Exact Colour Reassignment in Tabu Search for the Graph Set T-Colouring Problem

Marco Chiarandini[1], Thomas Stützle[2] and Kim Skak Larsen[1]

[1] University of Southern Denmark, IMADA, Odense, Denmark
marco,kslarsen@imada.sdu.dk
[2] Université Libre de Bruxelles, CoDe, IRIDIA, Brussels, Belgium
stuetzle@ulb.ac.be

Abstract. The graph set T-colouring problem (GSTCP) is a generalisation of the classical graph colouring problem and it is used to model, among other problems, the assignment of frequencies in mobile networks. The GSTCP 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 various objective functions of interest, here we focus on the minimisation of the difference between the largest and the smallest integers used, the span.
The large size of the instances arising in practical applications makes the use of heuristic algorithms necessary. In this article, we propose two hybrid algorithms that combine a basic tabu search algorithm for this problem with an exact algorithm for the re-assignment of the colours to a vertex. We compare these two algorithms to the basic tabu search algorithm and tabu search algorithms working on a transformation of the GSTCP into a T-colouring problem. The experimental comparison of these algorithms, which is based on a rigorous statistical analysis, establishes that the two hybrids perform better on a variety of instance classes for the graph set T-colouring problem.

1 Introduction

A *set T-colouring* of a graph is an assignment of sets of nonnegative integers (colours) to the vertices of the graph such that (i) every vertex receives exactly the number of colours it requires, (ii) each pair of integers assigned to a single vertex satisfies the vertex separation constraints, and (iii) every pair of numbers assigned to two adjacent vertices satisfies the edge separation constraints. The edge and vertex separation constraints are expressed by sets of integers that represent the forbidden differences between the integers assigned to the vertices. In the special case of the sets being composed of only consecutive integers, we speak of *separation distance* constraints and the largest integer in each set suffices to represent them.

There exist various objective functions for the optimisation version of the *Graph Set T-Colouring Problem* (GSTCP). Common objective functions to be minimised are the *span*, that is, the difference between the largest and the smallest integers assigned to any vertex or the *order*, that is, the number of integers [1, 2] effectively used. In frequency assignment, more often, it is required to minimise the constraints violations by keeping the number of colour fixed. However, in the literature on this problem the focus has been mainly on the minimisation of the span since most solution approaches tend to solve this problem by minimising constraint violations. Here we also address the minimal span objective.

The GSTCP is a generalisation of the vertex colouring problem and it arises in the modelling of various real-life problems, the most important being the assignment of frequencies to radio transmitters when designing mobile phone networks. In this case, vertices represent transmitters and colours the frequencies to be assigned to the transmitters subject to certain interference constraints, the T-constraints [1]. Other applications arise in traffic phasing and fleet maintenance [3] 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 [2].

Due to its practical interest, the GSTCP has received significant attention both in graph theory [2–6] as well for its algorithmic solution. The development of solution algorithms has mainly focused on approximate methods, mainly because exact methods cannot efficiently solve large size instances as those arising in frequency assignment [7, 8]. Among the approximate algorithms, the overwhelming part of the literature focuses on stochastic local search (SLS) [9] algorithms. Among the first of these approaches, Dorne and Hao apply a tabu search algorithm for the GSTCP [10]. Further research on the GSTCP has been inspired by the *Computational Challenge on Graph Colouring and its Generalisations* organised by Johnson, Mehrotra and Trick (see `http://mat.gsia.cmu.edu/COLORING02/`). [3] Phan and Skiena build a metaheuristic algorithm based on swap operations and simulated annealing using their general-purpose platform *Discropt* [11], while Prestwich proposes a randomised backtracking algorithm [12]. In later research, Lim, Zhang and Zhu designed a squeaky wheel algorithm for this problem [13].

In this article, we present a new tabu search algorithm for the GSTCP. This algorithm is very similar to that of Dorne and Hao, but it uses an exact algorithm for the re-assignment of the colours at a vertex. A further contribution of this article is a comprehensive experimental comparison of the new and known algorithms on GSTCP instances with separation distance constraints. This study uses a rigorous statistical analysis and it shows that our new tabu search search algorithms perform better than previous known versions on specific GSTCP instance classes.

The paper is organised as follows. Section 2 introduces some formalism, transformations of the problem and the benchmark instances. Section 3 describes the SLS algorithms that are experimentally compared in Section 4. We end with some concluding remarks in Section 6.

[3] Note that in this challenge, the separation distance GSTCP is called bandwidth multi-colouring problem.

2 Definitions, Transformations and Benchmark Instances

A GSTCP instance is defined by (i) an undirected graph $G = (V, E)$, with V being the set of $n = |V|$ vertices and E being the set of edges, (ii) a set of integer numbers (called colours) Γ, (iii) a number $r(v)$ of required colours at each vertex $v \in V$, and (iv) a collection of sets 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 $u \in V$ of disallowed separations of colours between vertices and within vertices. The task in the decision version of the GSTCP is to find a multi-valued function $\varphi : V \mapsto \Gamma$ such that that the three following groups of constraints

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

$$|\varphi(u,i) - \varphi(u,j)| \notin T_u \qquad \forall \varphi(u,i), \varphi(u,j) \in \varphi(u), i \neq j \tag{2}$$

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

are satisfied. We call such a multi-valued function φ a *proper set T-colouring*, if all these constraints are satisfied and *improper*, otherwise. The three groups of constraints to be satisfied are called *requirement constraints*, *vertex constraints*, and *edge constraints*, respectively. Various objective functions can be defined for the GSTCP. Here, we consider the span, that is, $\max_{u,v,i,j} |\varphi(v,i) - \varphi(u,j)|$, the maximal difference between the colours used. With $\min_{u,j}\{\varphi(u,j)\}$ fixed to 1, the span corresponds to $k - 1$ if $\Gamma = \{1, 2, \ldots, k\}$. In the optimisation version, we are searching for the minimal span, that is, for the minimal value of k such that a proper set T-colouring exists.

Finding a proper set T-colouring for a graph G and a T-set T is known as the *graph set T-colouring problem* (GSTCP). The case $r(v) = 1, \forall v \in V$, is a special case of it which is called *graph T-colouring problem.* In the T-colouring problem there are no particular requirement and vertex constraints. Both, graph set T-colouring and T-colouring problems, are $\mathcal{NP}$–complete because they are generalisation of the *k-colouring problem.*

An instance of the GSTCP problem, consisting of a graph G, a T-set T and vertex requirements $r(v), \forall v \in V$, is equivalent to an instance of the T-colouring problem 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 $v \in V(G)$ so that at the end $|V^S| = \sum_{v \in V} r(v)$. 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_{wv}. The graph G^S is called *split graph* and there is a bijective correspondence between a solution on G and on G^S.

Our benchmark set comprises two classes of randomly generated instances. The first comprises the instances introduced by Michael Trick for the "Computational Challenge on graph colouring and its generalisations," the second class comprises random uniform graphs. Such graphs were also used in [10] for testing a tabu search algorithm. We generated an additional large number of relatively small such instances, since the instances proposed by Dorne and Hao result in very high computation times due to their size, which would limit strongly the number of trials run. A final instance set stems from a well known frequency assignment instance. All benchmark instances we consider are *separation distance* instances, that is, the sets of disallowed distances are composed only of consecutive integers and represented by a single integer, t_u or t_{uv}. The instance classes are described next.

Random Geometric instances (DIMACS). The graphs are formed by vertices that correspond to points in a $[10,000 \times 10,000]$ grid with randomly generated coordinates. Each vertex is connected by an edge to another one, if the points are close enough. Distances associated to edges are inversely proportional to the distances between nodes in the grid. Requirements and vertex distances are assigned to vertices by uniformly choosing in the sets $\{1,\ldots,r\}$ and $\{1,\ldots,t\}$, respectively. The size of these instances ranges from 20 to 120 vertices. We denote sparse instances as GEOMn and denser instances as GEOMna and GEOMnb. The instances GEOMnb have higher requirements per node than GEOMna. Statistics of this class of instances are summarised in Table 1a.

Random Uniform instance. Uniform graphs are typically identified as G_{np} where n is the number of vertices and each of the $\binom{n}{2}$ possible edges is present with a probability p. These instances were generated by the algorithm of [14], which was modified to produce set T-colouring instances. The requirement, vertex and edge constraints are constructed as in the previous class, with $t = t_u = t_{uv}$, $\forall\, u, v \in V$. The values assigned to the parameters and the number of instances are reported in Table 1a. Note that the instances in [10] were generated in an analogous way.

Philadelphia instances. These instances are characterised by 21 hexagons representing the cells of a cellular phone network around Philadelphia [15]. For each cell, a demand $r(v)$ is given. In case 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 reuse distance d by a series of non-increasing values $d^0, \ldots, d^k$. For more on these instances we refer to the FAP web repository.[4] In conformity with the frequency assignment literature we denote these instances by P1-P9.

3 SLS Algorithms for the GSTCP

In this section, we describe the main components of the SLS algorithms we examine: the construction heuristic for generating the initial solution, the local search approaches to the GSTCP and the high-level description of the SLS method (that is, the metaheuristic part of the SLS algorithm).

A design choice of our overall SLS algorithms, which are all based on Tabu Search, is that they solve directly the optimisation version of the GSTCP. This is done by starting with some large value of k and successively trying to reduce

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

$\|V\|$	$\bar{\rho}$	r	t_u	$\overline{t_{uv}}$	# instances
$20, 30, \ldots, 120$	0.1	3	10	4.5	–
		10	10	4.5	11 (`GEOM`n)
	0.2	3	10	4.5	11 (`GEOM`n`a`)
		10	10	4.5	11 (`GEOM`n`b`)

(a) Random Geometric instances

$\|V\|$	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

(b) 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]

(c) FAP instances

Table 1: Statistics on the benchmark instances. t_u and t_{uv}, respectively are used to indicate that the range of values differs among vertex and edge constraints.

the number of colours used, which directly minimises also the span. The best solution returned is the minimum value of k for which a proper set T-colouring is found.

3.1 Construction Heuristic

We use a *generalised DSATUR* heuristic for constructing the initial solution. This heuristic resulted to be the best one in a study reported in [16]. This heuristic works on the split graph and for each colour assignment, first a next vertex is chosen and next this vertex is assigned a colour. Generalised DSATUR chooses the vertex to colour next based on the *saturation degree*, *i.e.*, the number of different colours forbidden by the assignment of colours in the adjacent vertices. For every vertex $u \in G^S$ receiving a colour c, the list of forbidden colours of the vertices v adjacent to u is updated with the colours in the interval $\{c - t_{uv}, c + t_{uv}\}$. The order of vertices is recomputed by assigning higher priority to vertices with higher saturation degree. In cases of ties, priority is given to vertices with the largest adjusted vertex degree, which is defined as $d(v) = \sum_{u \in V^S, uv \in E^S} t_{uv}$.

The application of the DSATUR heuristic generates a proper set T-colouring and, hence, an upper bound k which is the maximal colour used.

3.2 Local search approaches

For applying local search to the GSTCP, we may distinguish between two fundamentally different approaches. The first tackles the problem as a series of decision problems, where for each current value k of available colours it is tried to find a proper set T-colouring. (In this first case, we further distinguish the approaches based on the problem representation in what follows.) The second leaves the value of k variable. These approaches form the basis of the tabu search algorithms that are explained later.

Approach 1: split graph, k fixed. In this case, solutions are represented as complete assignments, *i.e.*, one colour is assigned to each vertex. The advantage of this solution representation is that the requirement constraints of the GSTCP are always satisfied. The evaluation function f is defined as the number of vertex and edge constraints broken and, hence, the goal becomes to minimise f to zero. The neighbourhood in this approach is defined through the well-known one-exchange neighbourhood, where the solutions s and s' are neighboured if they differ in the colour assignment of one single vertex. Often, it is useful to restrict the neighbourhood examination of the one-exchange to vertices that are involved in constraint violations. We call this restricted neighbourhood N_E in the following.

Approach 2: original graph, k fixed. A variant to the previous approach uses the original graph representation. The main implication of this choice is on the representation of a solution, which is now given naturally by a set of $r(v)$ colours for each vertex $v \in V$. Contingently, the vertex constraints may be used to reduce the effective search space to only those candidate assignments that satisfy them. Hence, requirement constraints and vertex constraints are always satisfied and the evaluation function needs only to count the number of unsatisfied edge constraints.

The one-exchange neighbourhood can here be restricted to N'_E, that is, the collection of colour changes at a vertex that maintain the vertex constraints satisfied. This operator is similar to N_E with an additional restriction on the set of new colours. In addition, the vertex exact colour reassignment N_R is defined, which entails the reassignment of all the colours of one single vertex. We require that the reassignment must satisfy the requirement, vertex and edge constraints acting on that vertex. As such, it can be seen as an exact solution to a subproblem, where the assignment of $\{1, \ldots, k\}$ colours to one vertex is searched under the condition that none of the other vertices changes its colour assignment.

Approach 3: split graph, k variable. In this approach, 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. Here, solutions can again be represented as complete colourings as in Approach 1. The difference is that solutions can be proper and improper set T-colourings and that the number of colours is bound by k_I which remains the same throughout the whole search. An evaluation function to guide the search toward proper colourings and toward colourings with smaller span was proposed by [17]. It is defined as

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

where k_{max} is the maximal colour, $I_e(u,v)$ and $I_v(v)$ are indicator functions that are one if the corresponding edge and vertex constraints are broken, $k_{max} - k_{min}$ is the span of the colouring, and $I_g(c)$ is an indicator function to determine whether any vertex has colour i; this last term computes the order. The edge 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, as in Approach 1, the same one-exchange neighbourhood may be used.

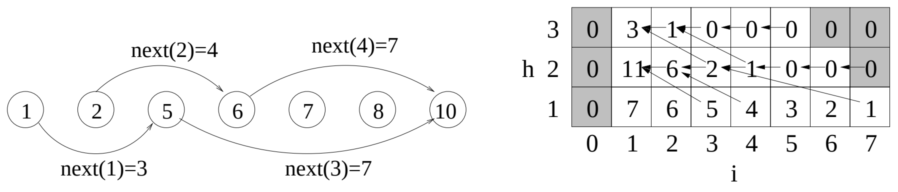
3.3 Exact colour reassignment neighbourhood

The effect of this neighbourhood operator is to modify the colour assignment to one vertex such that the requirement, vertex and edge constraints are all satisfied. Once a vertex is chosen, a set F is determined comprising 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 $\mathcal{O}(|V|k)$ if the usual speed-up techniques known from the graph colouring problem are implemented. If we simply looked for $r(v)$ colours from F such that the vertex constraints are satisfied this would be easy: we just have to order the values in F and scan the set once, skipping the values that are not sufficiently distant from the previous ones. Yet, this procedure is deterministic and we would obtain the same reassignment of colours visiting a vertex twice if nothing in the adjacent vertices changed. We want to avoid this and make the search randomised, such that, visiting the vertex the second time, we obtain a different configuration that can be profitably propagated. Implementing this strategy corresponds to determine all subsets of F of size $r(v)$ that are proper, *i.e.*, that satisfy the vertex constraints, and pick one at random.[5]

We solve the problem of determining all the proper subsets of F in a dynamic-programming fashion by solving subproblems and saving their answers in a table. 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. 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 randomly by selecting among all the existing solutions.

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

[5] Formally, the problem can be stated as:
Subsequence of length L of K integers with mutual distance not smaller than D. Given an arbitrary sequence of integers $s = \{s_1, \ldots, s_K\}$, a subsequence $l = \{l_1 \ldots l_L\}$ of length L with $l_i \in s$, $\forall\ i = 1, \ldots, L$ and mutual distance not smaller than D is a subsequence satisfying: $|l_i - l_j| \geq D$, $\forall ij$. The goal is to find one such subsequence. *Enumerating all subsequences of length L of K integers with mutual distance not smaller than D.* Given an arbitrary sequence of integers $S = \{s_1, \ldots, s_K\}$ find all subsequence of length L of S with mutual distance not smaller than D.



h \ i	0	1	2	3	4	5	6	7
3	0	3	1	0	0	0	0	0
2	0	11	6	2	1	0	0	0
1	0	7	6	5	4	3	2	1

Fig. 1: An example of reassignment interchange for a case with $F = \{1, 2, 5, 6, 7, 8, 10\}$, $L = |F| = 7$, $D = t_v = 4$ and $K = r(v) = 3$. 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 which stored values are used for computing the entries. The grey cells indicate the values without a proper meaning and hence assigned by convention.

$next(i) = \min_j\{j | j > i,\ s[j] - s[i] >= D\}$. For each subsequence l of s, the number of proper subsequences of s of length $h \in \{1, \ldots, H\}$ containing l, can be determined by the recursion

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

with $N_h[0] = 0$ 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 $N_{h-1}[next(i)]/N_h[i]$. 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 1, 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 table can then be computed in $\mathcal{O}(\max(D, H, \log L)L)$. The exact colour reassignment of a vertex requires then only one more scan of the sequence s.

3.4 Tabu search algorithms

As said, all the SLS algorithms we applied here rely on tabu search, a technique that has shown to work particularly well for graph colouring problems. We give details on all the five algorithms that we tested in our experimental comparison.

Approach 1: split graph, *k* fixed, complete colourings We use a standard tabu search algorithm that chooses at each iteration a best non-tabu move or a tabu but "aspired" neighbouring solution from the restricted one-exchange neighbourhood (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 that of the successful tabu search algorithms for the graph colouring problem [18]. We denote this algorithm SF-TS.

SF-TS is very similar to the Tabu Search algorithms proposed in [19], [20], and [17]. In those papers, Tabu Search was shown to perform better than simulated annealing and other genetic algorithms. We tested the FASoft system [17] on the random geometric graphs and results were worse than those of SF-TS. The reason for this may also be due to the worse solution quality reached by the construction heuristics in FASoft when compared to that obtainable by our generalised DSATUR.

Approach 2: original graph, *k* fixed, complete colourings An application of Tabu Search on the one exchange neighbourhood N'_E on the original graph was already designed by [10]. Other versions of this algorithm for frequency assignment [21] differ only in the management of the tabu length or are more rudimentary [22]. Our version uses the same tabu tenure definition as SF-TS and is in this equal to the version by [10]. We denote this algorithm OF-TS.

We also include two enhanced versions of OF-TS which make use of the newly introduced 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 neighbourhood N_R, restricted to only 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 after no move is found in N_R of 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*.[6]

The tabu search mechanism applied to moves in N'_E is the same used in N_E and [10] (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 is preferable.

Approach 3: split graph, *k* variable, complete colourings We test a tabu search algorithm on N''_E based on the same framework as in the case of k fixed and neighbourhood N_E, but using the evaluation function of Equation 4. We denote this algorithm as SV-TS.

[6] 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 can be avoided in another implementation.

Parameter settings The tabu search algorithms require a number of parameters to be adapted to the class of problem instances under consideration. To accomplish this task we used the racing algorithm of Birattari [23], which is a fully automatic procedure based on sequential testing. In the tabu search algorithms designed the only parameter to be decided is δ. For each of the algorithms (except OF-TS+R) we used as candidates the set of numbers $\{0.5; 1; 10; 20; 30; 40; 50; 60; 70; 100\}$ and the best values found were 10 for the uniform, 30 for random, and 40 for Philadelphia instances. In the following for each algorithm we only consider the version with the best set of parameters for the instance class.

4 Experimental Analysis

We evaluate experimentally the five versions of tabu search. We maintain the classes of instances separated as we expect to see differences in performance. Unfortunately, given the larger among of computational effort for solving even small GSTCP instances, we could not use the large instances of Dorne and Hao; we used them, however, to check that our re-implementation of OF-TS gives results comparable to those earlier published [16].

To compare the algorithms under roughly fair conditions, we imposed a same computation time limit for them. This is necessary, since the single iterations of OF-TS and OF-TS+R have different computation time requirements. To determine a time limit we run OF-TS for $I_{max} = 10^5 \times \sum_{v \in V} r(v)$ iterations. Being the only algorithm previously published the choice of OF-TS as reference algorithm is justified. Note that the time limit varies among instances, moreover, it is a stochastic variable. We used, therefore, a multiple regression model from the results of 5 runs per instance to define the time limits. We omit here the details of this procedure and in the tables given next we report the time limits predicted by the model which refer to a machine 2GHz AMD Athlon MP 2400 Processor with 256 KB cache and 1 GB RAM, running Debian Linux.

On the 27 random geometric instances (disconnected graphs were removed) and the 90 random uniform instances, we collected 3 runs per algorithm per instance. On the 9 Philadelphia instances we collected instead 5 runs per algorithm per instance.

4.1 Tabu search comparison

The analysis of results is carried out through simultaneous confidence intervals for multiple comparisons. In particular the Friedman rank-based statistical procedure is used to infer the simultaneous confidence intervals of the average rank performance of each algorithm [24, 25]. In this procedure, each result in terms of colour span is ranked with all other results in the same instance, thus removing the problem of different scale of results among the instances and allowing an aggregate analysis (within the class).

We report the results in Figure 2, 3 and 4 where two algorithms are significantly different if the two confidence intervals of the corresponding average rank do not overlap. The more the interval is shifted towards the left the better are the algorithm results.

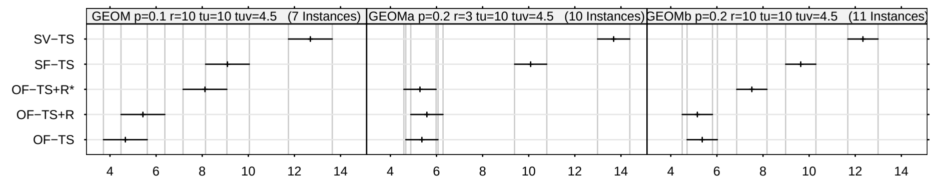


Fig. 2: Multiple comparisons through simultaneous confidence intervals on the random geometric instances. The x-axis indicates the average rank while the confidence intervals are derived from the Friedman test..

The first observation is that, as expected, results vary among the instances. On the geometric graphs the exact colour reassignment does not introduce any significant improvement and, in fact, seems to worse the basic OF-TS. Note, however, that the random geometric graphs are representative of only a restricted portion of the space of all possible instances: they represent only graphs with very small edge density and fixed separation constraints.

More informative in this sense is the analysis on the random uniform graphs (Figure 3). Here the space of instances is better sampled as graphs with different edge density and vertex requirement are also present. This allows to conjecture on performance variations due to instance features. The most important result is that exact colour reassignment becomes worth when the edge density is at least 0.5. In these cases, indeed, the performance of OF-TS+R becomes clearly better than that of OF-TS. The vertex requirements appear also to have an influence which is reflected by the better performance of OF-TS+R* when vertex requirements are high. This latter tendency seems to be confirmed on the Philadelphia instances where OF-TS+R* outperforms all other versions of tabu search.

Finally a note on SV-TS and SF-TS. Given the much worse performance of these two algorithms compared to the other three, we conclude that the use of the split graph is not a promising approach for local search algorithms for the GSTCP. (This seems to be different for constructive heuristics [16].) Additionally, the worse performance of SV-TS compared to SF-TS suggests that the usage of a variable k is not advisable for the GSTCP.

4.2 Comparison with state-of-art

In tables 2 and 3 we report the numerical results for comparison with previously published results. On the random geometric graphs previous results on the span are due to [11–13]. On the Philadelphia instances results on lower bounds and upper bounds are available at `http://fap.zib.de/problems/Philadelphia/`. In all the instances the best upper bounds of approximate algorithms went over time to coincide with the lower bound thus proving optimality for these instances. Note however that no algorithm has solved all the instances alone and we refer to the FAP web repository for a complete list of references for each

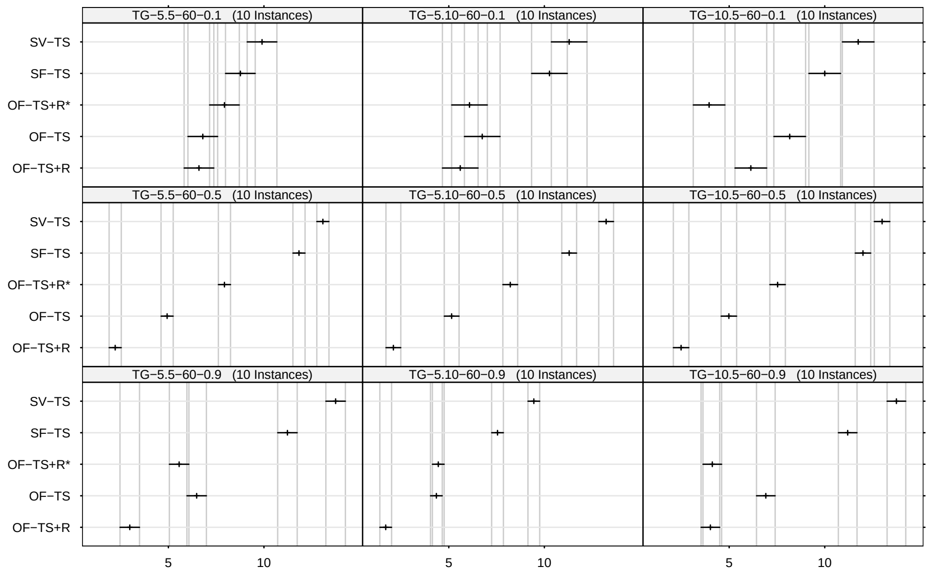


Fig. 3: Confidence intervals for the all pairwise comparisons of SLS algorithms on aggregated uniform random instances of the GSTCP. The x-axis indicates the average rank while the confidence intervals are derived from the Friedman test.

result (the most robust seems to be the genetic algorithm by [26]). The same reasoning holds for the construction heuristics whose results we also report for comparison. On both instance classes the results that we report are the lower bounds produced by the procedure described in [16], and the best and median value for the generalised DSATUR heuristic and the tabu search algorithms. Results are in terms of maximal number of colours used, hence to derive the span one has to subtract one.

On the random geometric graphs our results improve the best known colourings on 11 instances, are worse on 9 instances and reach the same performance on 8 instances. There seems to be therefore no significant difference. However, such kind of comparison is not reliable as our data are based on aggregated best values from more than one algorithm and more than one run. Nevertheless, the results prove the high quality of the results here discussed.

On the Philadelphia instances OF-TS+R* produces 7 times out of 9 the optimal solution. Only the algorithm by [26] can reach similar performance with 8 out of 9 optimal results. As far as the results of our generalised DSATUR construction heuristic its results are worse than those attained by a portfolio of heuristics implemented in the system FASoft [17]. However a comparison with the results of [10] shows that our generalised DSATUR is much better. No

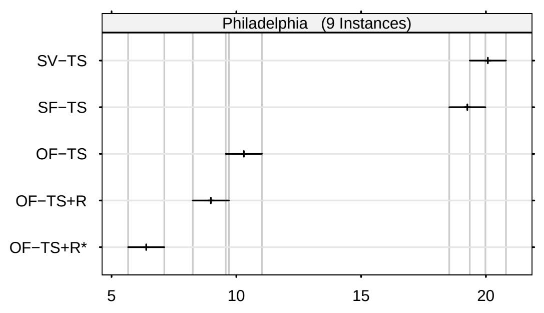


Fig. 4: Confidence intervals for the all pairwise comparisons of SLS algorithms on Philadelphia instances of the GSTCP. The x-axis indicates the average rank while the confidence intervals are derived from the Friedman test.

result for the construction heuristic is instead reported on the random geometric instances.

5 Summary

In this article, we studied a hybrid tabu search algorithm for the GSTCP problem. This algorithm uses a canonical tabu search algorithm based on a one-exchange neighbourhood operator and it enhances the canonical tabu search by an exact reassignment of colours to vertices when opportune. The exact reassignment would be trivial if implemented in a deterministic manner. We proposed instead an algorithm which is capable to return a random exact colour reassignment. This feature favours the diversification of the search, which is often a key mechanism for making an SLS algorithm successful. A major contribution of this article is, hence, the recursive formula with polynomial time-complexity for this random reassignment.

Another contribution is the experimental analysis of various tabu search algorithms on 3 classes of instances. The results of this comparisons are that (i) tabu search algorithms working on a split-graph representation are less efficient than the tabu search algorithms working on the “natural” problem representation, (ii) on instances characterised by a low edge density in the graph and low vertex requirements, the occasional exact reassignment does not improve the underlying tabu search algorithms, (iii) on the other instances and above all on the Philadelphia instances, which stem from the literature on frequency assignment, the hybrid algorithm performs, often by quite a large margin, better than the basic tabu search algorithms.

Acknowledgements. Thomas Stützle acknowledges support of the Belgian FNRS, of which he is a research associate. Kim S. Larsen was supported in part by the Danish Natural Science Research Council (SNF).

Instance	LWB	Best	DSATUR		max time	OF-TS		OF-TS+R		OF-TS+R*		SF-TS		SV-TS	
GEOM60	230	258	258	258	710	**258**	258	258	258	258	258	258	258	258	258
GEOM70	260	273	283	287.5	1060	**269**	270	270	271	271	272	270	278	274	285
GEOM80	365	383	392	395	1490	**384**	385	386	386	388	390	389	389	393	394
GEOM90	313	332	335	338.5	1810	**332**	333	332	332	335	337	333	334	335	340
GEOM100	378	404	412	416	2170	**411**	414	412	414	411	411	411	414	414	415
GEOM110	348	383	400	410	2510	383	383	**381**	382	387	389	381	389	403	404
GEOM120	343	402	412	419	2730	**402**	404	404	405	404	405	409	417	416	419
GEOM30a	182	209	238	238	380	**212**	213	212	212	212	212	222	228	234	235
GEOM40a	160	213	229	229	500	**214**	215	215	216	217	217	220	220	225	226
GEOM50a	199	318	335	345	1080	**318**	318	319	321	321	322	329	337	337	340
GEOM60a	290	358	369	373	1420	**361**	361	361	362	362	364	363	363	373	373
GEOM70a	425	469	487	487	1470	484	484	484	484	**481**	484	483	486	487	487
GEOM80a	241	379	388	396	1510	371	372	371	371	**368**	369	378	383	391	394
GEOM90a	285	377	398	405	1910	398	401	398	401	**389**	389	398	402	398	405
GEOM100a	302	459	462	471	2500	444	448	445	446	**443**	447	454	459	462	465
GEOM110a	385	494	523	523	3120	506	506	504	505	**498**	500	507	509	516	518
GEOM120a	514	556	571	578	3690	**550**	556	553	556	558	560	556	558	578	578
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	75	75	76	76
GEOM50b	67	87	92	94	200	**84**	85	85	85	84	85	86	87	87	88
GEOM60b	79	116	123	132	300	120	120	**118**	119	119	119	120	120	121	122
GEOM70b	94	121	133	135	380	121	121	**120**	122	122	122	122	123	125	125
GEOM80b	110	141	148	149	490	140	140	**139**	140	141	142	140	141	140	142
GEOM90b	112	157	160	161	590	150	150	**149**	149	149	150	152	153	153	155
GEOM100b	133	170	173	179	690	**162**	163	164	164	162	165	169	170	172	172
GEOM110b	182	206	221	225.5	790	208	209	**206**	209	213	213	212	213	214	215
GEOM120b	172	199	206	219.5	910	**195**	198	197	198	201	201	200	201	202	203

Table 2: Numerical results on the random geometric instances. Given are the instance identifier, a lower bound, the best solution known so far, the results of our generalised DSATUR, our computation time limits, and the results of our five tabu search algorithms.

Instance	OPT	LWB	Known Heur.	DSATUR		max time	OF-TS+R*		OF-TS+R		OF-TS		SF-TS		SV-TS	
P1	427	371	448	480	485	490	**427**	427	427	427	427	427	479	480	480	480
P2	427	428	476	458	464	712	**427**	427	427	448	428	459	482	483	484	484
P3	258	258	285	268	268	333	**258**	258	258	258	258	258	266	266	262	263
P4	253	254	269	260	266	225	**253**	254	253	253	253	253	264	264	264	264
P5	240	240	251	250	255	327	**240**	240	240	240	240	240	240	240	240	240
P6	180	179	231	195	199	279	**183**	184	185	185	185	186	195	197	195	197
P7	856	743	895	969	973	2439	**856**	857	871	877	860	871	967	969	967	969
P8	525	461	593	539	539	610	**525**	525	525	525	527	528	548	548	549	549
P9	1714	1487	1801	1938	1946	10381	**1715**	1717	1756	1758	1755	1759	1938	1938	1938	1938

Table 3: Numerical results on the Philadelphia instances. Given are the instance identifier, the known optimum, a lower bound, the best heuristic solution of the FASoft system, the results of our generalised DSATUR, our computation time limits, and the results of our five tabu search algorithms.

References

1. Hale, W.K.: Frequency assignment: Theory and applications. Proceedings of the IEEE **68** (1980) 1497–1514
2. Tesman, B.A.: Set T-colorings. Congressus Numerantium **77** (1990) 229–242
3. Roberts, F.S.: T-colorings of graphs: Recent results and open problems. Discrete Mathematics **93** (1991) 229–245
4. Tesman, B.A.: List T-colorings. Discrete Applied Mathematics **45** (1993) 277–289
5. Giaro, K., Janczewski, R., Malafiejski, M.: The complexity of the T-coloring problem for graphs with small degree. Discrete Applied Mathematics **129** (2003) 361–369
6. Giaro, K., Janczewski, R., Malafiejski, M.: A polynomial algorithm for finding T-span of generalized cacti. Discrete Applied Mathematics **129** (2003) 371–382
7. Eisenblätter, A., Grötschel, M., Koster, A.M.C.A.: Frequency assignment and ramifications of coloring. Discussiones Mathematicae Graph Theory **22** (2002) 51–88

8. Aardal, K.I., van Hoesel, C.P.M., Koster, A.M.C.A., Mannino, C., Sassano, A.: Models and solution techniques for the frequency assignment problem. ZIB-report 01–40, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany (2001)
9. Hoos, H., Stützle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann Publishers, San Francisco, CA, USA (2004)
10. Dorne, R., Hao, J.: Tabu search for graph coloring, T-colorings and set T-colorings. In: Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers (1998)
11. Phan, V., Skiena, S.: Coloring graphs with a general heuristic search engine. In Johnson, D.S., Mehrotra, A., Trick, M., eds.: Proceedings of the Computational Symposium on Graph Coloring and its Generalizations, Ithaca, New York, USA (2002) 92–99
12. Prestwich, S.: Hybrid local search on two multicolouring models. In: International Symposium on Mathematical Programming, Copenhagen, Denmark (2003)
13. Lim, A., Zhu, Y., Lou, Q., Rodrigues, B.: Heuristic methods for graph coloring problems. In: SAC '05: Proceedings of the 2005 ACM Symposium on Applied Computing, New York, NY, USA, ACM Press (2005) 933–939
14. Culberson, J., Beacham, A., Papp, D.: Hiding our colors. In: Proceedings of the CP'95 Workshop on Studying and Solving Really Hard Problems, Cassis, France (1995) 31–42
15. Anderson, L.G.: A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system. IEEE Transactions on Communications **21** (1973) 1294–1301
16. Chiarandini, M.: Stochastic Local Search Methods for Highly Constrained Combinatorial Optimisation Problems. PhD thesis, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany (2005)
17. Hurley, S., Smith, D.H., Thiel, S.U.: FASoft: A system for discrete channel frequency assignment. Radio Science **32** (1997) 1921–1939
18. Galinier, P., Hao, J.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization **3** (1999) 379–397
19. Costa, D.: On the use of some known methods for T-colorings of graphs. Annals of Operations Research **41** (1993) 343–358
20. Castelino, D., Hurley, S., Stephens, N.: A tabu search algorithm for frequency assignment. Annals of Operations Research **63** (1996) 301–320
21. Hao, J.K., Dorne, R., Galinier, P.: Tabu search for frequency assignment in mobile radio networks. Journal of Heuristics **4** (1998) 47–62
22. Hao, J.K., Perrier, L.: Tabu search for the frequency assignment problem in cellular radio networks. Technical Report LGI2P, EMA-EERIE, Parc Scientifique Georges Besse, Nimes, France (1999)
23. Birattari, M.: The race package for R. Racing methods for the selection of the best. Technical Report TR/IRIDIA/2003-37, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2003)
24. Conover, W.: Practical Nonparametric Statistics. third edn. John Wiley & Sons, New York, NY, USA (1999)
25. Chiarandini, M., Basso, D., Stützle, T.: Statistical methods for the comparison of stochastic optimizers. In Doerner, K.F., Gendreau, M., Greistorfer, P., Gutjahr, W.J., Hartl, R.F., Reimann, M., eds.: MIC2005: The Sixth Metaheuristics International Conference, Vienna, Austria (2005) 189–196
26. Matsui, S., Tokoro, K.: 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), Morgan Kaufmann Publishers (2001) 1359–1366