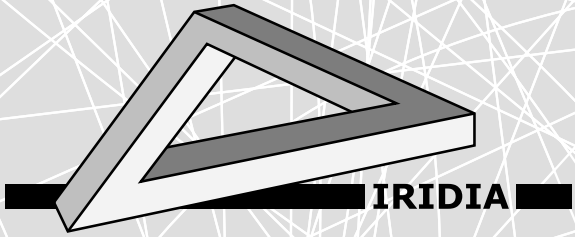


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Stochastic Local Search Algorithms for Graph Set T-Colouring and Frequency Assignment

Marco Chiarandini and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2006-019

July 2006

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/2006-019

Stochastic Local Search Algorithms for Graph Set T-Colouring and Frequency Assignment

Marco Chiarandini Thomas Stützle

Abstract

The graph set T-colouring problem generalizes the classical graph colouring problem; it is useful to model, for example, the assignment of frequencies in mobile networks. The graph set T-colouring problem asks for the assignment of sets of nonnegative integers to the vertices of a graph so that constraints on the separation of any two numbers assigned to a single vertex or to adjacent vertices are satisfied and some objective function is optimised. Among the objective functions of interest are the minimisation of the difference between the largest and the smallest integers used (the span), the minimisation of the number of integers used (the order), or the minimisation of the number of violated constraints.

The large size of the instances arising in practical applications makes the use of heuristic algorithms necessary. In this article, we propose new algorithms for solving this problem and we provide a comprehensive experimental study of previously known and new algorithms. This analysis is carried out using the objective of minimising the span. The empirical study comprises simple construction heuristics as well as elaborated stochastic local search algorithms. We show that the best construction heuristic is a generalisation of DSATUR that works on the transformation of the graph set T-colouring problem to a T-colouring problem; the best performing stochastic local search algorithms are Squeaky Wheel and a new combination of Tabu Search with an exact algorithm for the colour reassignment to single vertices.

1 Introduction

The *Graph Set T-Colouring Problem* (GSTCP) is a generalisation of the graph (vertex) colouring problem (GCP) and it arises in a number of applications, the most important being the assignment of frequencies to radio transmitters. In the GSTCP, one is given a graph, the number of colours, which are represented as integers, that are required at each vertex and separation constraints for each vertex and for each pair of vertices connected by an edge. The separation constraints represent the disallowed differences between the integers assigned to each vertex. The decision version of the GSTCP asks for an assignment of sets of nonnegative integers (colours) to the vertices of the graph such that all the constraints are satisfied. Such an assignment is also called a *proper* set

T-colouring. There are various optimisation versions of the GSTCP, which differ in the objective function to be minimised (Hale, 1980; Tesman, 1990). One version is to determine the *T-order* (or T-chromatic number), that is, the minimal number of integers, such that a proper set T-colouring exists. A second optimisation version requires to determine the *T-span*, that is, the minimal difference between the largest and the smallest integers such that a set T-colouring exists. If the number of available colours is limited the objective can also be to minimise the number, or a weighted number, of constraint violations.

The GSTCP arises in a number of real-world applications. The most important example is the assignment of frequencies to radio transmitters. In this case, vertices represent transmitters and colours (integers) the frequencies to be assigned to the transmitters subject to certain interference constraints, the T-constraints (Hale, 1980). These T-constraints are modelled by the vertex and inter-vertex separation constraints. Other applications arise in traffic phasing and fleet maintenance (Roberts, 1991) or in the task assignment problem, where a large task is divided into incompatible subtasks (for example, due to resource conflicts) and the problem is to assign a set of time periods to each subtask so that incompatible subtasks are in different time periods (Tesman, 1990).

In the frequency assignment application there are two possible scenarios of interest: the design scenario in which it has to be decided which frequency band to acquire in order to satisfy a forecast network utilisation; and the planning scenario where it must be decided which frequencies to assign to each cell given that the frequency band available for the provider is fixed and that the service must be maximised.[1] The first scenario corresponds to minimise the span while the second scenario to minimise the number of violated constraints. It may be the case then that penalties are associated to disallowed separations and acceptable interferences are weighted differently from forbidden interferences. This gives rise to the *minimal interference problem*, where the total penalty has to be minimised (Aardal et al., 2001; Eisenblätter et al., 2002). Here, also due to its practical relevance, we focus on the *minimal span GSTCP* on general and large graphs. Moreover, algorithms for this problem are easily adaptable also to the *minimal interference* case.

The GSTCP has received significant attention in theoretical and algorithmic studies. The main theoretical contributions concern properties of the generalised colouring of graphs such as computational complexity, lower bounds, and approximation schemes. From the original papers of Roberts (1991) and Tesman (1990, 1993), the attention has moved to specific graph typologies and specific problem formulations. Giaro et al. (2003a) and Giaro et al. (2003b) give complexity for special cases of the GSTCP. Lower bounds have been investigated in frequency assignment (see Allen et al., 1999; Janssen and Wentzell,

[1] Both cases of frequency assignment give raise to what is called, more precisely, the Fixed Channel Assignment Problem, whose name is used to emphasise that the model is static, *i.e.*, the set of connections remains stable over time.

2000; Smith et al., 2000; Montemanni, 2001), while few approximation results are available (Simon, 1989 and Janssen and Narayanan, 2001).[2]

Much of the algorithmic work on the GSTCP has been inspired from research on frequency assignment and, more recently, by the *Computational Challenge on Graph Colouring and its Generalisations (COLOR02/03/04)*,[3] which was organised by Johnson, Mehrotra and Trick.[4] Typically, these algorithmic approaches tackle large instances, as such predominantly arise in the frequency assignment application. Since exact algorithms have often proved to be inefficient for large instances (see the survey by Eisenblätter et al., 2002; Aardal et al., 2001 for a review on exact solution algorithms and their limits), most work has focused on stochastic local search (SLS) algorithms.

Costa (1993) is the first to address the GSTCP by means of heuristics and proposes a generalisation of the DSATUR heuristic for the classical graph colouring. Dorne and Hao (1998) apply Tabu Search to very large, randomly generated GSTCP instances. In the context of the COLOR02/03/04 DIMACS challenge, Phan and Skiena (2002) devised a simulated annealing algorithm using their general-purpose platform *Discropt*. Prestwich (2003) presented a randomised backtracking algorithm while Lim et al. (2003, 2005) designed a Squeaky Wheel algorithm. This latter approach also gives the best results for the benchmark instances that were proposed in the COLOR02/03/04 DIMACS challenge. However, from the published results it is unclear how this algorithm compares to the earlier proposed Tabu Search algorithm of Dorne and Hao (1998).

In this article, we give two main contributions. The first is the development of new algorithms for the GSTCP. The new algorithms include variants of construction heuristics and also the development of new SLS algorithms for the GSTCP. The most important is the extension of a Tabu Search algorithm by a colour reassignment procedure that at each vertex tries to replace the current set of colours by a different one, such that all constraints in which the concerned vertex is involved are satisfied. Additionally, we present also new adaptations of high-performing SLS algorithms for the GCP to the GSTCP. A second major contribution is an extensive experimental study of construction heuristics and more performing SLS algorithms for the GSTCP. This study comprises SLS algorithms that have previously been proposed for this problem and that have been re-implemented for this study and the newly developed SLS algorithms. We have decided to use re-implementations of the available algorithms to increase the fairness of the experimental comparison, since all the algorithms use the same framework, data structures, platform and implementer's ability. By doing so, we also can enlarge the comparison to more

[2]Theoretically driven research on the GSTCP was also the target of the *DIMACS/DIMATIA/Renyi Working Group on Graph Colourings and their Generalizations*; see `<http://dimacs.rutgers.edu/Workshops/GraphColor/main.html>` (June 2006) for more details.

[3]See `<http://mat.gsia.cmu.edu/COLORING04/>` (June 2006).

[4]Note that in this challenge, the separation distance GSTCP is called bandwidth multicolouring problem.

test instances, since for several of the previously known algorithms, results are available for only a small instance set. Our benchmark collection comprises several sets of randomly generated instances including random graphs and geometric graphs as well as instances derived from the frequency assignment literature (Anderson, 1973). The experimental study makes use of statistical methodologies sound such as a rigorous experimental design and a careful examination of the statistical significance of the observed differences. This experimental study allows us (i) to examine which are the key algorithmic features that are responsible for the good performance of construction heuristics, (ii) to give a more detailed picture of the current state-of-the-art in GSTCP solving and (iii) to assess the impact of the new SLS algorithms.

The paper is organised as follows. Section 2 introduces the notation and provides some basic results. Section 3 defines the benchmark instances. Section 4 describes construction heuristics which are empirically assessed in Section 5. Section 6 is dedicated to the description of the SLS algorithms. Section 7 reports the experimental analysis on SLS algorithms. Finally, Section 8 resumes the results of this work.

2 Definitions, Notation and Basic Results

The GSTCP is defined by

(i) an undirected graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges, with uv representing an edge in E;

(ii) a set of integers (called colours) Γ;

(iii) a number $r(v)$ of required colours for each vertex $v \in V$; and

(iv) a collection of sets $\mathcal{T}$ (called T-set), such that there is a set T_{uv} for each edge $uv \in E$ and a set T_u for each vertex $u \in V$. The T-sets represent the disallowed separations of colours expressed by nonnegative integers between vertices and within vertices, respectively.

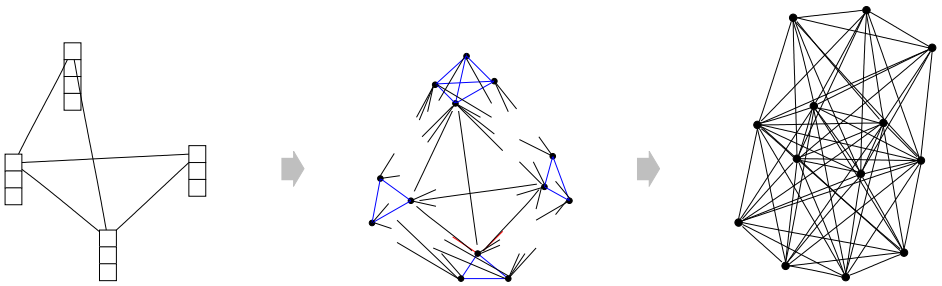
The task in the decision version of the GSTCP is to find, for a given number of $k = |\Gamma|$ colours, a mapping $\varphi : V \mapsto \mathcal{P}(\Gamma)$ such that the three following groups of constraints

$$|\varphi(v)| = r(v) \qquad \forall\, v \in V \tag{1}$$

$$|x - y| \notin T_u \qquad \forall u \in V,\ \forall x, y \in \varphi(u),\ x \neq y \tag{2}$$

$$|x - y| \notin T_{uv} \qquad \forall uv \in E,\ \forall x \in \varphi(v),\ \forall y \in \varphi(u) \tag{3}$$

are satisfied. The three groups of constraints to be satisfied are called *requirement constraints*, *vertex constraints*, and *edge constraints*, respectively. We call a mapping φ a *proper set T-colouring*, if all constraints are satisfied and *improper*, otherwise.



(a) Original Graph *(b)* Split Graph

Figure 1: The generation of the split graph.

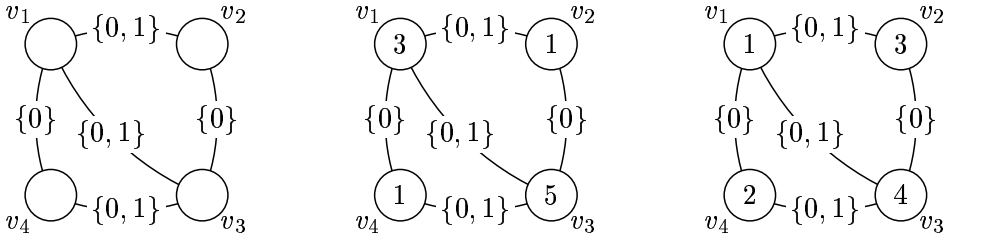
The special case of the GSTCP, where we have $r(v) = 1$, $\forall v \in V$ is called the *Graph T-colouring Problem* (GTCP). In the GTCP, only the edge constraints are present. Both, the GSTCP and the GTCP are obviously in $\mathcal{NP}$ and, hence, they are also $\mathcal{NP}$-complete, since they are generalisations of the $\mathcal{NP}$-complete *k-colouring problem* (Garey and Johnson, 1979).

A GSTCP instance for a given graph $G = (V, E)$, T-set $\mathcal{T}$ and vertex requirements $r(v), \forall v \in V$, is equivalent to a GTCP instance on a graph $G^S = (V^S, E^S)$. The graph G^S is obtained from G by creating a vertex u for each requirement of a vertex $v \in V(G)$ such that $|V^S| = \sum_{v \in V} r(v)$. The vertices $u \in V^S$ derived from a vertex $v \in V$ form a clique of order $r(v)$ in which each edge receives the set of constraints T_v. Every such vertex is then connected with each vertex of the clique induced by another vertex $w \in V$ if $vw \in E$. The colour separation associated with these edges is T_{vw}. The graph G^S is called *split graph* and there is a bijective correspondence between solutions on G and on G^S. (The generation of the split graph is illustrated in Figure 1.)

Two properties of a proper (set) T-colouring φ are of interest for its quality: the *order*, *i.e.*, the number of different colours effectively used by φ and the *span*, *i.e.*, $\max_{u,v \in V}\{|x - y| : x \in \varphi(v), y \in \varphi(u)\}$, the maximal difference between the colours used. The minimum order for which a proper k-colouring exists is called the *(set) T-chromatic number* $\chi_T(G)$; [5] the minimum span is the *(set) T-span* $sp_T(G)$. If $\min_{u \in V}\{x : x \in \varphi(u)\}$ is fixed to 1, the span corresponds to $k - 1$, where k is the maximal number of colours. Minimising the order, the span or both is the objective in the optimisation versions of the GSTCP. For all graphs G and T-sets $\mathcal{T}$ it holds that $\chi_T(G) \leq sp_T(G)$ (Hale, 1980). Note that $\chi_T(G) = \chi_T(G^S)$ and $sp_T(G) = sp_T(G^S)$ and, hence, the solution to a GSTCP can be found by solving the GTCP on the split graph.

For the GTCP, Cozzens and Roberts (1982) proved that the T-chromatic number of a graph G, $\chi_T(G)$, is equal to the chromatic number $\chi(G)$. However, already Hale (1980) pointed out that minimising the span of a T-colouring is different from minimising the order, or the span of a colouring without the T-

[5] Note that the set T-chromatic number can be smaller than k.



(a) The original instance.

(b) A solution with 3 colours and span 4.

(c) A solution with 4 colours and span 3.

Figure 2: A GTCP instance with $\chi_T(G) = 3$ and $sp_T(G) = 3$. Every solution that uses 3 colours has span at least 4, whereas every solution with span 3 uses 4 colours.

constraints. There are cases, indeed, where no optimal span T-colouring gives an optimal order and vice-versa. For example, if the graph G and the T-set $\mathcal{T}$ are those of Figure 2, then $\chi_T(G) = 3$ and $sp_T(G) = 3$ but a T-colouring with order 3 has span at least 4, whereas a T-colouring with span 3 has order 4.

In the frequency assignment application, more relevant is the case in which all $T_u, T_{uv} \in \mathcal{T}$ consist of consecutive integers $\{0, 1, \ldots, t_{uv} - 1\}$ for all $uv \in E$ and $\{0, 1, \ldots, t_u - 1\}$ for all $u \in V$. This particular version of the problem is known as *separation distance GSTCP* problem and the values t_u, t_{uv} are called *colour distances* (Eisenblätter et al., 2002). The edge and vertex constraints then have the form $|c_1 - c_2| \geq t$, where c_1 and c_2 are two assigned colours.

Finally, we need some few additional notation in what follows. We denote by $A_G(v)$ the set of vertices adjacent to a vertex $v \in V$, that is, the set of all vertices $u \in V$ with $uv \in E$. The degree $d(v) = |A_G(v)|$ of a vertex v is the number of vertices adjacent to v.

3 Benchmark instances

To evaluate the performance of the algorithms that are presented in the following sections, we define a benchmark collection that comprises three sets of random instances that differ in structural aspects of the graphs and the constraints, and a set of instances derived from the frequency assignment application. The first set comprises randomly generated instances that were proposed by M. Trick in the context of the "DIMACS Computational Challenge on graph colouring and its generalisations".[6] The second set was introduced by Dorne and Hao (1998) for testing their tabu search algorithm. Since some of these instances entail very high computation times, we generated an additional, third set of instances with similar characteristics as those of Dorne and Hao (1998) but smaller in size. We use this third instance class for running extensive computational tests. Finally, we added a fourth set of instances

[6] M. Trick. "Computational Series: Graph Coloring and its Generalizations.", `<http://mat.gsia.cmu.edu/COLOR04/>`. (last visited June 2006.)

that are based on the well-known Philadelphia instances from the frequency assignment literature (Anderson, 1973).

Random geometric instances (DIMACS). In these instances, first a set of points with coordinates that are uniformly distributed at random in a $[10,000 \times 10,000]$ square is generated. To each point a vertex is associated and the vertices are connected by an edge if the corresponding points are close enough. The separation distances associated to edges are inversely proportional to the distances between the points. Vertex requirements are chosen uniformly at random from the set $\{1, \ldots, r\}$ and vertex separation distances are fixed to 10. The size of these instances ranges from 20 to 120 vertices. We denote this set by `GEOM`, its class of sparse instances as `GEOM`n and the classes of denser instances as `GEOM`n`a` and `GEOM`n`b`. The instances `GEOM`n`b` have higher requirements per node than `GEOM`n`a`. Statistics of this instance set are given in Table 1a.

Random uniform instances (Dorne and Hao, 1998). These instances are based on uniform, random graphs, which are typically identified as G_{np}, where n is the number of vertices and each of the $\binom{n}{2}$ possible edge is included in the graph with a probability p. Vertex requirements, vertex-distances and edge-distances are generated according to a uniform distribution in the interval $\{1, \ldots, t\}$. Statistics on these instances are in Table 1b. We call this set of instances `HD-RU` and maintain the original nomenclature for the single instances: `essai`$.n.R.p$ where $R = \sum r(v) = |G^S|$.

Random uniform instances (new). These instances are also based on G_{np} graphs and we generated them using the generator of Culberson et al. (1995) that was modified to produce set T-colouring instances. The graphs and the vertex-distance, and edge-distance constraints are constructed as described for the set `HD-RU`. Vertex requirements are instead chosen uniformly at random from the interval $\{1, \ldots, r\}$ with possibly $r \neq t$. The values assigned to the parameters and the number of instances are reported in Table 1c. We denote this set of instances `NRU` and each single class as `TG-`n-p-$r.t$.

Philadelphia instances (Anderson, 1973). They are characterised by 21 hexagons corresponding to the cells of a cellular phone network around Philadelphia. For each cell, a demand $r(v)$ is given. If the mutual distance between the centre of two cells is less than d (normalised by the radius of the cells), it is not allowed to assign the same frequency to both cells. This case is generalised by replacing the separation distance d by a series of non-increasing values $d^0, \ldots, d^k$. For example, the values $2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0$ for $d^0, \ldots, d^5$ imply that frequencies assigned to the same cell should be separated by at least 4 other frequencies, whereas frequencies assigned to adjacent sites should be at a distance of at least 2, and frequencies assigned to a second and third 'ring' of cells should still

$\|V\|$	$\bar{\rho}$	r	t_u	$\overline{t_{uv}}$	# instances
$20, 30, \ldots, 120$	0.1	3	10	4.5	–
		10	10	4.5	11 (GEOMn)
	0.2	3	10	4.5	11 (GEOMna)
		10	10	4.5	11 (GEOMnb)

(a) Random geometric instances

$\|V\|$	p	$r = t_u = t_{uv}$	# instances
$30, 100, 300, 500, 1000$	0.1	5	5
	0.5	5	5
	0.9	5	5

(b) Random uniform instances (Dorne and Hao, 1998)

$\|V\|$	p	r	t	# instances
60	0.1	5	5	10
			10	10
		10	5	10
	0.5	5	5	10
			10	10
		10	5	10
	0.9	5	5	10
			10	10
		10	5	10

(c) New random uniform instances

Inst	$\bar{\rho}$	$[r^{min}; r^{max}]$	t_u	$[t_{uv}^{min}; t_{uv}^{max}]$
P1	0.73	[8;77]	5	[1; 2]
P2	0.49	[8; 77]	5	[1; 2]
P3	0.73	[5; 45]	5	[1; 2]
P4	0.49	[5; 45]	5	[1; 2]
P5	0.73	[20; 20]	5	[1; 2]
P6	0.49	[20; 20]	5	[1; 2]
P7	0.73	[16; 154]	5	[1; 2]
P8	0.73	[8; 77]	5	[1; 2]
P9	0.73	[32; 308]	5	[1; 2]

(d) Philadelphia instances

Table 1: Statistics on the benchmark instances. t_u and t_{uv} are used to indicate that the range of values differs among vertex and edge constraints. $\bar{\rho}$ gives the measured edge density of the graphs.

differ. For more on these instances we refer to the FAP web repository.[7] In conformity with the frequency assignment literature we denote these instances by P1-P9.

4 Construction Heuristics

Construction heuristics are the fastest methods to generate good quality solutions for optimisation problems. Their other main use is to provide good initial solutions to more performing SLS algorithms. For the GSTCP, various construction heuristics have been presented and the most comprehensive study so far is that of Hurley et al. (1997), who present different sequential heuristics. They conclude that it is difficult to predict which variant will perform well on a specific instance; however, their comparison is limited to small size instances of around 20 vertices and no statistical analysis is applied to support the results.

Here, we provide a comprehensive study of construction heuristics for the GSTCP that includes the sequential heuristics derived from Hurley et al.

[7] A. Eisenblätter and A. Koster. "FAP web – A website devoted to frequency assignment". September 2005. <http://fap.zib.de.> (January 2006).

Function T-greedy$(G^S, \pi, \mathcal{T})$;
$\Gamma = Z^+ \setminus 0$;
$\varphi(v_{\pi(1)}) = \{1\}, \varphi(v_{\pi(2)}) = \emptyset, \ldots, \varphi(v_{\pi(n)}) = \emptyset$;
for $i = 2, \ldots, |V^S|$ **do**
  $\varphi(v_{\pi(i)}) = \min\{l \in \Gamma : \forall j < i,\ |l - \varphi(v_{\pi(j)})| \geq t_{ij},\ v_{\pi(i)}v_{\pi(j)} \in E^S\}$;
Let $k = \max\{l : \varphi(v_i), v_i \in V^S\}$;
return k and the T-colouring $\varphi(v_1), \varphi(v_2), \ldots, \varphi(v_n)$;

Algorithm 1: Greedy construction heuristic for the GTCP. For short we write t_{ij} for $t_{v_{\pi(i)}, v_{\pi(j)}}$

(1997), heuristics derived from the GCP and the assignment heuristics proposed by Sivarajan et al. (1989). Some of the proposed heuristics are new, like the RLF derived from the GCP. Other heuristics, like our generalised DSATUR, constitute a strong enhancement with respect to previously published versions, which is obtained by the appropriate combination of different profitable choices.

4.1 Split Graph (T-Colouring) Approach

Sequential heuristics. Sequential heuristics iteratively assign colours to vertices until a complete colouring is reached. Each assignment decision consists of two steps: first, the vertex that is to be coloured next is chosen, and then this vertex is assigned a colour.

As for the GCP, there is a basic heuristic for assigning the colour, which is called *greedy algorithm*. For the GTCP it was first studied by Cozzens and Roberts (1982). This algorithm colours the vertices of the graph with the smallest feasible colour, proceeding in a given vertex order and it always produces a proper T-colouring for any graph and T-set. A pseudo-code of the greedy algorithm, denoted by T-greedy, is given in Algorithm 1. It takes as input a graph G and a permutation π of the vertices. An efficient implementation of the T-greedy algorithm maintains a set of forbidden colours for each vertex that is still to be coloured and updates it at each iteration. The complexity is $\mathcal{O}(k|V^S|)$, where, differently from the GCP, k is not bounded by the number of vertices.

The permutation π of the vertices can be determined in a static manner, that is, before starting the solution construction according to a prescribed strategy. Inspired by the vertex degree information in graph colouring, the sorting strategies we examined are: Random (RO algorithm), Largest First (LF algorithm), Smallest Last (SL algorithm). The LF algorithm orders vertices in non-increasing order of their degree. The SL algorithm arranges the vertices such that the vertex at position i, $v_{\pi(i)}$ has the smallest degree in the subgraph $G' \subset G$ induced by $V' = \{v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(i)}\}$. This sequence is determined starting at the final vertex and proceeding in reverse order. We

denote the three heuristics G-ROS, G-LFS, and G-SLS.

Various authors have used an *adjusted vertex degree* that uses the additional information of the separation distance constraints between adjacent vertices and that is defined as $d^T(v) = \sum_{u \in A_G(v)} t_{uv}$. We include therefore in our analysis for all the sequential heuristics two versions, either using the vertex degree or the adjusted vertex degree.

Generalised DSATUR heuristics. The DSATUR heuristics are also based on T-greedy but they order the vertices dynamically at solution construction on the basis of their *saturation degree*, *i.e.*, the number of different colours forbidden by the colours assigned to the adjacent vertices. For every vertex v that receives a colour c, the list of forbidden colours of vertices $u \in A_{G^S}(v)$ is updated by inserting the colours in the interval $(c - t_{uv}, c + t_{uv})$.

We tested two different sorting strategies: largest saturation first and smallest saturation first. In both cases, ties are broken by the largest vertex degree (ties occur, however, much less frequently than in DSATUR for GCP). Preliminary tests indicated that a smallest saturation first order is preferable. Note that this result is in contrast to what is used in DSATUR for GCP and to the implementations of generalised DSATUR by Costa (1993); Hurley et al. (1997); Borndörfer et al. (1998); Dorne and Hao (1998). Hurley et al. (1997), also suggest to colour first vertices in the largest clique but we could not detect any improvement by this procedure and hence we decided to omit it for the sake of a simpler heuristic. We denote our version of generalised DSATUR G-DSATUR.

Together with the generalised DSATUR, we study a modification of the T-greedy algorithm that selects, whenever more than one previously used colour is feasible, the colour that is feasible for the smallest number of uncoloured vertices. In this way, one seeks to first use the colours with less chances to be used later in the process. We denote this algorithm as T-greedy$^{\#}$. Its worst case complexity when applied to a static vertex order is $\mathcal{O}(k^2|V^S|^2)$. Empirical observations showed that even for high density graphs with $\rho(G) = 0.9$, where vertices are less likely to be colourable with previously used colours, this rule applies in about 50% of the decisions taken.

Finally, as for the sequential heuristics, we study the effect of the use of an adjusted vertex degree for breaking ties in G-DSATUR, which is also a new element of DSATUR-based construction heuristics for the GTCP.

Generalised RLF heuristic. For the vertex colouring problem, one of the best performing construction heuristics, the *Recursive Largest First* (RLF) heuristic, is based on a partitioning approach. It iteratively fills colour classes by colouring as many vertices as feasible with the same colour before using a new colour. When applied to the GTCP, the concept of independent set is still valid but vertices are also subject to distance constraints and hence their insertion in a set must also guarantee the separation constraints between the colours that will be assigned to such sets. We denote this adapted RLF

heuristic by G-RLF. For G-RLF, we also study the effect of an adjusted vertex degree.

4.2 Original Graph (Set T-Colouring) Approach

Sequential heuristics. The T-greedy algorithm can be adapted to work directly on the GSTCP by assigning to a vertex v at each construction step $r(v)$ colours. The colour assignment policy remains the same: priority is given to the smallest feasible colours. The order of the vertices to be coloured can be determined using the same rules (RO, LF, SL) as they were presented before. We implemented these heuristics using both, the vertex degree and the adjusted vertex degree.

Other assignment heuristics. The heuristics proposed by Sivarajan et al. (1989) look at the problem from an assignment perspective. These heuristics were not yet studied extensively on instances with more than 21 vertices and not in comparison with the other heuristics above. These heuristics work at the level of single vertex requirements. They first dispose in a certain order the requirements by combining a *vertex order* and a *requirement order* and then assign to the requirements in the given order a colour according to an *assignment policy*. Each heuristic results as one combination of the alternative choices of three factors.

Vertex order: For each vertex, the adjusted degree $d_A^T(v) = \sum_{u \in A_G(v)} r(v)t_{uv} - t_v$ is computed. This is a surrogate measure for the number of individual constraints acting on each requirement. On the basis of this information the order is formed according to these alternatives:

- smallest last (identifier C);
- largest first (identifier D).

Requirement order: Once vertices have been ordered, requirements can be arranged in a matrix $|V| \times \max_{v \in V} r(v)$. Each row of the matrix corresponds to a vertex while each column to a single colour requirement. The rows are arranged in the vertex order determined previously while the colour requirements are arranged such that all the columns of the matrix have about the same number of requirements. Requirements in the first row start at the first column. Requirements in the second row start at column $(r(v_{\pi(1)}) + 1)$ and fill the row cyclically. Similarly, requirements in the remaining rows start where requirements in the previous row end. Once requirements have been arranged in this way in the matrix, two orderings may be obtained:

- row-wise ordering (identifier R);
- column-wise ordering (identifier C).

	T-colouring approach	set T-colouring approach
Sequential	order={RO,LF,SL} adjust={yes,no}	order={RO,LF,SL} adjust={yes,no}
DSATUR	assign={T-greedy,T-greedy$^{\#}$} adjust={yes,no}	–
RLF	adjust={yes,no}	–
Other Assignment	–	order={C,D} requ.={R,C} assign={F,R}

Table 2: Implementation choices for the construction heuristics for GSTCP.

Assignment policy: Once the requirement order is fixed, two strategies can be adopted to assign the colour:

- Colour exhaustive: assign the smallest colour that does not introduce any violation to each requirement selected in the defined order (identifier F);
- Requirement exhaustive: for each colour in order from 1 to k scan all yet uncoloured requirements in the defined order and assign the current colour if feasible (identifier R).

We identify these assignment heuristics by the letter abbreviations as defined above. For example, CRR refers to a heuristic that uses smallest last vertex order, row-wise requirement order and an requirement exhaustive assignment policy.

Table 2 summarises again the factors that influence the construction heuristics and that we will study experimentally in the next section.

5 Empirical Assessment of Construction Heuristics

The goal of the experimental study of construction heuristics is to, firstly, identify the algorithmic features that are important for the heuristics of each class to reach good performance and to, secondly, identify the best performing construction heuristics. In fact, the attempt to distinguish and separate the effects of the single choices that compose the heuristics is a novelty of this analysis which is made possible by a careful organisation and design of experiments. Some of these choices are, for example, the use of the GSTCP formulation or the GTCP transformation or the use of an adjusted vertex degree.

Concerning performance, here we focus on the solution quality returned by the heuristics. For all heuristics, the computation times are very modest and are below 0.15 seconds on our computer, a 2 GHz AMD Athlon MP 2400+ processor with 256 KB cache and 1GB RAM[8]

[8]The output of the benchmark code available from the DIMACS "Computational Challenge" web site is: `DFMAX(r500.5.b) 8.64 (user)`. The machine runs under Debian Linux, the code was implemented in `C++` and compiled with `gcc` and flags -O3.

Since randomised decisions may occur due to random tie breaking, we collected for each of the heuristics 10 runs per instance (seven instances per instance class) and as the response variable we measured for each trial the span returned. In a first step, we examined the influence of the various design factors for the heuristics on performance. This experimental design is described by Table 2. It is rather complex and it requires to split the analysis in four separate designs, one for each type of approach, that is, one for each row of the table, in order to distinguish main and interaction effects of the implementation factors. We treat the single instances as blocking factor and use the ranking of returned span on each instance to normalise the results between instances. In the experiments we used the instance classes {GEOM, GEOMa, GEOMb, TG-120-0.1-5.5, TG-120-0.5-5.5, TG-120-0.9-5.5} taken from the GEOM and NRU benchmark sets. For each class of construction heuristics, we also study the interaction algorithm–instance class, that is, we examine whether and how the relative performance of the algorithms depends on the various instance classes. The study of this interaction is reasonable, since instance features can be recognised *a-priori* and a best algorithm for each instance class could be selected.

Sequential Heuristics. We considered four factors problem representation {GTCP, GSTCP}, adjusted degree {Y, N}, vertex order {RO, LF, SL} and instance class {GEOM, GEOMa, GEOMb, TG-120-0.1-5.5, TG-120-0.5-5.5, TG-120-0.9-5.5}. An analysis of variance (ANOVA) (Montgomery, 2005) on the results indicated that all factors and all interactions have significant effects; only the factor instance class does not have a significant main effect but this is due to the transformation of results in ranks.

To obtain insights into the importance of the various choices if best performance is desired, we use regression trees (Bartz-Beielstein and Markon, 2004; Breiman et al., 1984). Regression trees are decision trees on the factors in the analysis; binary splits occur if effects are statistically significant and the importance of the factor can be derived from the level where the split occurs in the tree. To obtain reasonably robust regression trees, we aggregated the analysis and considered two sets: one for the instances of the GEOM set and one for those of the NRU set; the regression trees are given in Figure 3. In the trees, each node reports the predicted average rank value for the selected configuration (i.e., the combination of factor levels that label the path from the root to that node). At each node, following the left path leads to the better performing combination; that is, the left-most path corresponds to the best choice. The only choice that seems to remain constantly significantly important is the use of an adjusted vertex degree. However, interestingly, a closer look revealed that on the classes GEOM and GEOMa even the random vertex order resulted the best heuristic. Moreover, computation times are too small to distinguish any important difference. A priori it is therefore not possible to discard any of these heuristics.

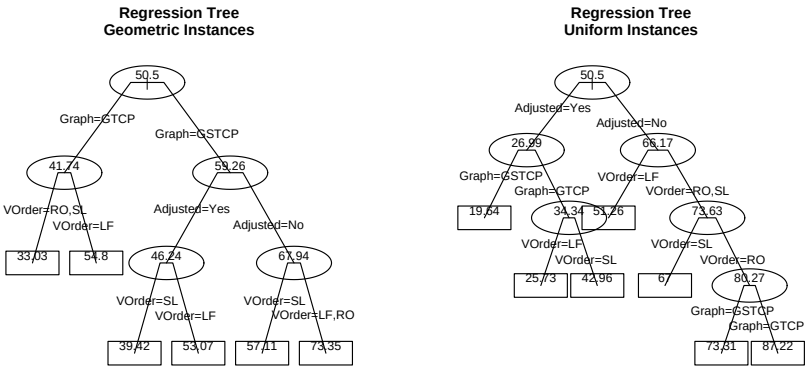


Figure 3: Regression tree analysis for sequential construction heuristics on the GSTCP problem. Numerical values in the nodes indicate the predicted average rank for the combination of factor levels that label the path from the root to that node.

DSATUR. We considered the factors greedy assignment {T-greedy, T-greedy$^{\#}$}, adjusted degree {Y, N} and instance class. An ANOVA analysis indicates as significant the main effect of the factor adjusted degree[9] and its interaction with the instance classes. A closer examination of the computational results showed that for all instance classes, the usage of the adjusted vertex degree results into significantly better performance. The only exception is the class TG-120-0.1-5.5, where adjusted and non-adjusted vertex degree result in similar performance; this latter observation also explains the significance of the interaction effect. Hence, we can conclude that G-DSATUR performs better with an adjusted vertex degree. No significant difference of the solution quality between using T-greedy or T-greedy$^{\#}$ was found. Since, T-greedy$^{\#}$ has a worse computational complexity and also empirically results in slightly larger computation times, we discarded its use.

RLF. The only factor to study for G-RLF is the use of an adjusted vertex degree. An aggregated analysis over all instance classes, using the Wilcoxon sum rank matched pairs test, indicates that the usage of the adjusted vertex degree results in significantly worse performance.

Other Assignments. The factors of interest here are vertex order {C, D}, requirement order {R, C}, assignment policy {F, R} and instance class. An ANOVA analysis indicates as significant all main effects but not the interactions. However, differences among the heuristics arise in the instance classes and the only choice that remains significantly the best across all instance classes is the use of a requirement exhaustive assignment policy in the heuristics. No perceivable differences can be detected in computation time and no

[9]The significance of the main effect of the adjusted vertex degree is also confirmed by the nonparametric Friedman rank sum test (Conover, 1999).

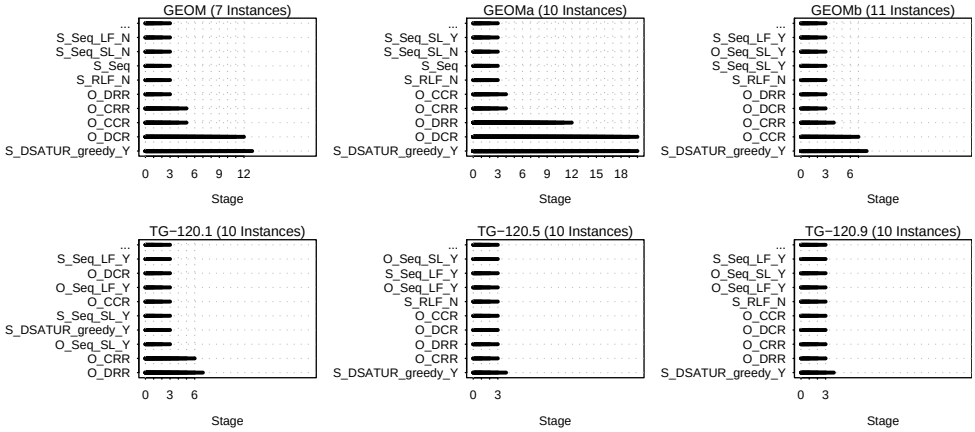


Figure 4: The outcome of the F-races for the construction heuristics on the 6 instance classes. A stage corresponds to the collection of one run of each heuristic on all the instances of a class. On the y-axis, algorithms are ordered according to average rank, the best being closer to the origin. The line indicates the life of the heuristic in the race. Only the best 9 algorithms out of 16 are depicted. In the labels the first letter refers to the split graph S or original graph O approach.

run took more than 50ms.

Selection of the overall best heuristic. The previous analysis indicated the best configuration for DSATUR, *i.e.*, T-greedy together with adjusted vertex degree, and RLF and allowed us to discard all the assignment heuristics that do not use a requirement exhaustive assignment policy. No general conclusions were instead possible for the sequential heuristics. In order to identify an overall best heuristic, we apply F-races, a racing algorithm for the tuning and selection for heuristics described in Birattari et al. (2002). Racing algorithms sequentially evaluate candidate algorithms and discard candidates as soon as statistically significant information against them arises. We run one race for each instance class {GEOM, GEOMa, GEOMb, TG-120-0.1-5.5, TG-120-0.5-5.5, TG-120-0.9-5.5}. Here, we use a Friedman test for replicated designs (Conover, 1999) to test the statistical significance of the differences among the algorithms but we do not discard candidates in the first three stages of the race. The F-race stops if only one winner remains or after a maximum of 20 stages. A total of 16 candidate construction heuristics took part in the race. The results are reported in Figure 4.

The overall best algorithm is G-DSATUR. Only on the instance class TG-120-0.1-5.5 it is not clearly the best heuristic while it dominates on all other classes. Its computation times on these instances are never above 70 msec. Overall, G-DSATUR seems a robust and efficient construction heuristic for GSTCP and we therefore also use it to generate the initial solutions for the SLS algorithms that are presented and studied in the next two sections.

6 Stochastic Local Search Algorithms

Perturbative local search algorithms iteratively move in the search space of complete candidate solutions $\mathcal{C}$, where the possible set of successor solutions is defined by a *neighbourhood structure* $\mathcal{N} : \mathcal{C} \to 2^{\mathcal{C}}$. Stochastic local search algorithms enhance basic perturbative local search procedures (Hoos and Stützle, 2004). They are known to be among the top performers for the GCP and previous experience suggests that this is also the case for the GSTCP. The new algorithms we developed and the previously known SLS algorithms for the GSTCP essentially follow three different schemes for tackling the problem. These schemes depend on the choices taken for the application of the underlying perturbative local search.

6.1 Schemes for applying SLS algorithms to the GSTCP

There are various possibilities of how to tackle the GSTCP by perturbative local searches. A first possibility is to consider the transformation of GSTCP instances into GTCP instances. The results on the construction heuristics have shown that this is an approach that deserves careful consideration. This choice has an impact in how vertex and edge constraints are treated. While in the GSTCP formulation it is easy to enforce that vertex constraints are always satisfied, in the GTCP formulation, edge and vertex constraints are put to a same level and both types of constraints may be violated during the search. (Note that the requirement constraints can easily be enforced in both representations.)

Another possible choice concerns the way how the search for an optimal span is organised. The first possibility is to tackle the GSTCP as a sequence of decision problems. In this case, for a given number of colours k a satisfying solution is searched and once it is found, k is reduced by one and so on. (There are other possibilities for defining the decision problems; however, this successive reduction of k in steps of one has shown to be successful for the GCP.) An alternative is to leave k free to decrease and increase during the search.

The combination of the different choices gives rise to several possible local search schemes that form the basis of the SLS algorithms we have implemented.

Scheme 1: split graph, k fixed. *Candidate solutions* are represented as complete assignments, *i.e.*, one colour is assigned to each vertex. The *evaluation function* f counts the number of violated vertex and edge constraints and, hence, the goal becomes to find a colour assignment with zero violations. As the *neighbourhood* one can use the well-known one-exchange neighbourhood, where for a given candidate solution s the neighbourhood comprises all those candidate solutions that differ in the colour assignment of exactly one vertex. Often, it is useful to restrict this neighbourhood such that only vertices that are involved in a constraint violation change their colour. We call this restricted neighbourhood N_E.

Scheme 2: original graph, k fixed. This scheme is similar to the previous one, but it works directly on the GSTCP formulation. Hence, in this scheme *candidate solutions* are now represented as sets of $r(v)$ colours for each vertex $v \in V$. In this representation, the vertex constraints, in addition to the requirement constraints, may be enforced to be satisfied and, hence, the effective search space searched may be strongly reduced. Hence, the evaluation function needs only to count the number of violated edge constraints. A perturbative local search for Scheme 2 can again be based on a one-exchange neighbourhood that is analogous to N_E but that restricts the set of new possible colours in the one-exchanges as to maintain the vertex constraints satisfied. We denote this neighbourhood by N'_E.

In addition, we define a new vertex colour reassignment neighbourhood. This neighbourhood consists of all possible reassignments of colours at a single vertex. Since this neighbourhood is of exponential size and to make the search more effective, we further restrict this neighbourhood to consider only reassignments that satisfy the requirement, vertex and edge constraints acting on the single vertex. This neighbourhood we denote as N_R. As such, N_R requires the exact solution of a sub-problem, *i.e.*, finding a feasible assignment of $r(v)$ colours among $\{1, \ldots, k\}$ to a vertex v under the condition that none of the other vertices changes its colour assignment. Clearly, such a reassignment of colours that satisfies all edge constraints at a vertex need not exist, which may lead to the case that $\mathcal{N}_R$ is empty.

Scheme 3: split graph, k variable. In this scheme, a first solution and an initial k_I is provided by a construction heuristic but the number of colours is then left free to vary at run time, bounded only by k_I. Here, *candidate solutions* can again be represented as complete colourings as in Scheme 1. The difference is that solutions can be proper and improper T-colourings. An *evaluation function* to guide the search toward proper colourings and toward colourings with small span was proposed by Hurley et al. (1997). It is defined as

$$f(s) = k_{max} + k_I \cdot \Big(\sum_{uv \in E} I_e(uv) + \sum_{v \in V} I_v(v) \Big) + (k_{max} - k_{min}) + \sum_{i=1}^{k_I} I_g(i) \quad (4)$$

where k_{max} is the maximal colour, $I_e(uv)$ and $I_v(v)$ are indicator functions that return one if the corresponding edge or vertex constraint is broken, $k_{max} - k_{min}$ is the span of the colouring, and $I_g(i)$ is an indicator function to determine whether any vertex has colour i; this last term computes the order. The edge and vertex conflicts are weighted by the largest number of colours, thus a solution which reduces the number of violations will always be preferred with respect to those that modify the other terms of the sum. The inclusion of the order in the sum contributes to break ties. The term k_{max} is the least important and contributes only to use the first colours, avoiding to move with the same span over and over through the interval $[1, k_I]$.

Finally, it is again straightforward to use the usual one-exchange neighbourhood as defined already for scheme 1.

6.2 Neighbourhood Examination

One-exchange neighbourhood. A crucial aspect of local search algorithms is the examination of the neighbourhood. A complete examination of N_E involves about $k \cdot |V^c|$ possible neighbours, where V^c is the set of vertices that are involved in a conflict. Additional data structures can be used for a fast evaluation of the neighbours, which are especially important to allow for the efficient implementation of a best-improvement pivoting rule, where the best neighbouring solution replaces the current one. To this aim, one can define for the split graph and for a fixed k, the matrix $\mathbf{\Delta}$ with $|V^S| \times k$ elements and each value $\mathbf{\Delta}(v, c)$ indicates the contribution to f of assigning colour c to vertex $v \in V^s$. This matrix can be initialised and updated taking into account of the usual speed-up techniques for the GCP; however, due to the separation distance constraints, the update of $\mathbf{\Delta}$ is by a factor t, where t is the maximum occurring separation distance, more complex than in the GCP case.

In a local search for the GSTCP formulation, essentially the same speed-ups apply; in addition, another matrix $\mathbf{\Delta}_2$ of size $|V| \times k$ is maintained to forbid the assignment of colours that break vertex constraints.

Vertex exact colour reassignment neighbourhood. In our implementation, we search N_R by considering in random order the vertices currently involved in at least one conflict. For each chosen vertex, a set F is determined that comprises the colours which are proper given the edge constraints and the colours assigned to the adjacent vertices. The construction of this set can be done in $\mathcal{O}(|V|k)$ using the auxiliary data structures introduced in the previous paragraph. If one simply looked for $r(v)$ colours from F such that the vertex constraints are satisfied, this would be easy: it suffices indeed to order the values in F and scan the set once, skipping the values that are not sufficiently distant from the previous ones. However, this procedure is deterministic and, when visiting a vertex a next time, the search would use the same colour reassignment if nothing had changed in the adjacent vertices. To avoid this cycling behaviour, some randomisation in the reassignment should be introduced to possibly propagate a different configuration.

One possibility for implementing this strategy is to determine all subsets of F of size $r(v)$ that satisfy the vertex constraints and pick one at random.[10]

We solve the problem of determining all the proper subsets of F in a dynamic-programming fashion. Given the ordered sequence of integers in F, a proper colouring is an ordered subsequence of integers composed by other subsequences, each allowing a number of proper solutions, corresponding to

[10] This problem, hence, involves the generation of all subsequences of length L of a set of H integers, $H > L$, such that all pair-wise distances between the integers of the subsequence are larger than a constant D.

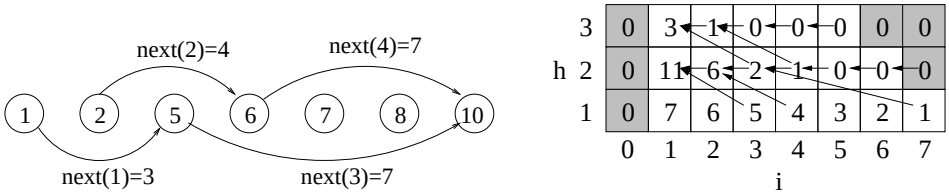


Figure 5: An example of vertex exact colour reassignment for a case in which $F = \{1, 2, 5, 6, 7, 8, 10\}$, $L = |F| = 7$, $D = t_v = 4$ and $H = r(v) = 3$. On the left the vector s of 7 integers. For each integer in the sequence the pointer $next()$ is computed, where not indicated it is set to 0. On the right the table of $N_h[i]$ values. Its construction starts from the low right corner. Arrows indicate the stored values that are used to compute the entries. The grey cells indicate the values without a proper meaning and hence assigned by convention.

the different ways the subsequence can be extended to a sequence of length $r(v)$ by adding elements from F. The total number of such solutions can be defined recursively and computed in a bottom-up fashion. Afterwards it is possible to choose one solution uniformly at random.

More specifically, let s be the ordered vector of integers in F, $L = |F|$, $D = t_v$ and $H = r(v)$. Then for each position i of the vector s we define $next(i) = \min_j\{j | j > i,\ s[j] - s[i] \geq D\}$. For each subsequence l of s, the number $N_H[i]$ of proper subsequences of s of length $h \in \{1, \ldots, H\}$ containing $l = \{s[1], \ldots, s[i]\}$, can be determined by the recursion

$$N_h[i] = \begin{cases} L - i + 1 & \text{if } h = 1 \\ N_{h-1}[next(i)] + N_h[i+1] & \text{if } L - i - 1 > h \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

if $i \geq 1$ and $N_h[0] = 0$ by convention.

Hence, the total number of proper subsequences of length $r(v)$ corresponds to $N_H[1]$. To select one solution at random among the $N_h[i]$ solutions one has then simply to scan the sequence s and select each element with probability $N_{h-1}[next(i)]/N_h[i]$, since this is the fraction of the extensions which contain the element in question. Whenever an element is chosen the scanning moves to $next(i)$. Scanning the sequence of numbers takes linear time, but computing each time the recursion 5 takes exponential time. However, this can be done much more efficiently if all values $N_h[i]$ are computed at the beginning and recorded in a table. Referring to Figure 5, right, if the table is filled from bottom to top and from right to left within each row, each new entry needs the values of $N_{h-1}[next(i)]$ and $N_h[i+1]$ which are already determined and stored. The $next$ function and the table together can then be computed in $\mathcal{O}(\max(D, H, \log L)L)$ while to chose a subsequence randomly a further scan of the sequence s is needed.

6.3 SLS Algorithms

Based upon the three schemes for tackling the GSTCP and the neighbourhood examination schemes, we have implemented various SLS algorithms. These include re-implementations of previously proposed algorithms for the GSTCP or their extension, the new adaptation of high-performing SLS algorithms for the GCP to solve the GSTCP, and newly developed algorithms. In all cases, we solve the optimisation version of the GSTCP by starting from a k and a colouring returned by the G-DSATUR heuristic, and solving a sequence of decision problems, trying to minimise the number of colours and, hence, also the span. (The only exception to this are the algorithms where k is kept variable.) Once a proper colouring for the decision problem with k colours is found, k is decreased by one and the vertices which had colour k are recoloured randomly. The best solution returned is the minimal value of k for which a proper set T-colouring is found.

6.3.1 SLS algorithms for Scheme 1: split graph, k fixed

Tabu Search. We implemented a standard Tabu Search procedure that chooses at each iteration a best non-tabu move or a tabu but "aspired" neighbouring solution from N_E. The tabu list forbids to reverse a move and the tabu tenure is chosen as $tt = random(10) + 2\delta|V^c|$, where V^c is the set of vertices which are involved in at least one conflict, δ is a parameter, and $random(10)$ is an integer random number uniformly distributed in $[0, 10]$; this choice follows a successful tabu search algorithm for the GCP by Galinier and Hao (1999). We denote this algorithm SF-TS.

SF-TS is very similar to the Tabu Search algorithms proposed in Costa (1993), Castelino et al. (1996), and Hurley et al. (1997). In those papers, Tabu Search was reported to perform better than Simulated Annealing and Genetic Algorithms. Our tests with a Simulated Annealing algorithm of the FASoft system Hurley et al. (1997) for the `GEOM` benchmark set confirmed these results.

Min-Conflicts. The min-conflicts heuristic (Minton et al., 1992) is based on N_E but implements a particular two-stage process for examining the neighbourhood. In a first stage, a vertex is chosen uniformly at random from the set V^c; in a second stage, this vertex is assigned a colour such that the number of conflicts is minimised, breaking ties randomly. As shown in Stützle (1998), the min-conflicts heuristic can be profitably enhanced by Tabu Search, where the reversal of a move is forbidden. If all colours of a vertex are tabu, one is chosen randomly. One advantage of this neighbourhood examination strategy is that it allows for an easier implementation, since it does not require the usage of auxiliary data structures as the $\mathbf{\Delta}$ matrix described in the previous section. In addition, the chances of cycling are reduced due to the random choices in the first stage of the selection process, allowing for shorter tabu lists. The tabu tenure is set constantly to an integer tt during the whole search.

Guided Local Search. Guided local search (GLS) is an SLS strategy that modifies the evaluation function in order to escape from local optima (Voudouris, 1997). An application of GLS to the GCP has shown excellent performance on geometric graphs (Chiarandini et al., 2005). GLS uses a penalty weight w_i that is associated to each edge i and that is initialised to one. On the GTCP, the algorithm derived from this strategy, SF-GLS, uses an augmented evaluation function g' defined as

$$g'(s) = f(s) + \lambda \cdot \sum_{i=1}^{|E^s|} w_i \cdot I_i(s),$$

where $f(s)$ is the usual evaluation function, λ a parameter that determines the influence of the penalties on the augmented cost function, and $I_i(s)$ an indicator function that takes the value 1 if the end points of edge i are in conflict in candidate solution s and 0 otherwise. The penalties are initialised to 0 and are updated each time an iterative improvement algorithm reaches a local optimum of g'. The modification of the penalty weights is done by first computing a utility u_i for each violated edge, $u_i = I_i(s)/1 + w_i$, and then incrementing the penalties of all edges with maximal utility by one. The underlying local search is a best-improvement algorithm on N_E. When a local optimum is reached, the search continues for a maximum number of sw plateau moves before the evaluation function g' is updated.

GLS was already applied in the context of frequency assignment by Tsang and Voudouris (1998). Their local search is more complex than ours since it includes the use of *don't look bits* to avoid the repetition of recent exchanges. This difference is, however, not significant, as we restrict the neighbourhood to vertices involved in conflicts and their method imposes similar restrictions.

Hybrid Evolutionary Algorithm. A hybrid evolutionary algorithm was shown to be a top performer for the GCP (Galinier and Hao, 1999). This algorithm uses a Tabu Search procedure as the underlying local search and the adaptation of this hybrid algorithm to the GTCP we denote as SF-HEA. SF-HEA starts with a population P of candidate solutions and then iteratively generates new candidate solutions by first re-combining two members of the current population that are improved by SF-TS. The recombination operator, which is deemed responsible for the high performance of this algorithm on the GCP, is the greedy partition crossover (GPX) Galinier and Hao (1999). The new candidate partition returned by GPX is then improved by SF-TS, run for l_{LS} iterations, and is inserted in the population P replacing the worse parent. The population is re-initialised if the average distance between colourings in the population falls below a threshold of 20. Beside the parameter δ of SF-TS, SF-HEA requires also a parameter to determine the number of iterations allowed for executing SF-TS.

6.3.2 Scheme 2: original graph, k fixed, complete colourings

Tabu Search. An application of Tabu Search using the one-exchange neighbourhood N'_E on the original graph was designed by Dorne and Hao (1998). Other versions of this algorithm for frequency assignment (Hao et al., 1998) differ only in the management of the tabu length or are more rudimentary (Hao and Perrier, 1999). Our version uses the same tabu tenure definition as SF-TS and is in this equal to the version by Dorne and Hao (1998). We denote this algorithm OF-TS.

We also include two enhanced versions of OF-TS which make use of the vertex reassignment neighbourhood N_R. Since the exploration of the union of N'_E and N_R would be computationally expensive, we adopt a heuristic rule for choosing the next move to apply. For short, first the best non tabu move in the neighbourhood N'_E is determined. If it improves on the current solution, it is accepted. If it leaves the evaluation function value unchanged or worsens it, a move is searched in the neighbourhood N_R, restricted to vertices involved in at least one conflict. If a proper reassignment is found, it is applied, otherwise the best non-tabu move in N'_E is applied. We call the overall algorithm OF-TS+R.

A variant of OF-TS+R considers a random vertex from V in case no move is found in N_R restricted to conflicting vertices. The motivation for this is that a random reassignment of colours to vertices, where no conflict is present, may produce a change that can propagate profitably. We denote this variant OF-TS+R*.[11]

The Tabu Search mechanism applied to moves in N'_E is the same used in OF-TS and Dorne and Hao (1998) (aspiration criterion included). No tabu search mechanism is instead applied to moves in N_R. In this case, repetitions in the search are avoided by the randomisation of the reassignment. This is the reason why preliminary experiments clearly indicated that the use of a randomised reassignment instead of a deterministic one results in much better performance.

Squeaky Wheel Optimisation. The iterated greedy algorithm by Culberson (1992) for the GCP consists in applying the greedy method repeatedly each time to a new permutation of the vertices that is derived by keeping vertices with the same colour in the previously generated colouring in consecutive positions of the permutation. A nice theorem shows that by generating the permutations as input to a greedy colouring procedure, this will result in a solution that does not use more colours than the previous colouring. On the GSTCP, due to the distance separation constraints, the result of the theorem is not true anymore. In fact, only permutations in which the distance between colour classes remains the same will guarantee such a result. One such permutation is the reverse order of the colour classes.

[11] By chance, a randomly chosen vertex can happen to be in conflict; in this case the best non tabu move in N_E is chosen as in OF-TS+R. Clearly, this case can be avoided in another implementation.

This algorithm has inspired the application of Squeaky Wheel Optimisation (Joslin and Clements, 1999) to the GSTCP by Lim et al. (2003). The method works in three phases: construction of a solution, analysis and identification of trouble makers, and priorisation of the trouble makers, which are thus handled earlier by the constructor in the next iteration. In the description of their first application (Lim et al., 2003) there is a lack of details which made impossible a precise re-implementation of the algorithm. Instead, we implemented our own version. The algorithm works as follows. An initial solution is determined by a construction heuristic. Then, a colouring that uses $k-1$ colours is constructed applying the T-greedy algorithm to a permutation of the vertices obtained from the previous colouring. The permutation maintains vertices that have the same colour in consecutive positions but reverses the order of the colour classes and shuffles vertices in the classes. The rationale behind this procedure is that, if the reverse order of colour classes yields a colouring that uses the same or a lower number of colours, then it is possible that this order, or small variations thereof, lead to a feasible colouring with $k-1$ colours. Perturbations in the reverse order are introduced by inserting vertices that had colour k before vertices that had colour b, with $b \in \{k-1, \ldots, 1\}$. Each colouring generated by T-greedy on a proposed permutation of vertices is then improved by OF-TS applied for $\mu \cdot V^c \cdot \sum_{v \in V} r(v)$ iterations. If no feasible solution is found and all possible values of b have been attempted, the search proceeds by OF-TS solely. We denote this algorithm as OF-SW. Note that the algorithm can be used both with OF-TS and with SF-TS. The preference for OF-TS is the result of preliminary experiments. In addition, we include another variant OF-SW+R, which uses the OF-TS+R as Tabu Search procedure.

More recently Lim et al. (2005) published an enhanced version of their algorithm. This version differ from ours in the fact that 5% of colours are placed after b in the new permutation and b is kept fixed to 8. Moreover their algorithm uses a Tabu Search on the split graph based on a swap neighbourhood. Experiments, however, show that our version produces often results superior to those reported in Lim et al. (2005).

6.3.3 Scheme 3: split graph, *k* variable

Tabu Search. We implemented a Tabu Search algorithm in all equal to SF-TS except that it the evaluation function of Equation 4. We denote this algorithm as SV-TS.

7 Empirical Assessment of SLS Algorithms

In this section, we experimentally compare the 10 SLS algorithms described in Section 6.3. In addition, we use also an algorithm proposed in Prestwich (2002b, 2003), called FCNS. This algorithm is a tree search algorithm that uses an incomplete randomised form of dynamic backtracking. The implementation of this algorithm was kindly made available by Prestwich.

The goals of this empirical analysis are to determine: (i) the best approach to solve the GSTCP (using k fixed versus variable, GTCP versus GSTCP formulation); (ii) the impact of the colour reassignment neighbourhood; and (iii) the best overall algorithm.

Differently from construction heuristics, the SLS algorithms studied here do not have a natural stopping criterion. To make the comparison fair among the algorithms, we need to allocate the same amount of resources. Since we cannot use the number of steps per algorithms, because they entail different computational effort among the algorithms, we use for all algorithms a common CPU-time limit. This approach is feasible, since the algorithms are all implemented in a same framework, sharing data structures where reasonable and they were implemented by the same person.[12]

7.1 Experimental setup

CPU time limit. For defining the CPU-time limit, we use OF-TS as a reference algorithm. This algorithm is run for $I_{max} = 10^5 \times \sum_{v \in V} r(v)$ iterations and the required computation time is measured; note that by the choice of I_{max}, the number of iterations increases with the number of vertices and the requirements.[13] The iteration limit was chosen such that on one side the algorithm reaches limiting behaviour and a further improvements in solution quality become increasingly difficult, but that on the other side the overall experimental study is still feasible with respect to overall computation time consumption. (For details on the analysis for determining the stopping criterion, we refer to Chiarandini 2005.) A second issue in the determination of the time limit is the stochasticity of the computation time taken by OF-TS, both because of the differences observed when running OF-TS on a same instance and also within instances of a same class. In order to have a uniform time limit among instances of the same class, we use a regression model on the 5 variables $|V|, \rho, \bar{r}, \bar{t}_{vv}, \bar{t}_{uv}$ and set as a time limit for each algorithm the one predicted by this model. The details of the regression model are given in Appendix A.[14]

Algorithm parameter tuning. All SLS algorithms in the comparison require some parameters to be tuned for the classes of problem instances and for the given computation time limit decided. In Table 3 we resume the parameters presented in Section 6.3 and for each of them the values tested and the values selected.

[12] Certainly, this does not remove any possible bias towards one algorithm; however, several factors that influence computation times like programming languages, computing environment etc. are excluded from being the main responsible for the observed differences.

[13] Only on the HD-RU set we adjusted the iteration limit to $I_{max} = 10^4 \times \sum_{v \in V} r(v)$, because of the large size of these instances and the very high resulting computation times.

[14] To give an impression of the computation times involved, we mention that the longest computation time for any uniform random instances with 60 vertices resulted to be about 3 hours and 30 minutes. For details, see the tables in Appendix B.

	NRU, HD-RU	GEOM	Philadelphia
OF-TS	δ ={0.5; 1; **10**; 20; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; **20**; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; 20; 30; **40**; 50; 60; 70; 100
OF-TS+R	δ ={0.5; 1; **10**; 20; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; **20**; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; 20; 30; **40**; 50; 60; 70; 100
OF-TS+R*	δ ={**20**; 40}	δ ={**20**; 40}	δ =20; **40**}
SF-TS	δ ={0.5; 1; **10**; 20; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; **20**; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; 20; 30; **40**; 50; 60; 70; 100
SV-TS	δ ={0.5; 1; **10**; 20; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; **20**; 30; 40; 50; 60; 70; 100}	δ ={0.5; 1; 10; 20; 30; **40**; 50; 60; 70; 100
SF-HEA	μ = {1000, **10000**}; δ = {**20**,40}	μ = {1000, **10000**}; δ = {**20**,40}	μ = {1000, **10000**}; δ = {**20**,40}
SF-GLS	λ = 1; sw = {1, **20**, 100, 200}	λ = 1; sw = {1, **20**, 100, 200}	λ = 1; sw = {1, **20**, 100, 200}
SF-MC	tt = {2, 10, **20**, 30}	tt = {2, 10, **20**, 30}	tt = {2, 10, **20**, 30}
FCNS	B=1	B=1	B=1

Table 3: The parameter values tested and selected (in bold) for the SLS algorithms.

The experimental design. We maintain four separated set of instances corresponding to the GEOM, the NRU, the HD-RU and the Philadelphia instances. We collected 5 runs per instance on all Philadelphia instances and 3 runs per instance on the other three sets. This amounts to a total of about 207 days of CPU-time on our reference machine (see Section 3), not taking into account the preliminary experiments for the tuning.[15] The instances tested comprise all those that are indicated in Table 1.

Statistical analysis. In all instance sets, diagnostic plots for checking the assumptions of parametric tests on different transformations of the results suggest that a parametric analysis is not appropriate. We therefore rely on rank-based tests. Results are ranked within each instances among the 11 algorithms and across the number of runs collected. (That is, the ranks on each instance range from 1 to 33, if 3 runs are collected per instance.) Within each instance set we divide the analysis on instance classes in order to take into account the interactions between algorithm performance and instance characteristics. Statistical significance of differences among the average ranks of the algorithms within each class is tested by the Friedman sum rank test (Conover, 1999). We present the results by means of the simultaneous confidence intervals produced by this test. Differences in the classes can be visually examined in Figures 6 to 9 for the four instance sets. On the y-axis, algorithms are ordered according to their overall performance in the instance set, the best algorithms being closer to the origin. On the x-axis is given the average rank with the confidence interval. Two algorithms are significantly different if their confidence intervals do not overlap. For a synthesis of the numerical results underlying these graphs we refer to Appendix B.

[15] The experiments were in fact run on a cluster of machines of equal computational power.

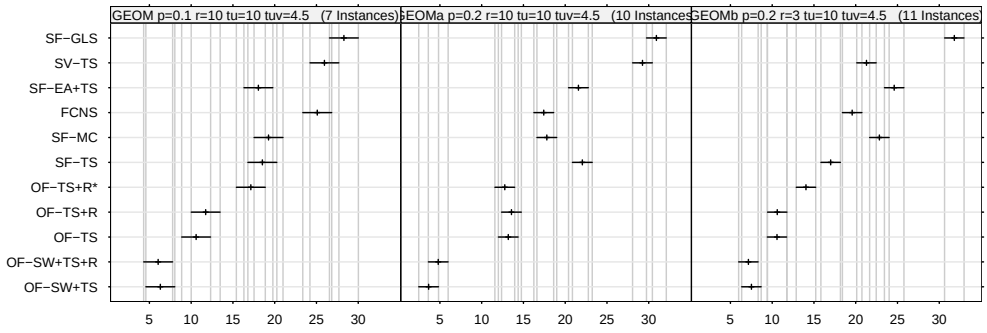


Figure 6: Simultaneous confidence intervals for SLS algorithms on the GEOM instance set of the GSTCP. Each algorithm was run 3 times per instance with a time limit corresponding to $10 \times I_{max}$ iterations of OF-TS. See the text for further details.

7.2 Experimental results

In this section, we summarise the computational results with respect to the three goals of our analysis. As the overall picture one can summarise that no algorithm dominates all others across all instance classes or benchmark sets and that the results, in part, depend strongly on the characteristics of the particular instances tackled.

Best approach for GSTCP. The experimental results for the construction heuristics identified a heuristic that makes use of the split graph as the overall best performing one. For the SLS algorithms, a completely different picture arises. On none of the instance classes, SLS algorithms using the split graph could reach results competitive to the better performing algorithms working on the original GSTCP instance. One interpretation of this result may be that it is important to maintain the vertex separation distance constraints satisfied in the local search and, thus, to reduce the effective search space, which is the case for all SLS algorithms working on the original GSTCP graph.

Additionally, keeping the number of colours k variable during the local search also proved not to be successful. This can be seen by the fact that SV-TS typically performed significantly worse than SF-TS for most instance classes. However, this latter assertion has to be taken with care, since SV-TS is a quite simplistic algorithm and maybe much better ones could be devised.

Colour reassignment neighbourhood. The comparisons of OF-TS *vs.* OF-TS+R and OF-SW *vs.* OF-SW+R show that the usefulness of the colour reassignment neighbourhood N_R is dependent on the particular benchmark set and on the instance classes in the set. On the GEOM set, the usage of N_R does not yield any improvement but its usage does not result in significantly worse performance. On the NRU set, however, the usage of N_R results in sta-

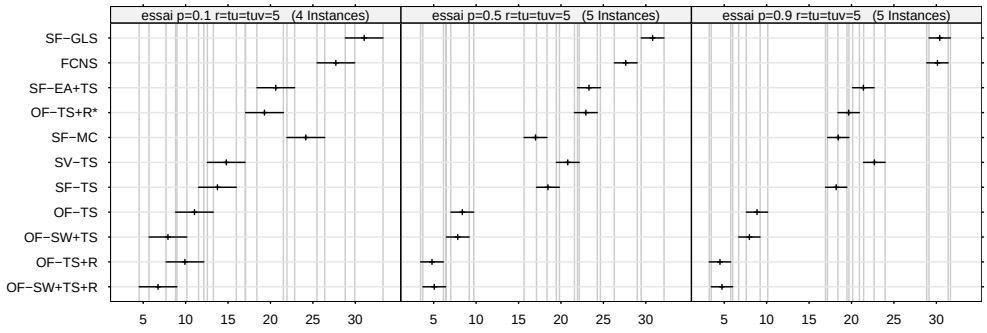


Figure 7: Simultaneous confidence intervals for SLS algorithms on the HD-RU instance set of the GSTCP. Each algorithm was run 3 times per instance with a time limit corresponding to I_{max} iterations of OF-TS. The three columns correspond to the three different densities of the graphs. See the text for further details.

tistically significant better performance on various instance classes. From a closer inspection of the results, it is possible to conjecture that this neighbourhood becomes profitable with the increase of edge density in the graph and vertex requirements. Finally, on the Philadelphia set, OF-TS+R*, a variant of OF-TS+R, is the best performing algorithm.

In Table 4, we try to gain insight into the contribution of N_R in OF-TS+R. OF-TS+R was run 10 times on a single instance from the corresponding class with a bound on the number of iterations equal to 10000. The values in the first column are the number of iterations in which no improving one-exchange move was found and the neighbourhood N_R was explored. The values in the second and third columns are the total number of improvements (either in N_E or N_R) and the number of improvements in N_R, respectively. We recall that the exploration of N_R is performed only when no improving one-exchange move is found and, therefore, the values in the third column indicate the improvements that would be missed without the use of N_R. The values in the last two columns are the average size of the set F (see Section 6.2) and its range of integers. We observe that at least for these instances, $|F|$, tends to be very close to the number of colours required at each vertex.

Best algorithm for the instance classes. As said, the SLS algorithms working on the split graph give overall worse performance than those working on the original problem instances. Among the SLS algorithms working on the split graphs, it is noteworthy that SF-GLS and SF-HEA, adaptations of two algorithms which belong to the state-of-the-art for solving specific instance classes of GCP, perform very poorly, for most instance classes significantly worse than SF-TS. Similarly, FCNS also did not result to be competitive.

We now focus on the algorithms working on the original GSTCP graph and in particular on the comparison between the Tabu Search algorithms and those

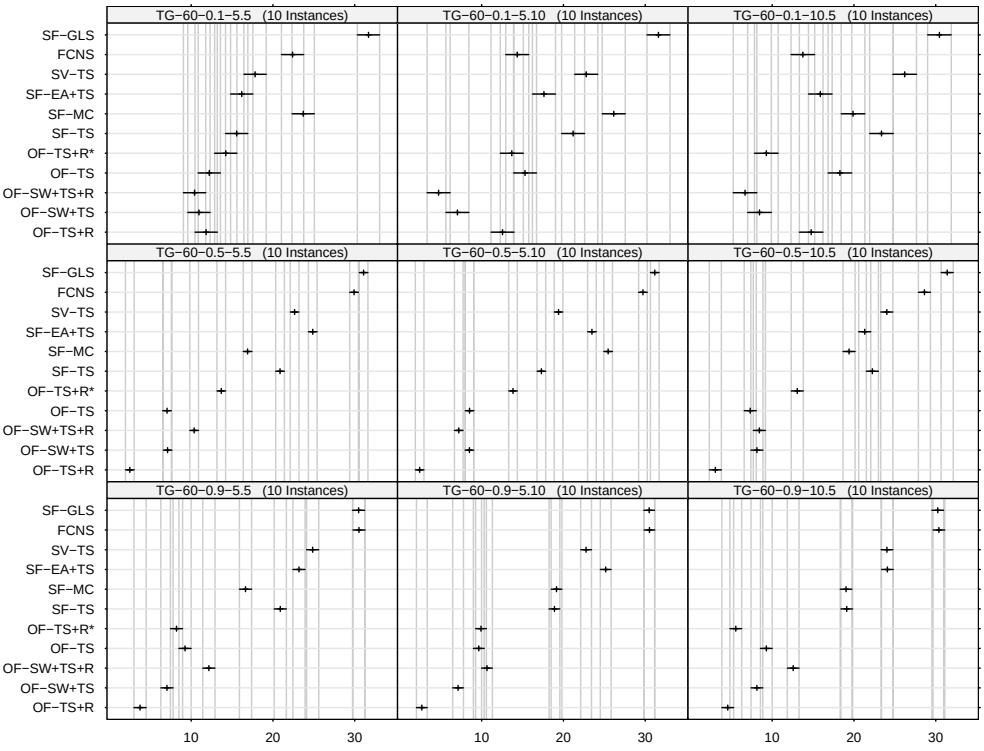


Figure 8: Simultaneous confidence intervals for SLS algorithms on the NRU instance set of the GSTCP. Each algorithm was run 3 times per instance with a time limit corresponding to $10 \times I_{max}$ iterations of OF-TS. The three rows comprise the instances for values of p equal to 0.1, 0.5, and 0.9, respectively; in each column the parameters for the requirements and separation distances are kept constant. See the text for further details.

based on Squeaky Wheel Optimisation. Squeaky Wheel Optimisation is clearly the best performing algorithm on the GEOM set, while on the Philadelphia set it is dominated by the Tabu Search variants. Given that the graphs of GEOM and the Philadelphia sets should not differ too much in their structure, the reason for such a different behaviour may be that the Philadelphia instances have much higher vertex requirements than the others, which may indicate that Squeaky Wheel Optimisation works worse for large vertex requirements. On the NRU set, the best results are typically found by OF-TS+R, *i.e.*, the Tabu Search with the colour reassignment neighbourhood, although Squeaky Wheel Optimisation remains competitive on the low density graphs.

Comparison with the results in the literature. In Appendix B we give detailed results of our algorithms, which allows a comparison to the best known values reported in the literature (Phan and Skiena, 2002; Prestwich, 2002a, 2003; Lim et al., 2003, 2005 and the FAP website). We point out the following

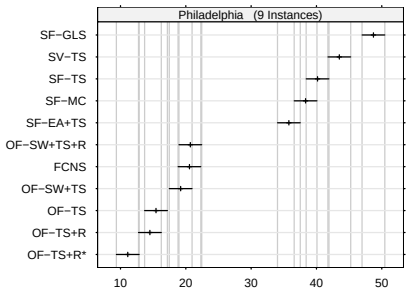


Figure 9: Simultaneous confidence intervals for SLS algorithms on the Philadelphia instance set of the GSTCP. Each algorithm was run 5 times per instance with a time limit corresponding to $10 \times I_{max}$ iterations of OF-TS. See text for further details.

class	iter.	total improv.	improv. in $\mathcal{N}_A$	$\lvert F \rvert$	$f_{max} - f_{min}$
T-G.5.5-60.0.1	8880	1179	59	7	25
T-G.5.10-60.0.1	9259	839	98	8	50
T-G.10.5-60.0.1	9994	6	0	26	73
T-G.5.5-60.0.5	9030	776	194	6	43
T-G.5.10-60.0.5	9187	926	113	6	133
T-G.10.5-60.0.5	9287	851	138	9	69
T-G.5.5-60.0.9	9232	651	117	5	53
T-G.5.10-60.0.9	9807	223	30	8	159
T-G.10.5-60.0.9	9437	640	77	10	209

Table 4: Report statistics on OF-TS+R. Data are collected on one instance of the corresponding class and are averages out of 10 algorithmic runs with a bound of 10000 iterations.

observations.

On the GEOM instance set, our version of Squeaky Wheel attains better results than the version by Lim et al. (2005). The results in the column best results are indeed almost all derived by that paper. Our best results found by OF-SW are better on 16 instances (with improvements of even 19 and 22 colours) and equal on 7, while they are worse only on 5 instances. Moreover the results by Lim et al. (2005) are the best out of 10 runs while in our case we collected only 3 runs per instance. Computation times seem to be longer in our case but a fair comparison is not possible because Lim et al. (2005) report only the time to the best solution found, while we prefer to give more importance to the whole time limit allocated to the algorithm.

On the HD-RU set we observe that our G-DSATUR is much better then the DSATUR presented by Dorne and Hao, while the results of OF-TS are inferior to those reported for the implementation of the same algorithm by Dorne and Hao (1998) and they remain inferior even for our best algorithm in that class, OF-SW+R. This difference may be due to the difference in the maximum number of iterations given to the two Tabu Search algorithms. In fact, for each decision problem in the optimisation process, Dorne and Hao allow 10^7 or even $2 \cdot 10^7$ iterations, which results in much longer computation

times (up to 7 days on their machine), while in our experimental setup, we limit the number of iterations across the full sequence of decision problems to often a lower value than 10^7. In fact, by comparing the computation times reported in Dorne and Hao (1998) and adjusting the machine speed, they used approximately 20 times or more computation time than in the experiments we reported.

On the NRU instance set, we have no comparison with known results. We aggregated therefore the results of the 10 instance per class. In general, we can observe a considerable improvement over the solution provided by G-DSATUR, indicating also the usefulness of the SLS algorithms.

On the Philadelphia instance set, our G-DSATUR is in general better than the DSATUR implemented in the FASoft system (Hurley et al., 1997). If compared with the best solution returned by a portfolio of 64 sequential heuristics (DSATUR included) of FASoft, our G-DSATUR is able to improve their solution on 6 out of 9 instances. Hence, G-DSATUR would also be an excellent candidate for extending the algorithm portfolio of FASoft. As for OF-TS+R*, it produces 7 times out of 9 the optimal solution. In the literature, only the algorithm by Matsui and Tokoro (2001) reached similar performance with 8 out of 9 optimal solutions found.

8 Conclusions

In this paper, we have proposed new algorithms for the GSTCP and presented the results of a comprehensive experimental study of construction heuristics and more performing SLS algorithms. We have solved the problem considering the minimal span objective. The experimental study comprised four benchmark sets, including the random geometric instances proposed by M. Trick, new and previously proposed benchmark instances based on uniform random graphs and the Philadelphia instances from the frequency assignment. This latter instance set permitted us to compare directly the algorithms studied only on the theoretical model, the Graph Set T-Colouring Problem with those developed for its practical application, the Frequency Assignment.

The study of the construction heuristics showed that our new version of G-DSATUR, a generalised form of DSATUR, solving the GTCP transformation of the GSTCP by colouring first vertices with *smallest* saturation number and using an adjusted vertex degree for breaking ties, is the best choice on both Geometric and Uniform graphs. In terms of solution quality, it performs better than the ROS and RLF heuristics adapted for solving the GTCP and it often performs better than a portfolio of 64 construction heuristics for the Frequency Assignment. In terms of computation times it is rather efficient and it produces a solution on split graphs of 3000 vertices in about a couple of seconds on a typical office PC.

The comparison of SLS algorithms indicated that, if the goal is to minimise the span, the best approach for tackling the GSTCP is to solve it as a sequence of decision problems with the number of colours fixed at each stage and to use

the GSTCP formulation, which enforces the requirement and separation vertex constraints to be always satisfied. However, we could not identify one single algorithm as the best overall algorithm. On the Geometric instance set, our re-implementation of a Squeeky Wheel Optimisation algorithm performed best and we were able to produce 16 new best results on the benchmark instances. On the Random Uniform graphs, our new Tabu Search algorithm with the colour reassignment procedure is overall the best performing algorithm. This tendency is also confirmed on the Philadelphia instances. Since these instances resemble the Geometric instances but have much higher vertex requirements, we conjecture that the usefulness of the exact colour reassignment neighbourhood increases with the requirements. On the Philadelphia instances, our new algorithm OF-TS+R* finds 7 times out of 9 the optimal solution which places it among the best algorithms reported in the FAP literature.

There are a number of possible ways to extend the research reported here. One possibility is to further develop the Tabu Search algorithm that is enhanced by an exact algorithm to explore the colour reassignment neighbourhood. The success of this algorithm indicates that also other methods for the examination of very large neighbourhoods may be useful to enhance SLS algorithms for the GSTCP. Another direction would be to extend the experimental study to other, related problems from the frequency assignment literature, where there is also a need for satisfying additional constraints, such as list colouring and preassignments. Ultimately, we hope that our experimental study will contribute to the interest for further work in this direction.

Acknowledgements. We acknowledge the support of the Frankfurt Center for Scientific Computing, who made available its cluster for running the experiments. Thomas Stützle acknowledges support of the Belgian FNRS, of which he is a research associate.

References

Aardal K.I., van Hoesel C.P.M., Koster A.M.C.A., Mannino C., and Sassano A. (2001). "Models and solution techniques for the frequency assignment problem". ZIB-report 01–40, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany.

Allen S.M., Smith D.H., and Hurley S. (1999). "Lower bounding techniques for frequency assignment". *Discrete Mathematics*, 197/198, pp. 41–52.

Anderson L.G. (1973). "A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system". *IEEE Transactions on Communications*, 21, pp. 1294–1301.

Bartz-Beielstein T. and Markon S. (2004). "Tuning search algorithms for real-world applications: A regression tree based approach". In *Congress on Evo-*

lutionary Computation (CEC'04), pp. 1111–1118. IEEE Press, Piscataway NJ.

Birattari M., Stützle T., Paquete L., and Varrentrapp K. (2002). "A racing algorithm for configuring metaheuristics". In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, edited by W.B. Langdon et al., pp. 11–18. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Borndörfer R., Eisenblätter A., Grötschel M., and Martin A. (1998). "Frequency assignment in cellular phone networks". *Annals of Operations Research*, 76, pp. 73–93.

Breiman L., Friedman J., Olshen R.A., and Stone C.J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, USA.

Castelino D., Hurley S., and Stephens N. (1996). "A tabu search algorithm for frequency assignment". *Annals of Operations Research*, 63, pp. 301–320.

Chiarandini M. (2005). *Stochastic Local Search Methods for Highly Constrained Combinatorial Optimisation Problems*. Ph.D. thesis, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany.

Chiarandini M., Stützle T., and Dumitrescu. I. (2005). "Stochastic local search algorithms for the graph colouring problem". In *Approximation Algorithms and Metaheuristics*, edited by T.F. Gonzalez. Chapman & Hall/CRC. To appear.

Conover W. (1999). *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, third ed.

Costa D. (1993). "On the use of some known methods for T-colorings of graphs". *Annals of Operations Research*, 41, pp. 343–358.

Cozzens M.B. and Roberts F.S. (1982). "T-colorings of graphs and the channel assignment problem". *Congressus Numerantium*, 35, pp. 191–208.

Culberson J. (1992). "Iterated greedy graph coloring and the difficulty landscape". Tech. Rep. 92-07, Department of Computing Science, The University of Alberta, Edmonton, Canada.

Culberson J., Beacham A., and Papp D. (1995). "Hiding our colors". In *Proceedings of the CP'95 Workshop on Studying and Solving Really Hard Problems*, pp. 31–42. Cassis, France.

Dorne R. and Hao J. (1998). "Tabu search for graph coloring, T-colorings and set T-colorings". In *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pp. 77–92. Kluwer Academic Publishers.

Eisenblätter A., Grötschel M., and Koster A.M.C.A. (2002). "Frequency assignment and ramifications of coloring". *Discussiones Mathematicae Graph Theory*, 22, pp. 51–88.

Galinier P. and Hao J. (1999). "Hybrid evolutionary algorithms for graph coloring". *Journal of Combinatorial Optimization*, 3(4), pp. 379–397.

Garey M.R. and Johnson D.S. (1979). *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco, CA, USA.

Giaro K., Janczewski R., and Malafiejski M. (2003a). "The complexity of the T-coloring problem for graphs with small degree". *Discrete Applied Mathematics*, 129(2-3), pp. 361–369.

Giaro K., Janczewski R., and Malafiejski M. (2003b). "A polynomial algorithm for finding T-span of generalized cacti". *Discrete Applied Mathematics*, 129(2-3), pp. 371–382.

Hale W.K. (1980). "Frequency assignment: Theory and applications". *Proceedings of the IEEE*, 68(12), pp. 1497–1514.

Hao J.K., Dorne R., and Galinier P. (1998). "Tabu search for frequency assignment in mobile radio networks". *Journal of Heuristics*, 4(1), pp. 47–62.

Hao J.K. and Perrier L. (1999). "Tabu search for the frequency assignment problem in cellular radio networks". Tech. Rep. LGI2P, EMA-EERIE, Parc Scientifique Georges Besse, Nimes, France.

Hoos H. and Stützle T. (2004). *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Hurley S., Smith D.H., and Thiel S.U. (1997). "FASoft: A system for discrete channel frequency assignment". *Radio Science*, 32(5), pp. 1921–1939.

Janssen J. and Narayanan L. (2001). "Approximation algorithms for the channel assignment problem". *Theoretical Computer Science*, 262, pp. 649–667.

Janssen J. and Wentzell T. (2000). "Lower bounds from tile covers for the channel assignment problem". Tech. Rep. G-2000-09, GERAD, HEC, Montreal, Canada.

Johnson D.S., Mehrotra A., and Trick M. (eds.) (2002). *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*. Ithaca, New York, USA.

Joslin D.E. and Clements D.P. (1999). "Squeaky wheel optimization". *Journal of Artificial Intelligence Research*, 10, pp. 353–373.

Lim A., Zhang X., and Zhu Y. (2003). "A hybrid method for the graph coloring and related problems". In *Proceedings of MIC'2003 – The Fifth Metaheuristics International Conference*. Kyoto-Japan.

Lim A., Zhu Y., Lou Q., and Rodrigues B. (2005). "Heuristic methods for graph coloring problems". In *SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing*, pp. 933–939. ACM Press, New York, NY, USA.

Matsui S. and Tokoro K. (2001). "Improving the performance of a genetic algorithm for minimum span frequency assignment problem with an adaptive mutation rate and a new initialization method". In *Proc. of GECCO-2001 (Genetic and Evolutionary Computation Conference)*, pp. 1359–1366. Morgan Kaufmann Publishers.

McGeoch C., Sanders P., Fleischer R., Cohen P.R., and Precup D. (2002). "Using finite experiments to study asymptotic performance". In *Experimental Algorithmics: From Algorithm Design to Robust and Efficient Software*, edited by R. Fleischer, B. Moret, and E.M. Schmidt, vol. 2547 of *Lecture Notes in Computer Science*, pp. 93–126. Springer Verlag, Berlin, Germany.

Minton S., Johnston M., Philips A., and Laird P. (1992). "Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems". *Artificial Intelligence*, 58(1-3), pp. 161–205.

Montemanni R. (2001). *Upper and lower bounds for the fixed spectrum frequency assignment problem.* Ph.D. thesis, Division of Mathematics and Statistics, School of Technology, University of Glamorgan, UK.

Montgomery D.C. (2005). *Design and Analysis of Experiments.* John Wiley & Sons, sixth ed.

Phan V. and Skiena S. (2002). "Coloring graphs with a general heuristic search engine". In Johnson et al. (2002), pp. 92–99.

Prestwich S. (2002a). "Coloration neighbourhood search with forward checking". *Annals of Mathematics and Artificial Intelligence*, 34(4), pp. 327–340.

Prestwich S. (2002b). "Constrained bandwidth multicoloration neighbourhoods". In Johnson et al. (2002), pp. 126–133.

Prestwich S. (2003). "Hybrid local search on two multicolouring models". In *International Symposium on Mathematical Programming.* Copenhagen, Denmark.

Roberts F.S. (1991). "T-colorings of graphs: Recent results and open problems". *Discrete Mathematics*, 93(2-3), pp. 229–245.

Simon H.U. (1989). "Approximation algorithms for channel assignment in cellular radio networks". In *Fundamentals of Computation Theory*, vol. 380 of *Lecture Notes in Computer Science*, pp. 405–415. Springer Verlag, Berlin, Germany.

Sivarajan K.N., McEliece R.J., and Ketchum J.W. (1989). "Channel assignment in cellular radio". In *Proceedings of the 39th IEEE Vehicular Technology Conference*, pp. 846–850.

Smith D.H., Hurley S., and Allen S.M. (2000). "A new lower bound for the channel assignment problem". *IEEE Transactions on Vehicular Technology*, 49(4), pp. 1265–1272.

Stützle T. (1998). *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications.* Ph.D. thesis, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany.

Tesman B.A. (1990). "Set T-colorings". *Congressus Numerantium*, 77, pp. 229–242.

Tesman B.A. (1993). "List T-colorings". *Discrete Applied Mathematics*, 45(3), pp. 277–289.

Tsang E. and Voudouris C. (1998). "Solving the radio link frequency assignment problem using guided local search". In *NATO Symposium on Radio Length Frequency Assignment.* Aalborg, Denmark. Also available on line at `<http://cswww.essex.ac.uk/CSP/papers.html>`. (June 2005).

Voudouris C. (1997). *Guided Local Search for Combinatorial Optimization Problems.* Ph.D. thesis, University of Essex, Department of Computer Science, Colchester, UK.

A A Regression Model for Computation Time

For predicting the run time of OF-TS on the basis of instance features, we tried to derive from the measured data bounds which are a function of the 5 variables $|V|, \rho, \bar{r}, \bar{t}_v, \bar{t}_{uv}$. To do so, we used the methods discussed in McGeoch et al. (2002) of how to adapt known linear regression techniques to the study of the asymptotic behaviour of algorithms. By using two of the techniques they recommended, namely log-log and Box-Cox, we obtained the models reported in Table 5. On the Geometric graphs, the Box-Cox technique did not return an answer because of lack of convergence. On the Uniform graphs instead it returned a model which is exponential, although the curve has to be regarded as an upper bound in the asymptotic behaviour. However, an exponential model is not very useful. Considerations on the complexity of OF-TS (see Section 6.1) lead us to conjecture that an iteration of OF-TS should have complexity $\mathcal{O}(|V|^a \rho^b (\bar{r} + \bar{T}_{vv} + \bar{T}_{uv}))$, which is more similar to the model provided by the log-log transformation. In the light of this fact, we decide to rely on the polynomial model provided by log-log. Its characterisation together with the coefficients is given in the last row of the table. Note that, in the case of random graphs, the model is a lower bound, hence, its predictions are

	Uniform	Geometric
log-log	$\lvert V\rvert^{2.36}\rho^{0.99}\bar{r}^{3.68}(\bar{t}_{uu}+\bar{t}_{uv})^{2.54}$ **l**	$\lvert V\rvert^{1.30}\rho^{0.12}\bar{r}^{1.30}(\bar{t}_{uu}+\bar{t}_{uv})^{8.23}$ **c**
Box-cox	$1.02^{\lvert V\rvert}13.06^{\rho}2.20^{\bar{r}}1.70^{\bar{t}_{uv}}$ **u**	
Selected	$t=\frac{1}{3.91\cdot 10^{5}}\cdot(\lvert V\rvert^{2.36}\rho^{0.99}\bar{r}^{3.68}(\bar{t}_{uu}+\bar{t}_{uv})^{2.54})-79.56$	$t=\frac{1}{5.12\cdot 10^{11}}\cdot 1.02^{\lvert V\rvert}13.06^{\rho}2.20^{\bar{r}}1.70^{\bar{t}_{uv}}$

Table 5: The outcome of the linear regression analysis for determining the asymptotic time behaviour of OF-TS in relation with instance features. Conform to the notation introduced by McGeoch et al. (2002), we indicate with the letters l, c and u the type of bound. The letter l is used for lower bound, u for upper bound and c if the curve fits the data without a prevalent diverging trend.

actually under-estimates. We use this prediction of the computation time for performing I_{max} iterations, to determine the time limit on each instance of the experiments for the SLS algorithms.

B Numerical Results

Instance	Lwb	Best known	G-DSATUR min	G-DSATUR med	Max time	OF-SW min	OF-SW med	OF-SW+R min	OF-SW+R med	OF-TS min	OF-TS med	OF-TS+R min	OF-TS+R med	OF-TS+R* min	OF-TS+R* med
GEOM60	230	258	258	258	710	**258**	258	258	258	258	258	258	258	258	258
GEOM70	260	273	283	287.5	1060	**267**	267	268	269	269	270	270	271	271	272
GEOM80	365	383	392	395	1490	**382**	382	382	382	384	385	386	386	388	390
GEOM90	313	332	335	338.5	1810	334	334	333	333	**332**	333	332	332	335	337
GEOM100	378	404	412	416	2170	**404**	405	404	404	411	414	412	414	411	411
GEOM110	348	383	400	410	2510	378	378	**376**	377	383	383	381	382	387	389
GEOM120	343	402	412	419	2730	**397**	400	397	400	402	404	404	405	404	405
GEOM30a	182	209	238	238	380	**209**	211	210	211	212	213	212	212	212	212
GEOM40a	160	213	229	229	500	**214**	215	214	215	214	215	215	216	217	217
GEOM50a	199	318	335	345	1080	**315**	316	316	317	318	318	319	321	321	322
GEOM60a	290	358	369	373	1420	**356**	356	360	360	361	361	361	362	362	364
GEOM70a	425	469	487	487	1470	**478**	478	478	478	484	484	484	484	481	484
GEOM80a	241	379	388	396	1510	**360**	364	361	362	371	372	371	371	368	369
GEOM90a	285	377	398	405	1910	**377**	378	377	379	398	401	398	401	389	389
GEOM100a	302	459	462	471	2500	**437**	440	440	442	444	448	445	446	443	447
GEOM110a	385	494	523	523	3120	**490**	492	491	495	506	506	504	505	498	500
GEOM120a	514	556	571	578	3690	**549**	550	552	553	550	556	553	556	558	560
GEOM20b	39	44	45	45	30	**44**	44	44	44	44	44	44	44	44	44
GEOM30b	38	77	78	78	80	**77**	77	77	77	77	77	77	77	77	77
GEOM40b	74	74	79	86	140	**74**	74	74	74	74	74	74	74	74	74
GEOM50b	67	87	92	94	200	**84**	84	84	84	84	85	85	85	84	85
GEOM60b	79	116	123	132	300	117	119	**115**	118	120	120	118	119	119	119
GEOM70b	94	121	133	135	380	120	121	**119**	119	121	121	120	122	122	122
GEOM80b	110	141	148	149	490	**139**	139	139	139	140	140	139	140	141	142
GEOM90b	112	157	160	161	590	**147**	147	147	147	150	150	149	149	149	150
GEOM100b	133	170	173	179	690	**159**	160	161	161	162	163	164	164	162	165
GEOM110b	182	206	221	225.5	790	208	209	210	210	208	209	**206**	209	213	213
GEOM120b	172	199	206	219.5	910	**191**	196	193	197	195	198	197	198	201	201

Instance	DH − DSATUR min	DH − DSATUR avr	G-DSATUR min	G-DSATUR avr	DH-TS min	DH-TS avr	Iter. ($\times 10^7$)	Max time	OF-SW+R min	OF-SW+R avr	OF-TS+R min	OF-TS+R avr	OF-SW min	OF-SW avr	OF-TS min	OF-TS avr	Iter. ($\times 10^7$)
essai.100.275.10	63	63	61	61	46	46.5	1	10	47	47.3	47	52.3	47	47	**46**	47	0.28
essai.300.937.10	113	115.6	111	114.3	81	81.5	2	19	**82**	82.3	82	82.7	82	82.7	84	87.7	0.94
essai.500.1507.10	158	167	143	143.7	113	114	2	55	**111**	111.3	111	111.3	112	112.3	111	111.7	1.51
essai.1000.3049.10	282	286.7	224	228	179	179	3(?)	294	**182**	182.7	183	183.3	183	183	184	184	3.05
essai.30.95.50	75	77.1	68	68	62	62.5	1	10	**62**	62.3	62	62.3	62	62.3	63	63	0.10
essai.100.304.50	171	177.2	139	143.3	115	115.5	1	10	**116**	118.7	118	119.3	118	119.7	118	119	0.30
essai.300.905.50	432	444.9	330	331	268	269.5	2	78	**284**	285.7	284	286	284	287	284	287.7	0.91
essai.500.1484.50	678	678	479	479.7	403	403	2	253	**418**	422	420	422	423	425.7	425	426.3	1.48
essai.1000.3024.50	1221	1227.3	869	874.3	847	847	3(?)	1459	796	799.3	**792**	795.3	808	809	803	811	3.02
essai.30.90.90	146	154.7	119	120	103	103.33	1	10	**104**	104.3	104	104	104	104.7	105	105.3	0.09
essai.100.299.90	321	341.3	259	260	225	225.33	1	10	**234**	235.3	234	234.7	235	237	236	236.7	0.30
essai.300.940.90	903	922.2	661	671.3	573	574.33	2	168	**601**	606.7	607	608.7	610	611.3	608	610	0.94
essai.500.1536.90	1410	1433.3	1020	1026	932	932	2	523	**940**	950.3	950	955	955	958	950	957.7	1.54
essai.1000.2975.90	2523	2542	1804	1819	1724	1724	3(?)	2395	**1747**	1753.3	1748	1753.3	1754	1756.3	1760	1762	2.97

Table 6: Numerical results on the GEOM instance set (above) and HD-RU instance set (below). Results are collected over 3 runs per algorithm–instance combination; the results were visualised in Figures 6 and 7, respectively. Results are expressed in terms of the number of colours, k, and the span may be derived by subtracting one. For each algorithm, we report the best and the median or average result. The time limit at disposal for producing a solution in a single run is expressed in seconds and the lower bound is determined through the TSP procedure described in Aardal et al. (2001). In the instances of the set HD-RU, DH − DSATUR and DH-TS are the DSATUR and Tabu Search algorithm proposed by Dorne and Hao (1998). We report also the number of maximal iterations that DH-TS was allowed for solving each single decision problem. This values are to be compared with the number of maximal total iterations used by our OF-TS algorithm.

Instance	Lwb		G-DSATUR		Max	OF-TS+R		OF-SW		OF-SW+R		OF-TS		OF-TS+R*	
	min	max	min	max	time	min	max	min	max	min	max	min	max	min	max
TG − 60 − 0.1 − 5.5	30	41	41	62	10	36	46	36	46	36	46	36	46	36	46
TG − 60 − 0.1 − 5.10	43	69	78	107	20	63	101	63	83	63	83	64	101	63	84
TG − 60 − 0.1 − 10.5	56	92	79	116	710	70	97	70	93	69	92	69	98	71	92
TG − 60 − 0.5 − 5.5	31	51	89	116	320	74	88	75	90	76	90	75	89	75	91
TG − 60 − 0.5 − 5.10	62	106	168	231	3440	135	168	138	173	138	172	139	170	140	177
TG − 60 − 0.5 − 10.5	67	101	148	222	6820	121	170	123	162	123	163	122	164	123	165
TG − 60 − 0.9 − 5.5	72	97	160	194	1250	137	164	138	169	139	167	138	165	137	172
TG − 60 − 0.9 − 5.10	92	126	304	370	6820	259	290	260	297	263	296	262	295	264	300
TG − 60 − 0.9 − 10.5	142	174	288	353	12880	252	299	251	302	254	304	252	301	249	305

Instance	Lwb	OPT	Known	G-DSATUR		Max	OF-TS+R*		OF-TS+R		OF-TS		OF-SW		FCNS	
			heur.	min	med	time	min	med	min	med	min	med	min	med	min	med
P1	371	427	448	480	485	490	**427**	427	427	427	427	427	427	428	427	427
P2	428	427	476	458	464	712	**427**	427	427	448	428	459	427	427	427	427
P3	258	258	285	268	268	333	**258**	258	258	258	258	258	258	262	259	260
P4	254	253	269	260	266	225	**253**	254	253	253	253	253	254	257	255	255
P5	240	240	251	250	255	327	**240**	240	240	240	240	240	240	240	241	241
P6	179	180	231	195	199	279	183	184	185	185	185	186	**182**	183	193	194
P7	743	856	895	969	973	2439	**856**	856	871	877	860	871	874	877	857	859
P8	461	525	593	539	539	610	**525**	525	525	525	527	528	525	527	534	534
P9	1487	1714	1801	1938	1946	10381	**1715**	1719	1756	1758	1755	1759	1770	1775	1747	1751

Table 7: Numerical results on the NRU instance set (above) and Philadelphia instance set (below). On the NRU instances results corresponding to 3 runs per algorithm on the 10 instances of each class are aggregated and the range is reported. On the Philadelphia instances, the best and median results are derived from 5 runs per algorithm–instance combination. These data constitute the numerical expression of the visual representation of Figure 8 and 9, respectively. Results are expressed in terms of k and the time limit in seconds. The lower bound is determined through the TSP procedure described in Aardal et al. (2001). Note that this lower bound requires an adjustment as the Hamiltonian path derived from the solution of the TSP is not always the shortest.