Soft Comput (2014) 18:2177–2190
DOI 10.1007/s00500-013-1192-6

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

Paola Pellegrini · Franco Mascia · Thomas Stützle · Mauro Birattari

Published online: 5 December 2013

Abstract In this paper, we assess the sensitivity of reactive tabu search to its meta-parameters. Based on a thorough experimental analysis of reactive tabu search applications to the quadratic assignment and the maximum clique problem, we show that its performance 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.

Keywords Reactive tabu search · Sensitivity · Meta-parameters · Quadratic assignment problem · Maximum clique problem

Communicated by A.-M. Cretu.

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

P. Pellegrini (✉)
IFSTTAR, ESTAS, Univ. Lille Nord de France, Villeneuve d'Ascq, Lille, France
e-mail: paola.pellegrini@ifsttar.fr

F. Mascia · T. Stützle · M. Birattari
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
e-mail: fmascia@ulb.ac.be

T. Stützle
e-mail: stuetzle@ulb.ac.be

M. Birattari
e-mail: mbiro@ulb.ac.be

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 TS, 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) 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 tabu search 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 of 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. Battiti and Tecchiolli (1994) proposed RTS, and since then several authors applied it with success to a wide variety of combinatorial optimization problems (Arntzen et al. 2006; Battiti and Bertossi 1999; Battiti et al. 2008; Battiti and Mascia 2010; Battiti and Protasi 1999; Chiang and Russell 1997; Datta et al. 2010; Fink and Voß 2003; Nanry and Barnes 2000; Osman and Wassan 2002; Russell et al. 2008).

A natural 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 (Battiti and Tecchiolli 1994) devoted 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. In the research on RTS, the meta-parameter settings recommended by Battiti and Tecchiolli (1994) are considered in several follow-up papers. For example, Battiti (1996) uses these settings in a comparison between two TS variants and RTS, without further investigating the impact meta-parameters might have. Battiti and Brunato (2005) adopt in their discussion of the design and implementation of RTS again the same meta-parameter settings. Although this paper focuses on the design and implementation of RTS, the authors do not mention the need for choosing appropriate meta-parameter settings. Some recent work has been devoted to study the impact of meta-parameters in parameter adaptation methods applied to evolutionary algorithms.[1] DaCosta et al. (2008) report the results of a sensitivity analysis of an adaptation method named dynamic multi-armed bandits to its meta-parameter settings, and compare it to the sensitivity of the underlying evolutionary algorithm to its parameter settings. Fialho et al. (2010) propose a sensitivity analysis of different adaptation methods, and they remark that these methods may actually be somehow sensitive to their meta-parameter settings, which they select through a tuning procedure named F-Race (Birattari et al. 2002). Recently, Maturana et al. (2012) explicitly state that the parameter adaptation methods are *supposed to be* rather insensitive to their meta-parameter settings, but for guaranteeing the efficacy of the methods, they should be off-line tuned through a tool for automatic algorithm configuration (Hoos 2012). Stützle et al. (2012) discuss parameter adaptation techniques for ant colony optimization (ACO) algorithms and show the usefulness of pre-scheduled parameter variations. Pellegrini et al. (2012) study the impact of online parameter adaptation versus off-line tuning on ACO algorithm performance and conclude that, in the context they studied, off-line tuning was more beneficial than having an online parameter adaptation. However, parameter adaptation methods for evolutionary algorithms or ACO differ very strongly from the mechanism used in RTS. Hence, it is not possible to generalize the conclusions drawn in the literature to RTS.

In this paper, we try to answer two research questions. The first is to understand how sensitive is RTS to its meta-parameters. The second is to compare the sensitivity of the meta-parameters of RTS to the original parameters of the underlying TS algorithm. 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 so, 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 plausible 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 performance significantly.

As a further contribution, the analysis of our results allows the observation of the relationship between the performance of RTS and TS when both are run with their optimal settings. In particular, 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 (Battiti and Tecchiolli 1994). 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 (Battiti and Mascia 2010). 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 as well as 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 our conclusions 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 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 the tabu list length is reduced.

[1] For a general discussion of parameter adaptation methods in evolutionary computation, we refer the interested reader to Eiben et al. (2007).

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 perturbation or through a restart. Perturbation means introducing a large, random modification to the current solution, as done by in the RTS algorithm for the QAP (Battiti and Tecchiolli 1994). Restart means randomly selecting a new initial solution for the local search, as done in the RTS algorithm for the MCP (Battiti and Mascia 2010).

The TS and RTS algorithms that we use in this paper differ only in the management of the parameters. As mentioned in Sect. 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

Reactive tabu search 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 (1994). 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

Reactive tabu search is currently one of the best performing algorithms available for tackling the 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 (2010). 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.

3 Analysis on the QAP

As a first step, we analyze the sensitivity of RTS_QAP to its meta-parameters. We use 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, $n \in \{60, 80, 100\}$, and of two types, structured and unstructured. We created 100 instances of each size and type using the generator and the parameters described in Pellegrini et al. (2012) and Hussin and Stützle (2010). 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 of 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. As a result, 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, which are reported in Table 2. The range of values we chose for TS_QAP's parameter settings include the values of the analogous parameter used in RTS_QAP when run with the default meta-parameter settings. While the frequency with which RTS_QAP uses each parameter 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 is rather similar. The range of RTS_QAP's meta-parameter settings is naturally bounded for what concerns *Tdecr*: since it is a multiplicative factor used for decreasing T, it must be positive and smaller than one. Conversely, *Tincr* must be greater than one. For *Tincr*, it is not possible to identify a natural upper bound and we fixed as the upper bound 2.5. The interval of values for *Tincr* 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 (1994) and on our own previous experience; furthermore, the interval we consider also includes the default value of 1.1 suggested by Battiti and Tecchiolli (1994). Moreover, we consider the granularity of 0.1 as fine enough to highlight differences in the performance of RTS_QAP that are due to the settings of meta-parameters *Tincr* and *Tdecr*. This granularity is also the same used by Battiti and Tecchiolli (1994) in the experimental analysis that led them to the default settings of 1.1 and 0.9. For setting the upper bound of meta-parameter *chaos* we made a similar reasoning: its default value is three (Battiti and Tecchiolli 1994) and its upper bound here is set to six.

Table 1 Sets of QAP instances tackled

Set	Size	Type
qap1	60	Structured
qap2	60	Unstructured
qap3	60	Structured and unstructured
qap4	80	Structured
qap5	80	Unstructured
qap6	80	Structured and unstructured
qap7	100	Structured
qap8	100	Unstructured
qap9	100	Structured and unstructured
qap10	60, 80 and 100	Structured
qap11	60, 80 and 100	Unstructured
qap12	60, 80 and 100	Structured and unstructured

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

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

We run the algorithms on Xeon E5410 quad core 2.33 GHz processors with 2×6 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 s for instances of size 60, 15 s for instances of size 80 and 30 s for instances of size 100. This allows RTS with the default meta-parameter settings ($Tincr = 1.1$, $Tdecr = 0.9$, $chaos = 3$ Battiti and Tecchiolli 1994) to perform about $1{,}300\ n$ 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 (Birattari 2004). 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 previously mentioned times. Second, we perform ten runs of the same duration using an ILS algorithm (Stützle 2006) with the default parameter setting. Note that ILS (Stützle 2006) 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. (2011), together with their best-known solution value.

In this paper, we report only the results obtained 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. (2011).

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 settings 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, Figs. 1 and 2 give the mean relative error obtained by RTS_QAP and TS_QAP on unstructured (Fig. 1) and structured (Fig. 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 (Pellegrini et al. 2011).

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 Figs. 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, even if 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. 2011).

The sensitivity of TS_QAP to its parameter settings is higher: the parameter landscape (bottom plots of Figs. 1 and 2) is less flat than that of 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, which we present in Sect. 3.2.

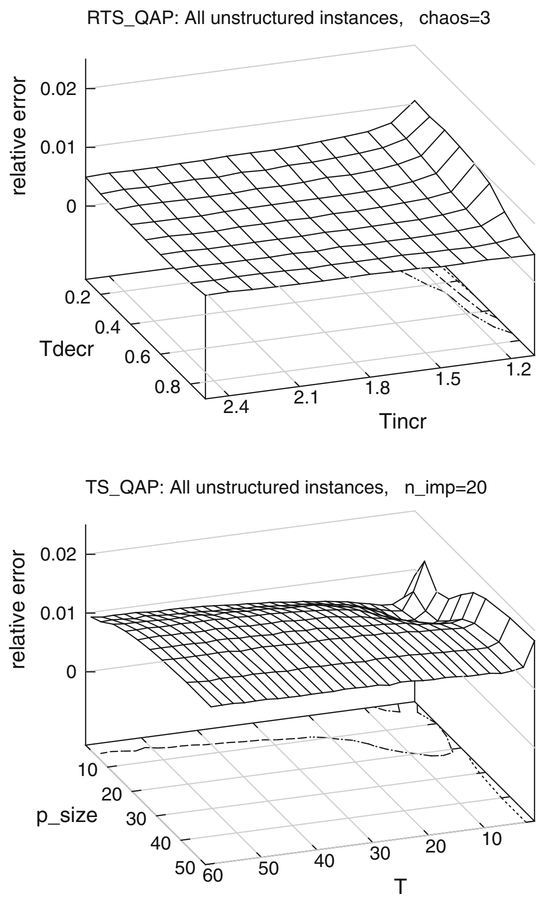


Fig. 1 Mean relative error of RTS_QAP (*top plot*) and TS_QAP (*bottom plot*) for unstructured QAP instances

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 presented in Figs. 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 ratio of RTS_QAP and TS_QAP settings that achieve a relative error that is smaller than or equal to the corresponding value on the x-axis. The distribution of this ratio for RTS_QAP is similar for structured and unstructured instances. Instead, the distribution of this ratio for 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 extent to which RTS_QAP pays its low sensitivity to meta-parameter settings in terms of performance is represented in Tables 3 and 4. These tables report the mean relative

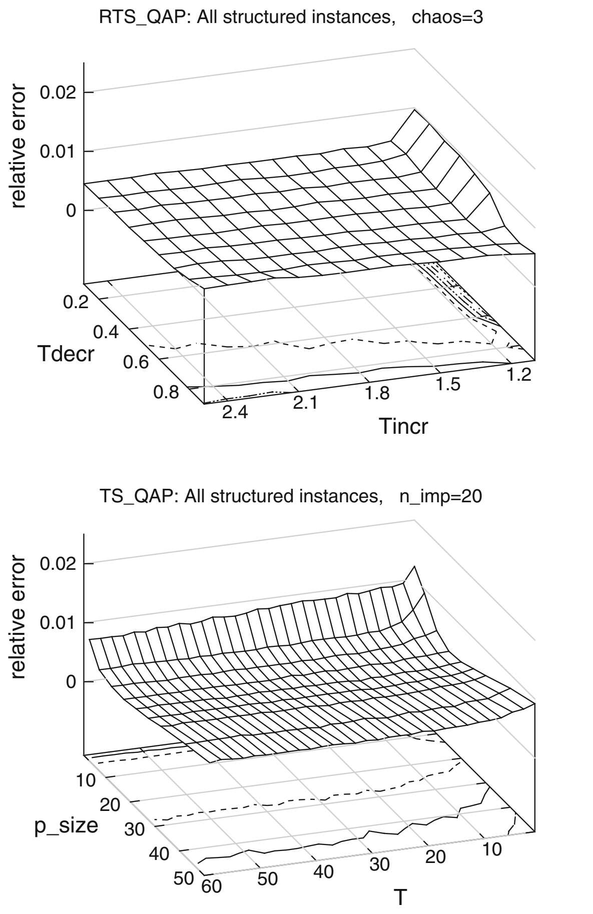


Fig. 2 Mean relative error of RTS_QAP (*top plot*) and TS_QAP (*bottom plot*) for structured QAP instances

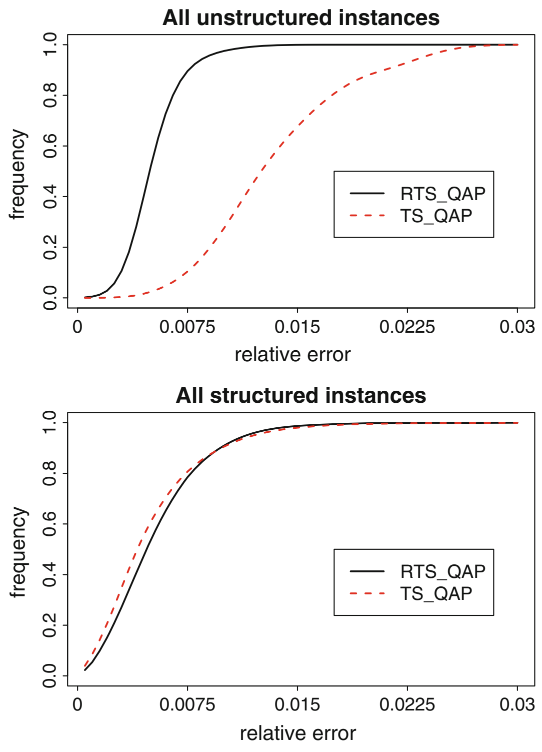


Fig. 3 Cumulative cost distribution of parameter and meta-parameter configurations for the QAP: all settings tested for RTS_QAP and TS_QAP

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 considering a single instance. In the same tables, 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 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 reach comparable results on the unstructured instances, and TS_QAP achieves better results than RTS_QAP in the structured ones. Both the best parameter and the best meta-parameter settings vary strongly as a function of the instance set tackled. No particular trend exists for the best meta-parameters; this probably depends on the fact that many meta-parameter settings reach similar performance, and, hence, any minor difference can have a strong impact on our identification of the best one. For the parameters, instead, some trend exists for T and *p_size*: the best value for T is low for the sets of all and the set of unstructured instances (between 5 and 7), while it is high for the sets of structured instances (>33); *p_size* is low for the sets of all and the set of structured instances ($=1$), while it is higher for the sets of unstructured instances (between 10 and 15). The same trends exist for the single instances shown in Table 4, even if they are slightly less marked. 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

The results reported in Sect. 3.1 show that the type of instances tackled has a strong impact on the response of RTS_QAP and TS_QAP to meta-parameters and parameters, respectively. Instead, the size of the instances does not appear to be very relevant here. In this section, we assess through a two-way analysis of variance (ANOVA analysis) the main effect of RTS_QAP meta-parameters and TS_QAP parameters in unstructured and structured instances. Through this analysis, we study the effect of the meta-parameter and parameter settings on the performance of the algorithms. In particular, the ANOVA analysis highlights the impact of each meta-parameter and parameter on the performance of

Table 3 Mean relative error across instances of the best parameter and meta-parameter settings for the QAP

	all instances		unstructured		structured	
	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP	RTS_QAP	TS_QAP
All sizes	**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						
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)

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. A star following the mean relative error indicates that the difference between the two algorithms is statistically significant according to the Wilcoxon rank-sum test with 95 % confidence level

RTS_QAP and TS_QAP, respectively, as well as the existing interactions.

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 an ad-hoc data transformation for each set of results. For ease of visualization, we report here the results obtained on the transformed data, after applying the reverse transformation. For example, we first applied a square root transformation to the results of RTS_QAP on structured instances, for satisfying the hypotheses necessary for the ANOVA analysis. Then, we applied the reverse transformation (i.e., the square transformation) to the results of the analysis for plotting the main effects identified. In this way, we can easily observe whether the impact of meta-parameter and parameter settings in terms of relative error made by RTS_QAP and TS_QAP, respectively, matches the intuition. At the same time, we can ensure the validity of the results since the transformed data verify the assumptions for an ANOVA analysis. The plots obtained for the transformed data are available in Pellegrini et al. (2011).

Figures 4 and 5 report the plots of the main effects on the sets of all 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, the variability here is so small that they are not even visible in the plot.

The top plots of Figs. 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 top plots of Figs. 4 and 5 are rather smooth, and confirm the low sensitivity to meta-parameters. Some interactions among meta-parameters exist (Pellegrini et al. 2011); hence, the best settings reported in Tables 3 and 4 cannot be identified by looking only at the main effect plots reported here. However, 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 (1994) are not the best possible ones, even if they do not lead to a strong worsening of the solution quality.

As it emerged from Figs. 1 and 2, the results of the ANOVA analysis show that the relationship between parameter settings of TS_QAP and performance depends strongly on the characteristics of the instances tackled (bottom plots of Figs. 4 and 5). In particular, in unstructured instances (bottom plot of Fig. 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 few perturbations are made, and when these perturbations are small. In structured instances (bottom plot of Fig. 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 perfor-

Table 4 Mean relative error across runs of the best parameter and meta-parameter settings for the QAP

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

We report in bold font the best mean for each instance. 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. A star following the mean relative error indicates that the difference between the two algorithms is statistically significant according to the Wilcoxon rank-sum test with 95 % confidence level

mance of RTS_QAP on these instances: RTS_QAP focuses on adapting a parameter whose value for this instance class has not a strong impact. 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 (Pellegrini et al. 2011).

4 Analysis on the MCP

We verify the validity of the conclusions drawn for the QAP by applying the same analysis to the MCP, a problem for which RTS_MCP is a state-of-the-art algorithm (Battiti and Mascia 2010).

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 (Johnson and Trick 1993). This subset includes the instances used by Battiti and Mascia (2010) that can be solved under a memory limitation of 500 MB RAM. The instances tackled are listed in Table 6. We perform 100 runs for each instance, imposing two alternative stopping criteria: the algorithm stops when either it has reached a specific bound on the solution quality value (here corresponding to the known optimal value or the best known solution value if optima are not available), or it has performed 10^8 steps. The optimal or best-known solution value of each instance is publicly available (DIMACS Center 2011).

We compare the performance of each setting of TS_MCP and RTS_MCP in terms of the number of steps necessary to reach the solution quality bound. 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 the solution quality bound; second, we sum these values for obtaining the mean number of steps necessary for solving all instances once.

Table 5 reports the parameter and meta-parameter settings tested for TS_MCP and RTS_MCP, respectively. The settings tested for the meta-parameters of RTS_MCP are the same used for the QAP. The values of T are a superset of the best static values for each instance (Mascia et al. 2013). Both, for RTS_MCP and for TS_MCP, we use the default setting of parameter *restart*, which is 100 (Battiti and Mascia 2010). As remarked in Sect. 2.2, the only parameter that RTS_MCP adapts is T. Thus, it is also the only parameter that we consider in the study of TS_MCP.

4.1 Sensitivity to parameters and meta-parameters on the MCP

In Figs. 6, 7 and 8, we report the meta-parameter and parameter landscape analysis for RTS_MCP and TS_MCP on the set of all instances (Fig. 6), and on instances DSJC1000.5 and gen400_p0.9_75 (Figs. 7, 8, respectively). The results for all the other instances are available in Pellegrini et al. (2011); they lead to the same conclusions as those 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 top plots of Figs. 6, 7 and 8 indicate that RTS_MCP is insensitive to specific settings of meta-parameters. For TS_MCP, the performance is strongly worsened by inappropriate parameter settings, as shown in the bottom plots of Figs. 6, 7 and 8. The bottom plots of Figs. 7 and 8 show that the relationship between parameter settings and performance in TS_MCP depends on the instance tackled, analogous to what was found in the analysis on the QAP.

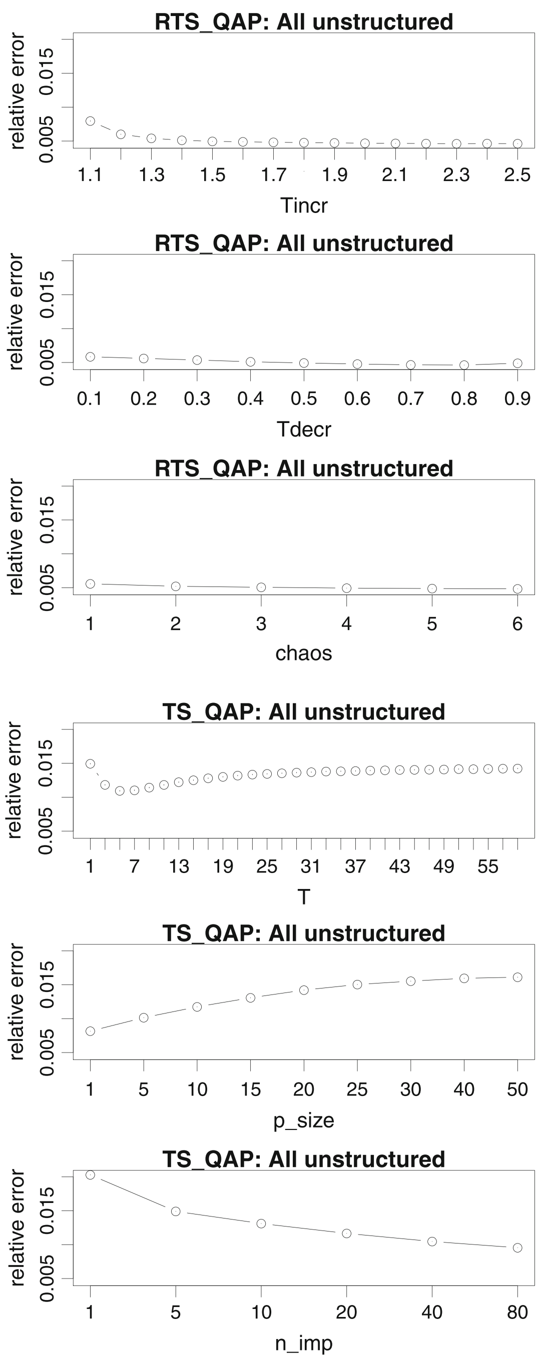


Fig. 4 Main effects on unstructured instances for RTS_QAP (*top*) and TS_QAP (*bottom*). Note that all plots use the same scale on the *y*-axis

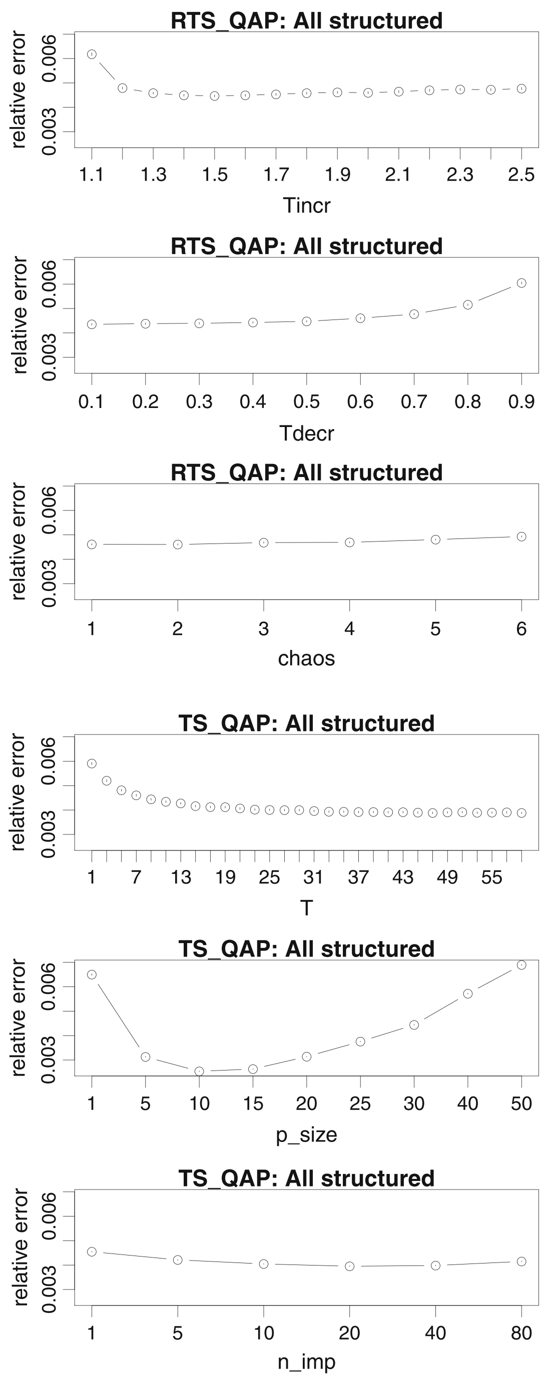


Fig. 5 Main effects on structured instances for RTS_QAP (*top*) and TS_QAP (*bottom*). Note that all plots use the same scale on the *y*-axis

In Fig. 9, we report the cumulative cost distribution of the RTS_MCP meta-parameter and TS_MCP parameter settings for instance DSJC1000.5 and instance gen40_p0.9_75. The plots show the frequency with which the algorithms reach the solution quality bound in dependence of the number of steps. This frequency represents the ratio of parameter and meta-parameter settings that can solve the specific instance in dependence of the average number of steps to reach the solution quality bound (given in the *x*-axis). It is computed considering all parameter settings tested, analogously to Fig. 3: the cost of a parameter configuration is here given by the average number of steps to reach the solution quality bound on the instance under consideration.

Figure 9 shows that, even if the performance of RTS_MCP depends on the particular instance being tackled, RTS_MCP quickly finds the target bounds on the clique size with a very high frequency. This is not the case of TS_MCP: although the best parameter settings yield high performance,

Table 5 Parameter and meta-parameter settings tested for the MCP

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

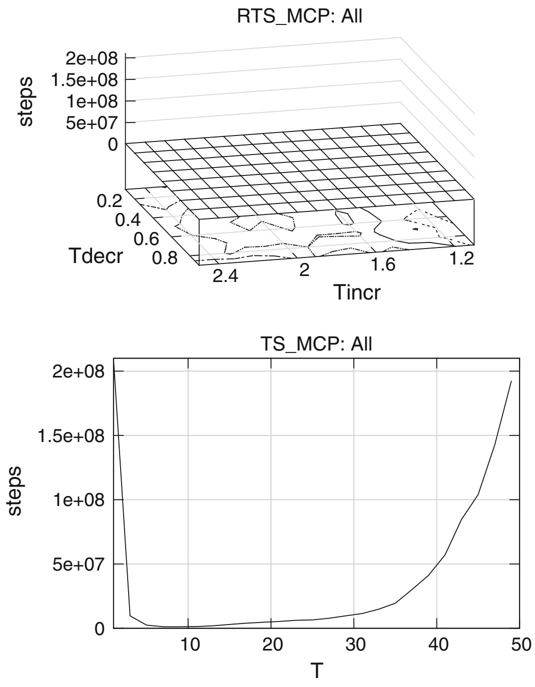


Fig. 6 Mean number of steps to reach the solution quality bound by RTS_MCP (*top*) and TS_MCP (*bottom*) on all MCP instances

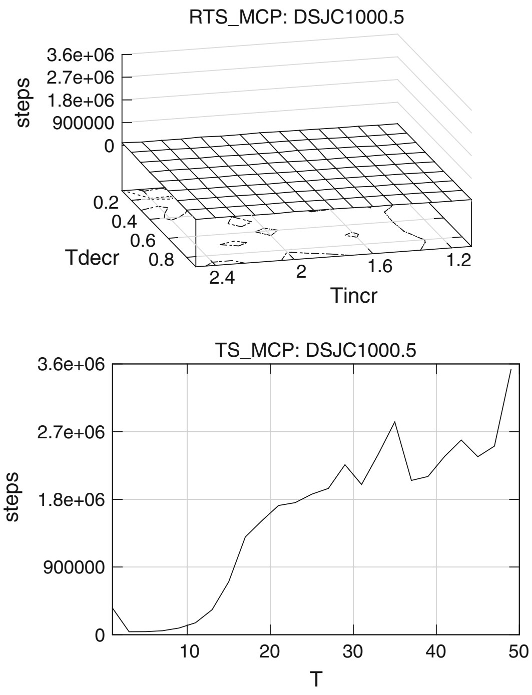


Fig. 7 Mean number of steps to reach the solution quality bound by RTS_MCP (*top*) and TS_MCP (*bottom*) on instance DSJC1000.5

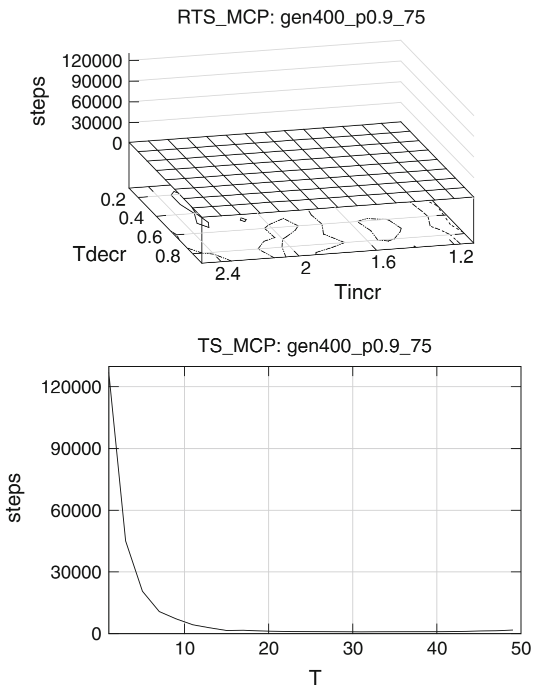


Fig. 8 Mean number of steps to reach the solution quality bound by RTS_MCP (*top*) and TS_MCP (*bottom*) on instance gen400_p0.9_75

TS_MCP is more sensitive with respect to parameter settings than RTS_MCP. On instance DSJC1000.5, the ratio of RTS_MCP meta-parameter settings is always higher than the ratio of TS_MCP parameter settings: whatever the number of steps, the empirical frequency of having meta-parameter configurations of RTS_MCP that reach the solution quality bound is higher than the one of TS_MCP. On instance gen400_p0.9_75, the frequency of parameter settings of TS_MCP is slightly higher than the one corresponding to RTS_MCP up to approximately 2,500 steps to reach the solution quality bound. The two algorithms need this number of steps to reach the solution quality bound in about 60 % of the observations. In the remaining 40 %, the frequency with which RTS_MCP needs the given number of steps to reach the solution quality bound is higher than the one of TS_MCP parameters. The difference between the two algorithms is quite large up to about 160,000 steps, when both algorithms manage to reach the solution quality bound with a frequency of 1.

Table 6 reports the mean number of steps to reach the solution quality 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 Figs. 6, 7 and 8, and we re-ran the experiments using these settings. When the appropriate settings are selected, TS_MCP outperforms

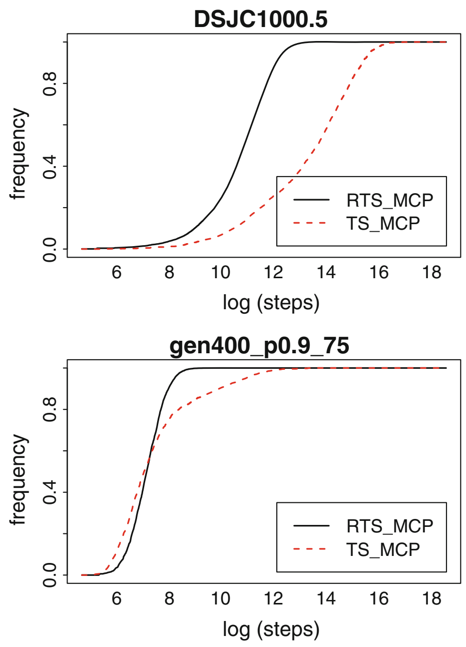


Fig. 9 Cumulative cost distribution of parameter and meta-parameter configurations for the MCP: all settings tested for RTS_MCP and TS_MCP

Table 6 Mean number of steps to reach the solution quality bound for the best instance-wise settings of RTS_MCP and TS_MCP

Instance	RTS_MCP		TS_MCP	
All	**996,578**	(1.1, 0.8)	1,119,774	(9)
C125.9	**125**	(1.1, 0.1)	147	(17)
C2000.5	37,817	(1.1, 0.1)	**36,556**	(3)
C250.9	1,365	(1.1, 0.8)	**1,115**	(17)
C500.9	78,059	(1.1, 0.7)	**41,951**	(13)
DSJC1000.5	41,700	(1.3, 0.3)	**31,949**	(3)
DSJC500.5	2,007	(1.1, 0.7)	**1,489**	(5)
MANN_a27	**104,910**	(1.9, 0.7)	107,614	(47)
brock200_2	101,054	(2.1, 0.4)	**76,410**	(11)
brock200_4	348,108	(1.9, 0.9)	**173,025**	(13)
gen200_p0.9_44	2,109	(1.1, 0.7)	**1,693**	(21)
gen200_p0.9_55	681	(2.2, 0.7)	**422**	(31)
gen400_p0.9_55	35,218	(1.4, 0.5)	**27,479**	(19)
gen400_p0.9_65	1,459	(1.9, 0.7)	**1,129**	(25)
gen400_p0.9_75	1,297	(2.3, 0.5)	**901**	(31)
p_hat1500-1	178,264	(1.2, 0.4)	**142,196**	(5)
p_hat1500-2	**844**	(1.4, 0.5)	848	(21)
p_hat1500-3	1,672	(1.1, 0.7)	**1,133**	(27)
p_hat300-1	135	(1.1, 0.5)	**125**	(3)
p_hat300-2	46	(2.3, 0.1)	**35**	(5)
p_hat300-3	787	(1.1, 0.6)	**577**	(15)
p_hat700-1	**1,359**	(1.1, 0.6)	1,556	(3)
p_hat700-2	137	(1.1, 0.1)	**107**	(11)
p_hat700-3	**319**	(1.2, 0.3)	402	(9)
hamming10-4	968	(1.4, 0.6)	**799**	(11)
hamming8-4	16	(1.1, 0.1)	16	(1)
keller4	164	(1.3, 0.1)	**36**	(11)
keller5	3,116	(1.1, 0.6)	**3,113**	(13)

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 hamming 8-4. We report in bold font the lowest mean number of steps for each instance and for the set of all instances. After 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

RTS_MCP in 21 of 27 instances. The mean relative increase of the number of steps to reach the solution quality 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 for these instances hardly measurable. In instance keller4, for example, the increase of the mean number of steps to reach the solution quality 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 (1) the best parameter and meta-parameter settings are quite different across instances, and (2) 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 (Battiti and Mascia 2010). In this case, the median difference in the number of steps to reach the solution quality bound when using the default and the best meta-parameter setting ($Tincr = 1.1$ and $Tdecr = 0.8$) is very small: according to the Wilcoxon rank-sum test with confidence level 0.95, the confidence interval is (−4,943, 2,150).

4.2 ANOVA analysis on the MCP

We performed an ANOVA analysis for RTS_MCP and TS-MCP on instance DSJC1000.5 and instance gen40_p0.9_75. As we did for the QAP, we applied an ad-hoc transformation to each set of results, and we report here the main effects after the application of the reverse transformation. Figures 10 and 11 depict the mean of the number of steps to reach the solution quality bound, 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.

Figure 10 shows that different meta-parameter settings have no remarkable impact on the performance (remark the extremely small scale used in the y-axis of the plots). For

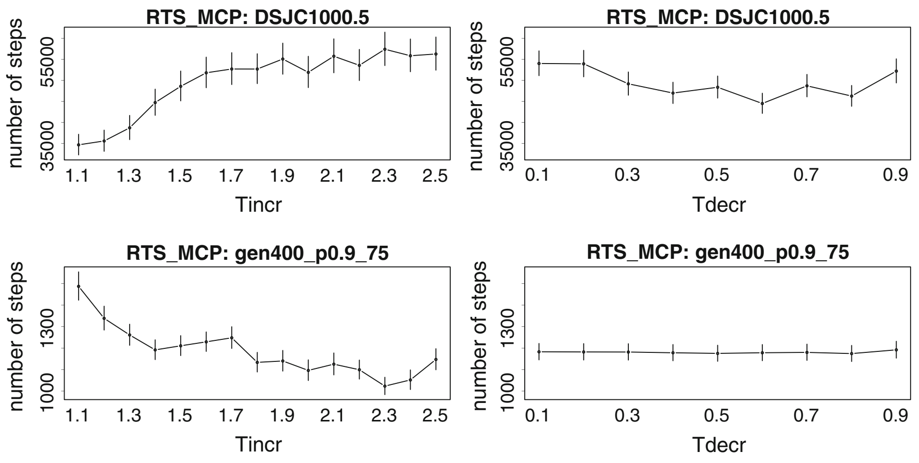


Fig. 10 Main effect in RTS_MCP. Instances DSJC1000.5 (*top*) and gen400_p0.9_75 (*bottom*). Note that all plots use different scale on the y-axis from the ones in Fig. 11, due to the extremely different variability of the results of RTS_MCP and TS_MCP

TS_MCP (Fig. 11), the appropriate settings are quite different on the two instances, as reported in Table 6. For TS_MCP, being T the only parameter considered, there are no interactions to account for, and the indications of the ANOVA results correspond to those of the analysis reported in Sect. 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 mean number of steps to reach the solution quality bound of a factor of 0.95 for instance DSJC1000.5 and of 130.12 for instance gen400_p0.9_75. Selecting an inappropriate meta-parameter setting for RTS_MCP may imply the increase of the mean number of steps to reach the solution quality bound of a factor of 0.66 for instance DSJC1000.5 and 0.45 for instance gen400_p0.9_75. Some interactions exist between RTS_MCP meta-parameters (Pellegrini et al. 2011), 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 (Pellegrini et al. 2011).

This study is motivated by the observation that RTS has been adopted in a number of researches. The usage of RTS is mostly due to the hope that RTS is little sensitive to its meta-parameters: when using RTS rather than TS, one implicitly exploits the intuition that, while for TS the parameter settings must be chosen specifically for each class of instances, this is not really necessary for RTS. However, this intuition has never been supported by strong experimental evidence.

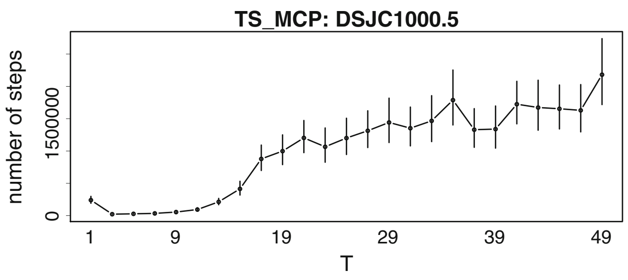
In our experimental analysis, we obtain strong results supporting this intuition. In particular, the results of the analyses on the two problems lead to the same conclusion:

1. RTS is little sensitive to its meta-parameter setting;
2. RTS is much less sensitive than TS to the setting of its parameters.

Moreover, there are no significant interactions between meta-parameters 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.

As for the best meta-parameter settings for the QAP and the MCP, our results underline that the ones given in the literature are not necessarily appropriate in different experimental setups. In particular, in the analysis for the QAP, we get to conclusions that are rather different from those drawn by Battiti and Tecchiolli (1994): 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



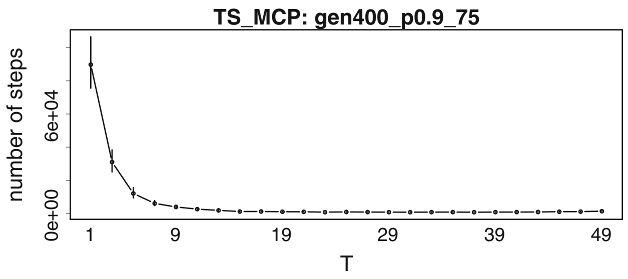


Fig. 11 Main effect in TS_MCP. Instances DSJC1000.5 (*top*) and gen400_p0.9_75 (*bottom*). Note that all plots use different scale on the y-axis from the ones in Fig. 10, due to the extremely different variability of the results of RTS_MCP and TS_MCP

$Tdecr = 0.5$ 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 (2010) ($Tincr = 1.1$ and $Tdecr = 0.9$) turned out to be very good on the set of all instances.

In summary, the paper shows that RTS is a very good option, 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 appropriate parameter settings for TS. In such a situation, RTS can be used even without tuning its 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. However, our results also show that an alternative to using RTS is to use parameter selection strategies where, depending on instance characteristics, instance-specific parameter values are chosen. This is an option we will explore further in follow-up research.

Acknowledgments 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. This research and its results have also received funding from the COMEX project within the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office. Mauro Birattari, Thomas Stützle and Franco Mascia acknowledge support from the Belgian F.R.S.-FNRS, of which they are Research Associates (MB, TS), or postdoctoral researchers (FM). The work of Paola Pellegrini has been partially funded by a Bourse d'excellence Wallonie-Bruxelles International.

References

Arntzen H, Hvattum LM, Lokketangen A (2006) Adaptive memory search for multidemand multidimensional knapsack problems. Comput Oper Res 33:2508–2525

Battiti R (1996) Reactive-search: toward self-tuning heuristics. In: Rayward-Smith VJ (ed) Modern heuristic search methods. Wiley, Chichester, pp 61–83

Battiti R, Bertossi A (2010) Greedy, prohibition, and reactive heuristics for graph-partitioning. IEEE Trans Comput 48(4):361–385

Battiti R, Brunato M (2005) Reactive search: machine learning for memory-based heuristics. In: Gonzalez TF (ed) Approximation algorithms and metaheuristics. Taylor & Francis Books (CRC Press), Washington, pp 21-1–21-17

Battiti R, Brunato M, Mascia F (2008) Reactive search and intelligent optimization. Operations research/computer science interfaces, vol 45. Springer, Berlin

Battiti R, Mascia F (2010) Reactive and dynamic local search for max-clique: engineering effective building blocks. Comput Oper Res 37(3):534–542

Battiti R, Protasi M (1999) Reactive local search techniques for the maximum k-conjunctive constraint satisfaction problem (MAX-k-CCSP). Discrete Appl Math 96–97:3–27

Battiti R, Tecchiolli G (1994) The reactive tabu search. ORSA J Comput 6(2):126–140

Birattari M (2004) 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

Birattari M, Stützle T, Paquete L, Varrentrapp K et al (2002) A racing algorithm for configuring metaheuristics. In: Langdon W (ed) Proceedings of the genetic and evolutionary computation conference 2002 (GECCO'02). Morgan Kaufmann, San Francisco, pp 11–18

Chiang W, Russell R (1997) A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS J Comput 9(4):417–930

DaCosta L, Fialho Á, Schoenauer M, Sebag M (2008) Adaptive operator selection with dynamic multi-armed bandits. In: Ryan C, Keijzer M (eds) Proceedings of genetic and evolutionary computation conference 2008 (GECCO'08). ACM, Atlanta, pp 913–920

Datta T, Srinidhi N, Chockalingam A, Rajan B (2010) Random-restart reactive tabu search algorithm for detection in large-MIMO systems. IEEE Commun Lett 14(12):1107–1109

DIMACS Center (2011) DIMACS implementation challenges. http://dimacs.rutgers.edu/Challenges/

Eiben AE, Michalewicz Z, Schoenauer M, Smith JE et al (2007) Parameter control in evolutionary algorithms. In: Lobo F (ed) Parameter setting in evolutionary algorithms. Springer, Berlin, pp 19–46

Fialho Á, DaCosta L, Schoenauer M, Sebag M (2010) Analyzing bandit-based adaptive operator selection mechanisms. Ann Math Artif Intell 60(1–2):25–64

Fink A, Voß S (2003) Solving the continuous flow-shop scheduling problem by metaheuristics. Eur J Oper Res 151(2):400–414

Hoos HH (2012) Automated algorithm configuration and parameter tuning. In: Hammadi Y, Monfroy E, Saubion F (eds) Autonomous search. Springer, Berlin, pp 37–71

Hussin M, Stützle T (2010) Tabu search vs. simulated annealing for solving large quadratic assignment instances. Tech. Rep. 2010–20, IRIDIA, Université Libre de Bruxelles, Brussels

Johnson DS, Trick MA (eds) (1993) Cliques, coloring and satisfiability: second DIMACS implementation challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 28. American Mathematical Society, Providence

Mascia F, Pellegrini P, Stützle T, Birattari M (2013) An analysis of parameter adaptation in reactive tabu search. Int Trans Oper Res (to appear)

Maturana J, Fialho Á, Saubion F, Schoenauer M, Lardeux F, Sebag M (2012) Adaptive operator selection and management in evolutionary algorithms. In: Hammadi Y, Monfroy E, Saubion F (eds) Autonomous search. Springer, Berlin, pp 161–189

Nanry W, Barnes J (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. Transp Res Part B Methodol 34(2):107–121

Osman I, Wassan N (2002) A reactive tabu search meta-heuristic for the vehicle routing problem with back-hauls. J Schedul 5(4):263–285

Pellegrini P, Mascia F, Stützle T, Birattari M (2011) Companion of: on the sensitivity of reactive tabu search to its meta-parameters. http://iridia.ulb.ac.be/supp/IridiaSupp2011-026/, IRIDIA Supplementary page

Pellegrini P, Stützle T, Birattari M (2012) A critical analysis of parameter adaptation in ant colony optimization. Swarm Intell 6(1):23–48

Russell R, Chiang W, Zepeda D (2008) Integrating multi-product production and distribution in newspaper logistics. Comput Oper Res 35(5):1576–1588

Stützle T (2006) Iterated local search for the quadratic assignment problem. Eur J Oper Res 174(3):1519–1539

Stützle T, López-Ibáñez M, Pellegrini P, Maur M, Montes de Oca MA, Birattari M, Dorigo M (2012) Parameter adaptation in ant colony optimization. In: Hammadi Y, Monfroy E, Saubion F (eds) Autonomous search. Springer, Berlin, pp 191–215