP. The trends shown are confirmed by the analysis of all results [17].

The performance of RTS_QAP appears almost insensitive to the meta-parameter settings: the meta-parameter landscape is relatively flat for both structured and unstructured instances (upper plots of Figures 1 and 2). Only the setting $Tincr = 1.1$ leads to a noticeable worsening of the results: if $Tincr = 1.1$ and $Tdecr < 0.8$, the performance is clearly worse than the one achieved using meta-parameter settings with a higher value of $Tincr$. The default combination $Tincr = 1.1$, $Tdecr = 0.9$ is not much worse than the best settings, yet never as good as them. The behavior of RTS_QAP remains the same independently of the type of instances (size, structure) being tackled (see also Pellegrini et al. [17]).

The sensitivity of TS_QAP to its parameter settings is higher: the parameter landscape (bottom plots of Figures 1 and 2) is less flat than that of the RTS_QAP. TS_QAP has very different behavior for unstructured and structured instances. As an example, note that the adoption of a low value of p_size leads to high performance in unstructured instances, and to low performance in structured ones. Instead, the shape of the meta-parameter landscape of RTS_QAP is almost indifferent to the type of instances tackled. These observations are confirmed by the ANOVA analysis of the results we present in Section 3.2.

Figure 3 illustrates the higher sensitivity of TS_QAP. This figure shows the cumulative cost distribution of the different meta-parameter and parameter configurations extracted from the results represented in Figures 1 and 2, considering all the settings tested for RTS_QAP and TS_QAP. The cost of a configuration is measured as the average relative error on the instance set under consideration. In particular, the plots report on the x-axis the relative error, and on the y-axis the frequency by which RTS_QAP and TS_QAP with any settings in the range defined in Table 2 achieve a relative error that is smaller than or equal to the corresponding value on the x-axis. The distribution of the results of RTS_QAP is similar for structured and unstructured instances. On the other hand, the distribution of the results of TS_QAP varies quite strongly in the two cases: in structured instances, the distribution is almost equal to the one of RTS_QAP, being slightly better for small relative errors. In unstructured instances, TS_QAP obtains very good results only with a small fraction of parameter settings.

The measure in which RTS_QAP pays its low sensitivity to meta-parameter settings in terms of performance is represented in Table 3. This table reports the mean relative error made by RTS_QAP and TS_QAP with the best meta-parameter and parameter settings, respectively: we selected the best settings based on the results of the analysis, and we evaluated their performance on an additional run for each instance, or on 100 additional runs when consider-

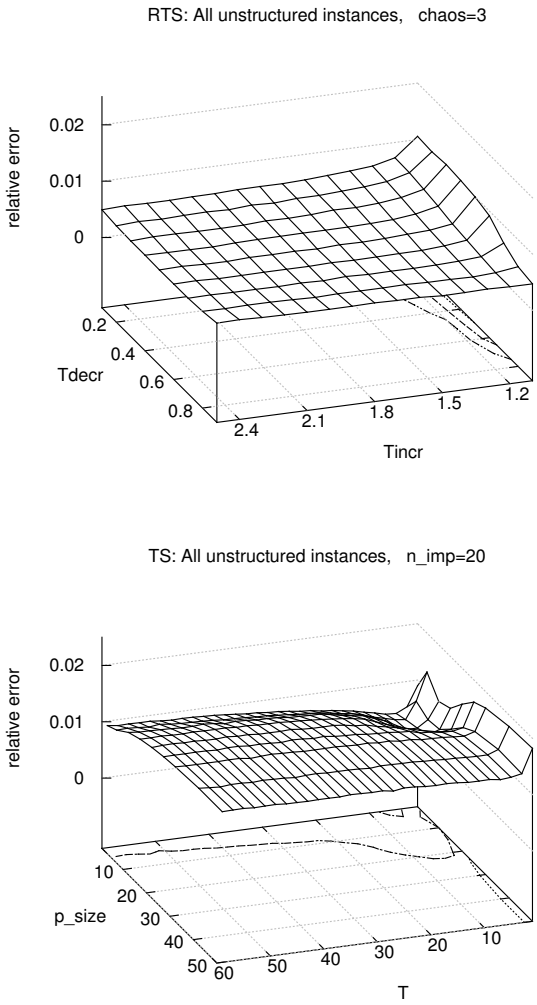


Fig. 1. Mean relative error of RTS_QAP (top plot) and TS_QAP (bottom plot) for unstructured QAP instances.

ing a single instance. In the same table, we report the best meta-parameter or parameter settings for each set of instances and for each considered single instance. When multiple instances are tackled by using the best set-specific settings, RTS_QAP performs consistently better than TS_QAP as far as unstructured instances are present: this holds for both sets including only unstructured instances, and sets including both structured and unstructured ones. When only

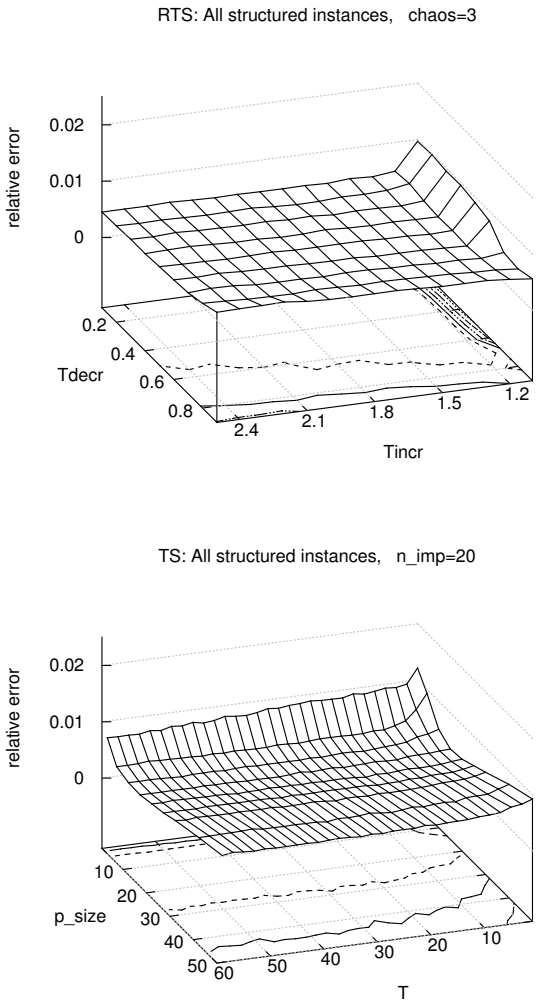


Fig. 2. Mean relative error of RTS_QAP (top plot) and TS_QAP (bottom plot) for structured QAP instances.

structured instances are to be tackled, TS_QAP is the best algorithm. In case of multiple runs on a single instance, when using instance-based optimal settings, the two algorithms are comparable on the unstructured instances, and TS_QAP achieves better results in the structured ones. Both the best parameter and the best meta-parameter settings vary quite significantly as a function of the in-

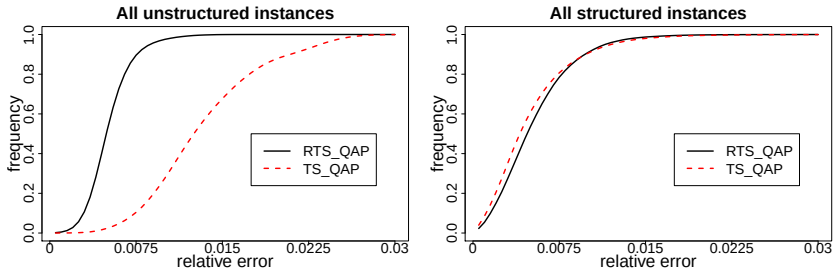


Fig. 3. Cumulative cost distribution of parameter and meta-parameter configurations for the QAP: all settings tested for RTS_QAP and TS_QAP.

stance set tackled. The size of the instances does not have a noticeable impact, neither on the results, nor on the optimal parameter settings.

3.2 ANOVA analysis on the QAP

We analyze through ANOVA the main effect of RTS_QAP meta-parameters and of TS_QAP parameters. The residuals in the original data have fat tails and depart somehow from the normal distribution. The normality of the residuals is strongly improved by a square-root transformation. The results obtained in the analyses on the original and the square-root transformed data are qualitatively equivalent. Thus, we show here the results obtained in the analysis of the original data, since they allow an easier match between the intuition and the observations. The results for the whole analysis are available in Pellegrini et al. [17].

Figures 4 and 5 report the plots of the main effects on unstructured and structured instances, respectively. The plots report the mean relative error obtained with each setting. The variability of the results of this analysis is extremely low. In principle, the plots report also the confidence intervals corresponding to the mean relative error, according to the Student t-test with confidence level 0.95. Yet, in our results the extremes of the confidence intervals coincide with the mean relative errors.

The left column of Figures 4 and 5 show that different trends appear, especially for *Tdecr*, for RTS_QAP meta-parameters when tackling unstructured and structured instances. Yet, on both unstructured and structured instances the differences in the performance achieved with different meta-parameter settings are very small. Moreover, in both unstructured and structured instances, the curves shown in the left column of Figures 4 and 5 are rather smooth, and confirm the low sensitivity to meta-parameters. Some interactions among meta-parameters exist [17]; hence, the best settings reported in Table 3 cannot be identified by looking only at the main effect plots reported here. Yet, it is quite easily remarkable that the ANOVA analysis supports the conclusion that the default meta-parameter settings of RTS_QAP proposed by Battiti and Tecchiolli [3] are

Table 3. Mean relative error across instances or runs of the best parameter and meta-parameter settings for the QAP. The difference between the results obtained by the best meta-parameter and the best parameter settings are statistically significant according to the t-test with 95% confidence level. We report in bold font the best mean for each set of instances. Below each result, we report in parenthesis the meta-parameter and parameter settings adopted. For RTS_QAP, the three values correspond to the setting of *Tincr*, *Tdecr*, and *chaos*, respectively. For TS_QAP, the three values correspond to the setting of T, *p_size*, and *n_imp*, respectively.

All sizes

all		unstructured		structured	
RTS_QAP	TS_QAP	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP
0.0044	0.0047	**0.0042**	0.0046	0.0040	**0.0028**
(2.5,0.4,1)	(7,10,20)	(2.0,0.6,5)	(7,1,40)	(1.5,0.2,2)	(33,15,40)

Individual sizes

	all		unstructured		structured	
	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP
size 60	**0.0045**	0.0055	**0.0050**	0.0055	0.0035	**0.0025**
	(2.3,0.4,2)	(7,10,40)	(2.5,0.2,1)	(5,1,10)	(2.2,0.6,5)	(49,10,20)
size 80	**0.0041**	0.0043	**0.0039**	0.0042	0.0038	**0.0028**
	(1.7,0.3,2)	(7,15,40)	(2.3,0.5,5)	(5,1,20)	(1.7,0.3,2)	(41,10,10)
size 100	**0.0038**	0.0043	**0.0034**	0.0039	0.0037	**0.0028**
	(1.5,0.5,2)	(7,15,40)	(1.9,0.7,4)	(7,1,40)	(1.4,0.4,2)	(35,10,10)

Randomly selected individual instances

	unstructured		structured	
	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP
60.a	0.0068	**0.0064**	0.0024	**0.0009**
	(2.4,0.5,3)	(5,1,40)	(2.4,0.1,1)	(21,5,1)
60.b	0.0074	**0.0051**	0.0031	**0.0018**
	(2.3,0.5,4)	(5,1,20)	(1.8,0.2,1)	(23,10,40)
80.a	**0.0053**	0.0064	0.0036	**0.0024**
	(2.0,0.5,6)	(5,1,40)	(1.3,0.5,2)	(23,20,40)
80.b	0.0060	**0.0054**	0.0059	**0.0043**
	(2.4,0.6,5)	(7,1,40)	(1.8,0.5,2)	(37,10,40)
100.a	**0.0040**	0.0054	0.0029	**0.0018**
	(2.1,0.6,5)	(7,1,40)	(1.4,0.5,1)	(51,5,5)
100.b	**0.0049**	0.0051	0.0036	**0.0022**
	(2.4,0.5,4)	(5,1,20)	(1.3,0.5,2)	(51,15,20)

not the best possible ones, even if they do not lead to a strong worsening of the solution quality.

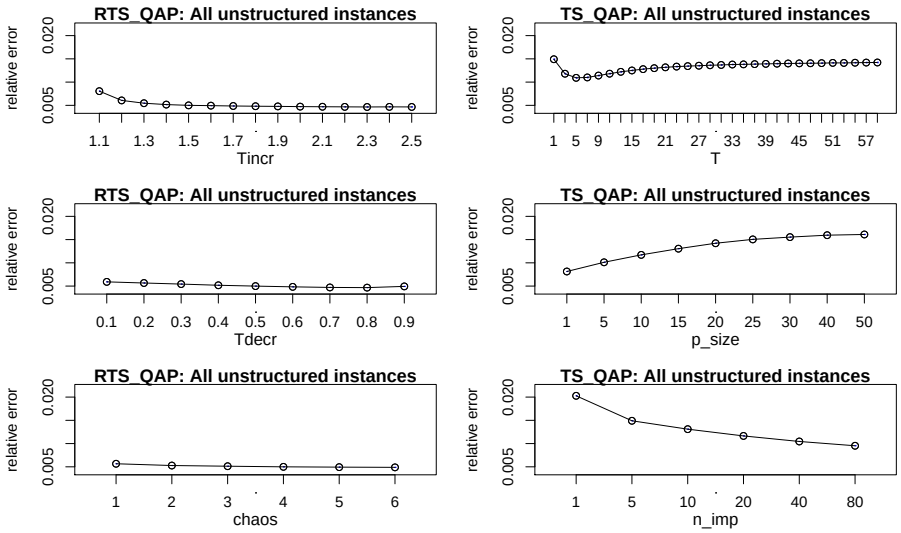


Fig. 4. Main effects on unstructured instances for RTS_QAP (left column) and TS_QAP (right column). Note that all plots use the same scale on the y-axis.

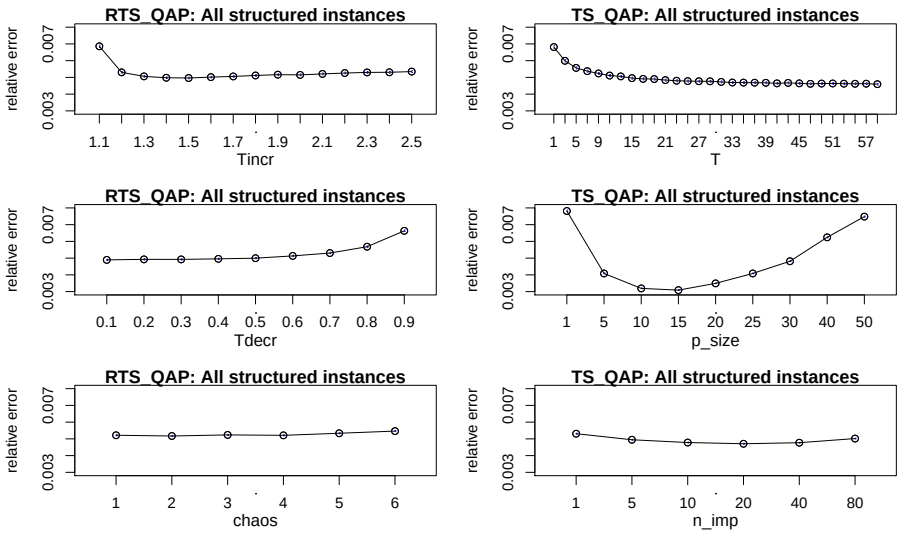


Fig. 5. Main effects on structured instances for RTS_QAP (left column) and TS_QAP (right column). Note that all plots use the same scale on the y-axis.

As it emerged from Figures 1 and 2, the results of the ANOVA analysis show that the relation between parameter settings of TS_QAP and performance vary quite strongly with the characteristics of the instances tackled (right column of Figures 4 and 5). In particular, in unstructured instances (right column of Figure 4), T must be set to a rather low value, with a small perturbation size and a large value of n_imp. Thus, TS_QAP in unstructured instances performs better when as few perturbations as possible are made, and when these perturbations are as small as possible. In structured instances (right column of Figure 5), instead, the performance is quite insensitive to the setting of T, provided that T is large enough. This is counterintuitive, since the typical expectation is that the tabu list length is the most important parameter of TS_QAP, and that the results are strongly related to its settings. This observation on the reduced impact of the tabu list length on structured instances may also explain the relatively worse performance of RTS_QAP on these instances: RTS_QAP focuses on adapting a parameter that for this instance class is not very relevant. In structured instances, the best performance is achieved by setting both p_size and n_imp to non-extreme values. This is very much in contrast with the behavior observed on unstructured instances, where in general perturbations are disadvantageous. This observation is confirmed by the interaction plots [17].

4 Analysis on the MCP

We verify the validity of the conclusions drawn for the QAP by reproducing the same analysis on the MCP, a problem for which RTS_MCP is a state-of-the-art algorithm [1].

We ran the experiments on the same hardware as the experiments for the QAP. In this analysis, we tackle a subset of the instances used in the DIMACS implementation challenge [13]. This subset includes the instances used by Battiti and Mascia [1] that can be solved under a memory limitation of 500 MB RAM. The instances tackled are listed in Table 5. We perform 100 runs for each instance, imposing two stopping criteria: the algorithm stops when either it has reached a global optimum (or a solution with the best-known solution value), or it has performed 10^8 steps. The optimal or best-known solution value of each instance is publicly available [9].

We compare the performance of each setting of TS_MCP and RTS_MCP in terms of the number of steps necessary to reach an optimal or best-known solution. Besides analyzing the results for each instance, we consider the set of all instances. In this case, we evaluate the performance in terms of the total number of steps needed for performing one run for each instance: first, we compute the mean number of steps needed to find an optimal or best-known solution for each instance; second, we sum these values for obtaining the mean number of steps necessary for solving all instances once. In the following, we will refer to this performance measure as the *number of steps to reach a bound*.

Table 4 reports the parameter and meta-parameter settings tested for TS_MCP and RTS_MCP, respectively. The settings tested for the meta-parameters of

Table 4. Parameter and meta-parameter settings tested for the MCP.

parameter settings: TS_MCP	meta-parameter settings: RTS_MCP
T $1, 3, 5, 7, ..., 49$	*Tincr* $1.1, 1.2, 1.3, ..., 2.5$
	Tdecr $0.1, 0.2, 0.3, ..., 0.9$

RTS_MCP are the same as used for the QAP. The values of T are a superset of the best static values for each instance [14]. Both for RTS_MCP and for TS_MCP we use the default setting of parameter *restart*, which is 100 [1]. As remarked in Section 2.2, the only parameter that RTS_MCP adapts is T. Thus, it is the only parameter that we consider in the study of TS_MCP.

4.1 Sensitivity to parameters and meta-parameters on the MCP

In Figure 6, we report the meta-parameter and parameter landscape analysis for RTS_MCP and TS_MCP on the set of all instances (top plots), and on instances DSJC1000.5 and MANN_a27 (center and bottom plots, respectively). The results for all the other instances are available in Pellegrini et al. [17]; they lead to the same conclusions that can be drawn based on the results reported here.

The conclusions of the analysis on the QAP are confirmed by these results: the sensitivity of TS_MCP to parameter settings is much higher than the one of RTS_MCP to meta-parameter settings. The extremely flat landscapes on the first two plots in the left column of Figure 6 indicate that RTS_MCP is insensitive to specific settings of meta-parameters. For what concerns the results achieved on instance MANN_a27, even if the meta-parameter landscape appears rather rough, the difference in the mean number of steps to reach the optimal clique size with the best and the worst setting is 31495 steps, the maximum being 116181; the difference between the 75^{th} and the 25^{th} percentile of the distribution of the results is 3946. Fixing *Tincr* to 1.1 when *Tdecr* is very low does not have a remarkable impact on the performance on the set of all instances and on DSJC1000.5, and it is a quite advantageous choice on MANN_a27. For TS_MCP, the performance is strongly worsened by inappropriate parameter settings, as shown in the right column of Figure 6. This is particularly true on the set of all instances and on DSJC1000.5. The parameter settings does not strongly impact the performance on MANN_a27: in this case, using the worst parameter settings of TS_QAP leads to an increase of 47269 of the number of steps to reach a bound with respect to using the best setting, the maximum being 121043; the difference between the the 75^{th} and the 25^{th} percentile of the distribution of the results is 11626. These differences are statistically significant according to the Wilcoxon rank-sum test with a confidence level of 0.95. The center and bottom plots show that the relation between parameter settings and performance in TS_MCP depends on the instance tackled, as it emerged in the analysis on the QAP.

The different sensitivity of TS_MCP to its parameter settings as a function of the instance tackled is very evident in Figure 7. Here, we report the cumulative

cost distribution of the RTS_MCP meta-parameter and TS_MCP parameter settings corresponding to the results shown on all instances, instance DSJC1000.5 and instance MANN_a27. The plots show on the $y-$axis the frequency with which the algorithms find the optimal or best-known solution in a given number of steps that is reported on the x-axis in a logarithmic scale. This frequency is computed considering all the settings tested, analogously to Figure 3: the cost of a configuration is the average number of steps to reach a bound on the instance or instance set under consideration.

Figure 7 shows that, even if the performance of RTS_MCP depends on the particular instance tackled, whatever the meta-parameter settings used, RTS_MCP quickly finds the target bounds on the clique size with a very high frequency. This is not the case of TS_MCP. In fact, despite the very best parameter settings allow the achievement of high performance, it is more sensitive with respect to parameters settings than RTS_MCP. In fact, the line representing RTS_MCP is always above the one representing TS_MCP: whatever the number of steps, the frequency with which RTS_MCP reaches the imposed bound is higher than the one of TS_MCP. The difference in the behavior of RTS_MCP and TS_MCP is not remarkable in instance MANN_a27, which is an exception here. Given that the difference in absolute number of steps between RTS_MCP and TS_MCP are minor and the choice of a log-scale on the x-axis, the curves representing RTS_MCP and TS_MCP appear to coincide. When the curves do not coincide, as in the QAP, RTS_MCP appears a safer option to choose if is the most appropriate parameter settings for TS_MCP for solving an instance are unknown.

Instead, if this knowledge is available, RTS_MCP does not give better results than TS_MCP. Table 5 reports the mean number of steps to reach a bound for the best RTS_MCP meta-parameter and the best TS_MCP parameter settings, together with the best settings adopted. As we did for the QAP, we selected the best settings based on the results reported in Figure 6, and we re-ran the experiments using these settings. When the appropriate settings are selected, TS_MCP outperforms RTS_MCP in 21 of 27 instances. The mean relative increase of the number of steps to reach a bound when passing from TS_MCP to RTS_MCP is 0.37, with a maximum relative increase of 3.56 in instance keller4, and a maximum relative decrease of 0.15 in instance C125.9. Let us remark here that the computation time needed for performing one step is hardly measurable. In instance keller4, for example, the increase of the mean number of steps to reach a bound of more than a factor of three corresponds to an increase of the mean solution time from 0.0001 to 0.0002 CPU seconds.

Differently, RTS_MCP is the best algorithm on the set of all instances that is, if instance-wise optimal parameter settings are not known. This result is not surprising given that (i) the best parameter and meta-parameter settings are quite different across instances, and (ii) for RTS_MCP the meta-parameter settings have relatively little impact on performance. In the set of all instances, the best meta-parameter setting is close to the default settings [1]. In this case, the median difference in the number of steps to reach a bound when using the default

Table 5. Mean number of steps to reach a bound for the best instance-wise settings of RTS_MCP and TS_MCP. The difference between the results obtained by the best meta-parameters and the best parameters are statistically significant according to the Wilcoxon rank-sum test with 95% confidence level, except for instance hamming8-4. We report in bold font the lowest mean number of steps for each instance and for the set of all instances. Below each result, we report in parenthesis the meta-parameter and parameter settings adopted. For RTS_MCP, the two values correspond to the setting of *Tincr* and *Tdecr*, respectively. For TS_MCP, the value corresponds to the setting of T.

instance	RTS_MCP	TS_MCP	instance	RTS_MCP	TS_MCP
All	**996578**	1119774	gen400_p0.9_75	1297	**901**
	(1.1, 0.8)	(9)		(2.3, 0.5)	(31)
C125.9	**125**	147	p_hat1500-1	178264	**142196**
	(1.1, 0.1)	(17)		(1.2, 0.4)	(5)
C2000.5	37817	**36556**	p_hat1500-2	**844**	848
	(1.1, 0.1)	(3)		(1.4, 0.5)	(21)
C250.9	1365	**1115**	p_hat1500-3	1672	**1133**
	(1.1, 0.8)	(17)		(1.1, 0.7)	(27)
C500.9	78059	**41951**	p_hat300-1	135	**125**
	(1.1, 0.7)	(13)		(1.1, 0.5)	(3)
DSJC1000.5	41700	**31949**	p_hat300-2	46	**35**
	(1.3, 0.3)	(3)		(2.3, 0.1)	(5)
DSJC500.5	2007	**1489**	p_hat300-3	787	**577**
	(1.1, 0.7)	(5)		(1.1, 0.6)	(15)
MANN_a27	**104910**	107614	p_hat700-1	**1359**	1556
	(1.9, 0.7)	(47)		(1.1, 0.6)	(3)
brock200_2	101054	**76410**	p_hat700-2	137	**107**
	(2.1, 0.4)	(11)		(1.1, 0.1)	(11)
brock200_4	348108	**173025**	p_hat700-3	**319**	402
	(1.9, 0.9)	(13)		(1.2, 0.3)	(9)
gen200_p0.9_44	2109	**1693**	hamming10-4	968	**799**
	(1.1, 0.7)	(21)		(1.4, 0.6)	(11)
gen200_p0.9_55	681	**422**	hamming8-4	16	16
	(2.2, 0.7)	(31)		(1.1, 0.1)	(1)
gen400_p0.9_55	35218	**27479**	keller4	164	**36**
	(1.4, 0.5)	(19)		(1.3, 0.1)	(11)
gen400_p0.9_65	1459	**1129**	keller5	3116	**3113**
	(1.9, 0.7)	(25)		(1.1, 0.6)	(13)

and the best setting ($Tincr = 1.1$ and $Tdecr = 0.8$) is very small: according to the Wilcoxon rank-sum test with confidence level 0.95, the condifence interval is $(-4943, 2150)$.

4.2 ANOVA analysis on the MCP

The ANOVA analysis shows the main effect of the meta-parameters of RTS_MCP and of the parameter of TS_MCP. Figures 8 and 9 depict the mean of the logarithm of the number of steps to reach a bound computed on the results shown in Section 4.1, for RTS_MCP and TS_MCP, respectively. The plots report also the confidence intervals according to the Student t-test with confidence level 0.95. When the confidence intervals are not visible, it means that the variability of the results is extremely low. We applied a logarithmic transformation to the results, so as to increase the normality of the residuals.

Figure 8 shows that different meta-parameter settings have no remarkable impact on the performance. For TS_MCP (Figure 9) the appropriate settings are quite different on the three sets of instances, as reported in Table 5. For TS_MCP, being T the only parameter considered, there are no interactions to accout for, and the indications of the ANOVA results correspond to those of the analysis reported in Section 4.1. As for the QAP, an inappropriate TS_MCP parameter setting has a great impact on performance: selecting an inappropriate setting may imply an increase in the number of steps to reach a bound of a factor of 188.2 for instance DSJC1000.5 and 1.63 for instance MANN_a27 (remark that the plots report the logarithm of the number of steps to reach a bound on the y-axis). Selecting an inappropriate meta-parameter setting for RTS_MCP may imply the increase of the number of steps to reach a bound of a factor of 1.6 for instance DSJC1000.5 and 1.37 for instance MANN_a27. Some interactions exist between RTS_MCP meta-parameters [17], but they do not impact the conclusions on the appropriate settings.

5 Conclusion

In this paper, we have compared the sensitivity of RTS to its meta-parameters with the sensitivity of TS to its parameters. We made this comparison on two combinatorial optimization problems, namely the QAP and the MCP using instances with different characteristics. We also performed an ANOVA analysis for studying the main effect of the parameters of TS and the meta-parameters of RTS, as well as their interactions [17].

The results of these analyses lead to the same conclusion: RTS is less sensitive than TS to settings of its parameters, and there are no significant interactions according to the ANOVA analysis. We measured the sensitivity by observing the difference in the performance achieved by the algorithms with different meta-parameter (RTS) or parameter (TS) settings. If we consider only the best settings for TS and RTS, in the QAP, TS performs better than RTS when only structured instances are tackled, and the opposite holds when also unstructured instances

are involved in the computation. In the MCP, TS typically performs better than RTS when the best settings are selected on an instance basis, and the opposite holds when the best setting is selected across multiple instances. If non-optimal parameter settings are fixed for TS, the performance can strongly worsen both on single and across multiple instances. This is not the case if non-optimal meta-parameter settings are fixed for RTS. Moreover, the best parameter setting for TS is strongly dependent on the characteristics of the instances tackled. Instead, in RTS, many meta-parameter settings perform similarly good on all the instances tested.

In the analysis for the QAP, we get to conclusions that are rather different from those drawn by Battiti and Tecchiolli [3]: from our results we would suggest to take a value larger than the originally suggested value of 1.1 for $Tincr$, and a value smaller than the originally suggested one of 0.9 for $Tdecr$. In particular, $Tincr = 2.0$ and $Tdecr = 0.5$ are good settings across all the sets of instances considered. It is not possible to identify a best overall setting for *chaos*, where the default value is a good compromise. In the analysis for the MCP, instead, the default values proposed by Battiti and Mascia [1] ($Tincr = 1.1$ and $Tdecr = 0.9$) turned out to be very good on the set of all instances.

Our results show that, if one aims at generally good results, or if it is not clear what are the characteristics of the instances that need to be considered for determining the appropriate parameter settings for TS, RTS is a very good option, and it can be used without any previous tuning of the meta-parameters: despite the default meta-parameter settings are typically not the best ones, their use does not have a strong negative impact on the performance of the algorithm. On the other side, they also show that there is potential for the usage of parameter selection strategies where, depending on instance characteristics, fixed instance-specific parameter values are chosen. This is an option we will explore further in follow-up research.

Acknowledgements

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Mauro Birattari and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates. The work of Paola Pellegrini has been partially funded by a Bourse d'excellence Wallonie-Bruxelles International.

References

1. Battiti, R., Mascia, F.: Reactive and dynamic local search for max-clique: Engineering effective building blocks. Computers & Operations Research 37(3), 534–542 (2010)
2. Battiti, R., Protasi, M.: Reactive local search techniques for the maximum k-conjunctive constraint satisfaction problem (max-k-ccsp). Discrete Applied Mathematics 96-97, 3–27 (1999)

3. Battiti, R., Tecchiolli, G.: The reactive tabu search. ORSA Journal on Computing 6(2), 126–140 (1994)
4. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Operations research/Computer Science Interfaces, Springer Verlag (2008)
5. Birattari, M.: On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs? Tech. Rep. 2004-01, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2004)
6. Chiang, W., Russell, R.: A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS Journal on Computing 9, 417–930 (1997)
7. Costa, L.D., Fialho, Á., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Ryan, C., Keijzer, M. (eds.) Genetic and Evolutionary Computation Conference (GECCO). pp. 913–920. ACM, Atlanta, USA (2008)
8. Datta, T., Srinidhi, N., Chockalingam, A., Rajan, B.: Random-restart reactive tabu search algorithm for detection in large-mimo systems. Communications Letters, IEEE 14(12), 1107–1109 (2010)
9. DIMACS Center: DIMACS implementation challenges. http://dimacs.rutgers.edu/Challenges/ (2011)
10. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence 60(1-2), 25–64 (2010)
11. Fink, A., Vo[ss], S.: Solving the continuous flow-shop scheduling problem by metaheuristics. European Journal of Operational Research 151(2), 400–414 (2003)
12. Hussin, M., Stützle, T.: Tabu search vs. simulated annealing for solving large quadratic assignment instances. Tech. Rep. 2010-20, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2010)
13. Johnson, D.S., Trick, M.A. (eds.): Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 28. American Mathematical Society, Providence, RI, USA (1993)
14. Mascia, F., Pellegrini, P., Stützle, T., Birattari, M.: An analysis of parameter adaptation in reactive tabu search. Tech. Rep. 026, IRIDIA-CoDE, Université Libre de Bruxelles, Brussels, Belgium (2011)
15. Nanry, W., Barnes, J.: Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research Part B: Methodological 34(2), 107 – 121 (2000)
16. Osman, I., Wassan, N.: A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. Journal of Scheduling 5(4), 263–285 (2002)
17. Pellegrini, P., Mascia, F., Stützle, T., Birattari, M.: Companion of: A detailed study on the meta-parameters of reactive tabu search. http://iridia.ulb.ac.be/supp/IridiaSupp2011-26/ (2011), IRIDIA Supplementary page
18. Pellegrini, P., Stützle, T., Birattari, M.: A critical analysis of parameter adaptation in ant colony optimization. Swarm Intelligence 6(1), 23–48 (2012)
19. Russell, R., Chiang, W., Zepeda, D.: Integrating multi-product production and distribution in newspaper logistics. Computers & Operations Research 35, 1576–1588 (2008)
20. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174(3), 1519–1539 (2006)

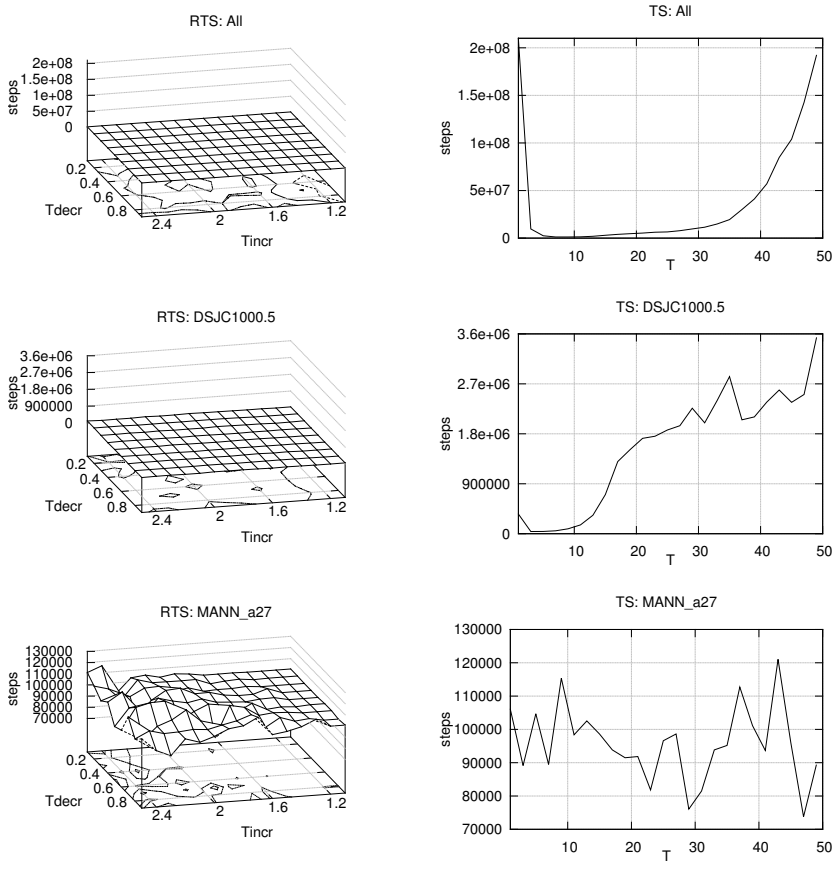


Fig. 6. Mean number of steps to reach the best-known or optimal clique size by RTS_MCP (left column) and TS_MCP (right column) on all MCP instances (top), instance DSJC1000.5 (middle) and instance MANN_a27 (bottom)

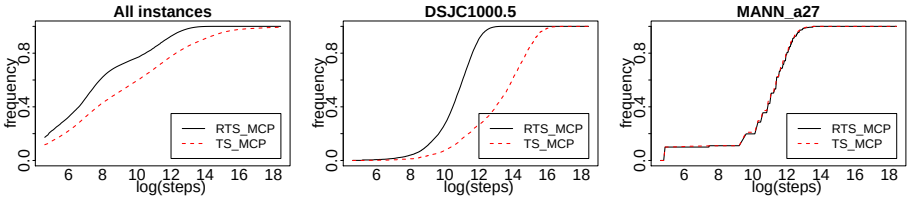


Fig. 7. Cumulative cost distribution of parameter and meta-parameter configurations for the MCP: all settings tested for RTS_MCP and TS_MCP.

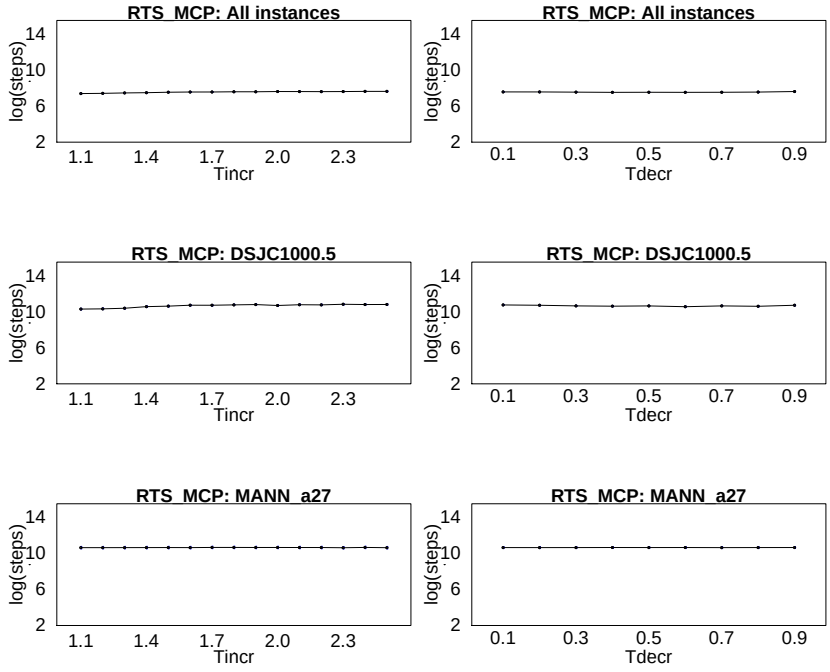


Fig. 8. Main effect in RTS_MCP. Set of all MCP instances (top), and instances DSJC1000.5 (middle) and MANN_a27 (bottom). Note that all plots use the same scale on the y-axis as the ones in Figure 9.

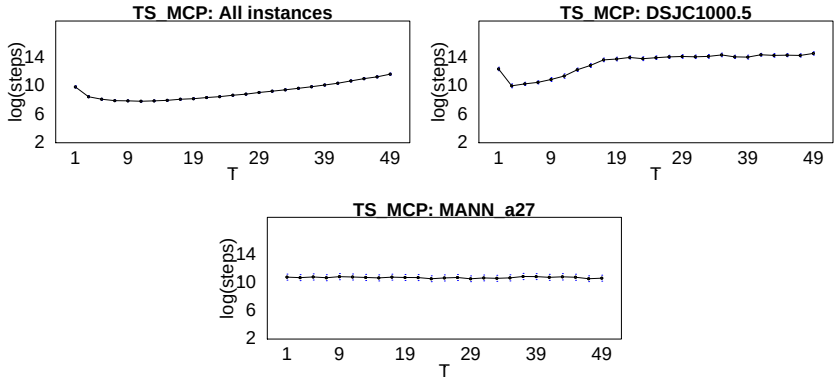


Fig. 9. Main effect in TS_MCP. Set of all MCP instances (top), and instances DSJC1000.5 (middle) and MANN_a27 (bottom). Note that all plots use the same scale on the y-axis as the ones in Figure 8.