Graph Coloring Tabu Search for Project Scheduling

Nicolas Zufferey

Geneva School of Economics and Management, GSEM - University of Geneva
Blvd du Pont-d'Arve 40, 1211 Geneva 4, Switzerland
n.zufferey@unige.ch

Abstract. Consider a project consisting of a set of n operations to be performed. Some pairs $\{j, j'\}$ of operations are *incompatible*, which can have two different meanings. On the one hand, it can be allowed to perform j and j' at common time periods. In such a case, incompatibility costs are encountered and penalized in the objective function. On the other hand, it can be strictly forbidden to perform j and j' concurrently. In such a case, the overall project duration has to be minimized. In this paper, three project scheduling problems (P_1), (P_2) and (P_3) are considered. It will be showed that tabu search relying on graph coloring models is a very competitive method for such problems. The overall approach is called *graph coloring tabu search* and denoted *GCTS*.

Keywords: Graph coloring, tabu search, project scheduling, combinatorial optimization, metaheuristics.

1 Introduction

Firstly, consider a project (P_1) consisting in n operations to be performed within k time periods, assuming that each operation has a duration of at most one time period. When an operation is assigned to a specific period, an assignment cost is encountered. In addition, for some pairs of operations, an incompatibility cost is encountered if they are performed at the same period. The goal is to assign a period to each operation while minimizing the costs. Secondly, consider problem (P_2), which is an extension of (P_1). For each operation, its duration is known as an integer number of time periods, and preemptions are allowed at integer points of time. The goal is to assign the required number of periods to each operation while minimizing the costs. Thirdly, consider problem (P_3), which consists in a set of operations to be performed, assuming the processing time of each operation is at most one time period. Precedence and incompatibility constraints between operations have to be satisfied. The goal is to assign a time period to each operation while minimizing the duration of the project.

Given a graph $G = (V, E)$ with vertex set V and edge set E, the k-coloring problem (k-GCP) consists in assigning an integer (called color) in $\{1, \ldots, k\}$ to every vertex such that two adjacent vertices have different colors. The *graph coloring problem* (GCP) consists in finding a k-coloring with the smallest possible

H.A. Le Thi et al. (eds.), *Advanced Computational Methods for Knowledge Engineering*,
Advances in Intelligent Systems and Computing 358, DOI: 10.1007/978-3-319-17996-4_10

value of k (the optimal value is denoted $\chi(G)$). Both problems are NP-hard [10] and many heuristics were proposed to solve them. It will be showed that problem (P_1) (resp. (P_2) and (P_3)) can be modeled as an extension of the *graph coloring problem* (resp. *graph multi-coloring problem*, *mixed graph coloring problem*).

Problems (P_1), (P_2) and (P_3) are new and there is no literature on it, except [3,22,30] on which this paper strongly relies. For a recent survey on graph coloring, the reader is referred to [21]. Relevant references for the multi-coloring problem with applications in scheduling are [5,9,12]. For more information on the mixed graph coloring problem and some applications in scheduling, the reader is referred to [1,8,13,26]. Graph coloring approaches for management and scheduling problems are given in [15,28,29]. The reader desiring a review on scheduling models and algorithms is referred to [24]. The reader interested in a general project management book with applications to planning and scheduling is referred to [17]. Finally, the reader interested in project scheduling is referred to [6,16,18,19].

The literature shows that tabu search has obtained competitive results for (P_1), (P_2) and (P_3). Let f be an objective function to minimize. Starting from an initial solution, a *local search* generates at each iteration a *neighbor* solution s' from a *current* solution s by performing a *move* m (i.e. a slight modification on s). In a *descent local search*, the best move is performed at each iteration and the process stops when the first local optimum is reached. To escape from a local optimum, in a tabu search, when a move m is performed to generate s' from s, then the reverse move is forbidden for *tab* (parameter) iterations. At each iteration, the best non tabu move is performed. The process is stopped for example when a time limit is reached. The reader interested in a recent book on metaheuristics is referred to [11], and to [27] for guidelines on an efficient design of metaheuristics according to various criteria.

In this work, it is showed that the project scheduling problems (P_1), (P_2) and (P_3) can be efficiently tackled with a tabu search metaheuristic relying on graph coloring models. The resulting overall approach is called *graph coloring tabu search* and denoted *GCTS*. In this paper, *GCTS* is adapted to (P_1) in Section 2 (relying on [30]), to (P_2) in Section 3 (relying on [3]), and to (P_3) in Section 4 (relying on [22]). The reader is referred to the above mentioned three references to have detailed information on the NP-hard state, the complexity issues, the literature review, the generation of the instances, the experimental conditions and the presentation of the results. For each problem, the main numerical results will highlight the efficiency of *GCTS*. A conclusion is provided in Section 5.

2 Problem (P_1)

2.1 Presentation of the Problem

Consider a project which consists of a set V of n operations to be performed. The project manager provides a target number k of time periods within which the

project has to be performed. It is assumed that: each time period has the same duration (e.g. a working day); there is no precedence constraint between operations; each operation has a duration of at most one time period. Let $c(j, j') \geq 0$ denote an *incompatibility* cost between operations j and j', which is to be paid if j and j' are performed at the same time period. The incompatibility cost $c(j, j')$ represents for example that the same staff has to perform operations j and j', thus additional human resources must be hired in order to be able to perform both operations at the same period. In addition, for each operation j and each time period t, an *assignment* cost $a(j, t)$ has to be paid if j is performed at period t. $a(j, t)$ represents for example the cost of the staff and machines which have to perform operation j at period t. The goal is to assign a time period $t \in \{1, \ldots, k\}$ to each operation $j \in V$ while minimizing the total costs.

A solution using k periods can be generated by the use of a function $per : V \longrightarrow \{1, \ldots, k\}$. The value $per(j)$ of an operation j is the period assigned to j. With each period t can be associated a set C_t that contains the set of operations performed at period t. Thus, a solution s can be denoted $s = (C_1, \ldots, C_k)$, and the associated encountered costs are described in Equation (1). (P_1) consists in finding a solution with k periods which minimizes these costs.

$$f(s) = \sum_{t=1}^{k} \sum_{j \in C_t} a(j, t) + \sum_{j=1}^{n-1} \sum_{j' \in \{j+1, \ldots, n\} \cap C_{per(j)}} c(j, j') \tag{1}$$

2.2 Graph Coloring Model Based on the k-GCP

Let $I(j)$ denote the set of operations j' such that $c(j, j') > 0$. From the input data of problem (P_1), an *incompatibility graph* $G = (V, E)$ can be built as follows. A vertex j is associated with each operation j, and an edge $[j, j']$ is drawn each time $j' \in I(j)$ (but not more than one edge between two vertices). A color t represents a time period t. Coloring G with k colors while minimizing the number of conflicting edges (which is exactly the k-GCP) is equivalent to assign a time period $t \in \{1, \ldots, k\}$ to each operation while minimizing the number of incompatibilities. (P_1) is actually an extension of the k-GCP, because the latter is a subcase of the former where $a(j, t) = 0$ $(\forall j, t)$, and $c(j, j') = 1$ $(\forall j$ with $j' \in I(j))$. From now on, the *project scheduling* terminology (e.g., operations, time periods) and the *graph coloring* terminology (e.g., vertices, colors) are indifferently used.

2.3 Tabu Search

An efficient approach for the k-GCP consists in giving a color to each vertex while minimizing the number of conflicts (a *conflict* occurs if two adjacent vertices have the same color). If this number reaches 0, a legal k-coloring is found. In such a context, a straightforward move is to change the color of a conflicting vertex [14]. For (P_1), the search space is the set of k-partitions of V and the objective function to minimize is the total cost f. A move consists in changing the

period assigned to an operation. In order to avoid testing every possible move at each iteration, only the $q\%$ (parameter tuned to 40%) most costly operations are considered for a move at each iteration. If operation j moves from C_t to $C_{t'}$ when going from the current solution s to the neighbor solution s', it is forbidden to put j back in C_t during $tab(j, C_t)$ iterations as described in Equation (2), where $R(u, v)$ randomly returns an integer in interval $[u, v]$ (uniform distribution), and (u, v, α) are parameters tuned to $(10, 20, 15)$. The maximum is used to enforce $tab(j, C_t)$ to be positive. The last term of Equation (2) represents the improvement $Imp(s, s')$ of f when moving from s to s'. If s' is better than s, $Imp(s, s')$ is positive and the reverse of the performed move will be forbidden for a larger number of iterations than if $Imp(s, s') < 0$. In addition, if the diversity of the visited solutions is below a predetermined threshold, $tab(j, C_t)$ is augmented from that time (for all j and t), and if the diversity becomes above the threshold, the tabu durations are reduced from that time. Note that a diversification mechanism also favors moves which are unlikely to be performed.

$$tab(j, C_t) = \max \left\{ 1, R(u, v) + \alpha \cdot \frac{f(s) - f(s')}{f(s)} \right\} \tag{2}$$

2.4 Results

The stopping condition of each method is a time limit T of one hour on an Intel Pentium 4 (4.00 GHz, RAM 1024 Mo DDR2). The following methods GR, DLS, $GCTS$ and AMA are compared on instances derived from the well-known graph coloring benchmark instances [21]. Note that GR and DLS are restarted as long as T is not reached, and the best encountered solution is returned.

- GR: a greedy constructive heuristic working as follows. Let J be the set of scheduled operations. Start with an empty solution s (i.e. $J = \emptyset$). Then, while s does not contain n operations, do: (1) randomly select an unscheduled operation j; (2) a time period $t \in \{1, \ldots, k\}$ is assigned to j such that the augmentation of the costs is as small as possible.
- $GCTS$: as described in Subsection 2.3.
- DLS: a descent local search derived from $GCTS$ by setting $q = 100\%$ (i.e. considering all the possible moves at each iteration), without tabu tenures.
- AMA: an adaptive memory algorithm [25], where at each generation, an offspring solution s is built from a central memory M (containing 10 solutions), then s is improved with a tabu search procedure relying on $GCTS$, and finally s is used to update the content of M.

For a fixed value of k, Table 1 reports the average results (over 10 runs) obtained with the above methods. Let $f^{(GCTS)}$ be the average value of the solution returned by $GCTS$ (rounded to the nearest integer) over the considered number of runs. $f^{(GR)}$, $f^{(DLS)}$ and $f^{(AMA)}$ are similarly defined. From left to right, the columns indicate: the instance name (incompatibility graph), its number n of operations, its density d (average number of edges between a pair of vertices),

the considered value of k, $f^{(GCTS)}$, and the percentage gap between $f^{(GR)}$ (resp. $f^{(DLS)}$ and $f^{(AMA)}$) and $f^{(GCTS)}$. Average gaps are indicated in the last line. It can be observed that $f^{(GCTS)}$ clearly outperforms the other methods.

Table 1. Results on the (P_1) instances

Graph G	n	d	k	$f^{(GCTS)}$	GR	DLS	AMA
DSJC1000.1	1000	0.1	13	241601	57.23%	28.49%	5.42%
DSJC1000.5	1000	0.5	55	250977	33.30%	18.43%	-0.28%
DSJC1000.9	1000	0.9	149	166102	10.30%	11.35%	-4.98%
DSJC500.5	500	0.5	32	98102	55.03%	34.76%	3.50%
DSJC500.9	500	0.9	84	64224	43.69%	42.71%	2.69%
flat1000_50_0	1000	0.49	33	665449	24.97%	10.82%	-0.82%
flat1000_60_0	1000	0.49	40	462612	28.63%	14.16%	-1.18%
flat1000_76_0	1000	0.49	55	246157	32.15%	18.23%	-1.92%
flat300_28_0	300	0.48	19	62862	51.20%	29.19%	1.50%
le450_15c	450	0.16	10	149041	40.75%	20.45%	2.86%
le450_15d	450	0.17	10	146696	42.89%	22.49%	5.19%
le450_25c	450	0.17	17	72974	39.99%	27.11%	21.32%
le450_25d	450	0.17	17	70852	43.40%	29.40%	23.28%
Average					**38.73%**	**23.66%**	**4.35%**

3 Problem (P_2)

3.1 Presentation of the Problem

(P_2) is an extension of (P_1) where the duration of an operation j is not limited to one time period, but to p_j (integer) periods. Preemptions are allowed at integer time points. The goal is to assign p_j (not necessarily consecutive) periods to each operation j while minimizing assignment and incompatibility costs. The assignment cost $a(j,t)$ is defined as in Subsection 2.1. In addition, let $c^m(j,j') > 0$ (with $m \in \mathbb{N}^\star$) denote the *incompatibility* cost between incompatible operations j and j', which is to be paid if j and j' have m common time periods. From a practical standpoint, it is reasonable to assume that $c^{m+1}(j,j') \geq c^m(j,j')$ $(\forall m)$. For compatible operations j and j', $c^m_{j,j'} = 0$ $(\forall m)$.

In order to represent a solution s, with each time period $t \in \{1,\ldots,k\}$ is associated a set C_t containing the operations which are performed at period t. Each operation j has to belong to p_j sets of type C_t in order to be totally performed. Let $\delta^m_{j,j'} = 1$ if operations j and j' are performed within m common time periods, and 0 otherwise. Thus, a solution s can be denoted $s = (C_1,\ldots,C_k)$, and the associated objective function $f(s)$ to minimize is presented in Equation (3).

$$f(s) = \sum_t \sum_{j \in C_t} a(j,t) + \sum_{j<j',\, m} c^m(j,j') \cdot \delta^m_{j,j'} \tag{3}$$

3.2 Graph Coloring Model Based on the Multi-coloring Problem

In the *k-multi-coloring* problem, each vertex j has to receive a predefined number p_j of colors in $\{1, \ldots, k\}$ such that adjacent vertices have no common color. A *conflict* occurs if two adjacent vertices have at least one color in common. The *graph multi-coloring* problem consists in finding the smallest k for which a k-multi-coloring exists. Among the few existing methods for the multi-coloring problem, tabu search was shown to provide very competitive results [5,7].

A vertex represents an operation, a color is a time period, and the required number p_j of colors to assign to vertex j is the duration of operation j. In contrast with the k-multi-coloring problem, conflicts are allowed in (P_2), but lead to incompatibility costs. In addition, assignment costs are also considered (while they are all equal in the k-multi-coloring problem and can thus be ignored). Therefore, (P_2) is an extension of the k-multi-coloring problem.

3.3 Tabu Search

A feasible solution is any assignment of the correct number of colors to each vertex. The initial solution is a random assignment of p_j colors to each vertex j. A neighbor solution is produced by changing exactly one color on a vertex. Thus, a move (j, t, t') consists in replacing, for a single operation j, a time period t with another time period t'. Assume that move (j, t, t') has just been performed. The moves (j, t, t') and (j, t', t) are then forbidden for *tab* (parameter) iterations, where *tab* is tuned in interval $[1, 50]$ depending on the instance size n.

In order to avoid exhaustive search at each iteration (i.e. evaluating all the possible moves), it is proposed to control the size of the evaluated set of candidate moves by two sensitive parameters N and K, which respectively indicate the considered proportion of operations and time periods. It was observed that the larger is n, the smaller should be N.

3.4 Results

The stopping condition of all (meta)heuristics is a time limit of T seconds, where T depends on the number n of vertices of the graph. $T = 300$ for $n \leq 30$, 600 for $n = 50$, 1200 for $n = 100$, and 3600 for $n = 200$. The following methods *GR*, *DLS* and *GCTS* are compared on randomly generated instances. Note that *GR* and *DLS* are restarted as long as T is not reached, and the best encountered solution is returned.

- *GR*: a greedy constructive algorithm working as follows. At each step, fully color a vertex while minimizing the augmentation of the costs.
- *GCTS*: as described in Subsection 3.3.
- *DLS*: a descent local search derived from *GCTS* by setting $N = K = 100\%$ (an exhaustive search is performed at each iteration), without tabu tenures.

– *GLS*: a genetic local search algorithm, where at each generation, offspring solutions are built from a central memory M (containing 6 solutions), then these solutions are improved with a tabu search procedure relying on *GCTS*, and finally they are used to replace the solutions of M (except the best one).

The tests were executed on an Intel® Core™ i7-2620M CPU @ 2.70GHz with 4GB of RAM (DDR3). For each instance is reported the very best objective function value $f^\star$ ever found by any algorithm during all the performed tests. Results on linear instances are reported in Table 2. The instance name is straightforward. For example, instance n10-d80-k14-p2-P5 has $n = 10$ operations, a density $d = 80\%$, $k = 14$ allowed time periods, and each operation j has a duration p_j in interval $[p, P] = [2, 5]$. On these linear instances, it was possible to use CPLEX 12.4 (during 4 hours, which is above T for any instance) to compute a lower (resp. upper) bound LB (resp. UB) on f. For each instance, the percentage gap of each method (with respect to $f^\star$) is given (averaged over 10 runs). The following observations can be made: (1) CPLEX can provide optimal solution for very small instances only (as $LB = UB$ only for the instances with $n = 10$); (2) *GR* and *DLS* performs very poorly, especially with larger n values; (3) *GCTS* and *GLS* have comparable performances, with a slight advantage to *GLS*, which highlights the benefit of the recombination operator when jointly used with *GCTS*.

Table 2. Results on the (P_2) linear instances

Instance	$f^\star$	LB	UB	GR	DLS	$GCTS$	GLS
n10-d50-k9-p2-P5	62.37	62.37	62.37	84.60%	0.00%	0.00%	0.00%
n10-d80-k14-p2-P5	44.66	44.66	44.66	181.40%	3.90%	0.20%	0.00%
n20-d50-k14-p2-P5	81.99	67.68	82.49	217.60%	2.90%	1.40%	0.50%
n20-d80-k17-p2-P5	181.12	95.2	199.17	138.90%	2.40%	1.30%	0.70%
n30-d50-k20-p2-P6	146.02	68.77	171.37	348.20%	14.40%	1.60%	1.90%
n30-d80-k30-p3-P6	438.28	73.53	640.02	166.30%	8.80%	0.60%	1.00%
n50-d20-k17-p2-P6	134.16	81.73	162.46	531.90%	29.30%	16.80%	3.50%
n50-d50-k22-p1-P4	85.94	38.73	183.59	1075.20%	43.20%	2.40%	3.00%
n100-d20-k20-p1-P5	170.84	85.18	449.86	1096.50%	66.30%	7.70%	7.50%
n200-d20-k25-p1-P5	883.17	100.91	∞	696.10%	65.30%	1.80%	3.70%
Average				**453.67%**	**23.65%**	**3.38%**	**2.18%**

4 Problem (P_3)

4.1 Presentation of the Problem

(P_3) is an extension of (P_1) where incompatibility costs are replaced with incompatibility constraints, assignment costs are ignored, precedence constraints have to be satisfied, and the total duration k of the project has to be minimized. An *incompatibility* constraint between operations j and j' is denoted by $[j, j']$.

For each operation j is given a set $P(j) \subset V$ of immediate predecessor operations. If $j' \in P(j)$, it means that operation j' has to be completely performed before j starts. Such a *precedence* constraint is denoted by (j', j). The goal is to assign a time period t to each operation j while minimizing the total duration of the project, and satisfying the incompatibility and precedence constraints.

Let $(P_3^{(k)})$ be the problem of searching for a feasible solution using k time periods. As explained in Subsection 2.1, such a solution can be generated by using a function *per*, and the notation $s = (C_1, \ldots, C_k)$ can be used. Problem (P_3) consists in finding a feasible solution s using k time periods with the smallest value of k. Starting with $k = n$, one can tackle (P_3) by solving a series of $(P_3^{(k)})$ with decreasing values of k, and the process stops when it is not possible to find a feasible solution with k time periods.

4.2 Graph Coloring Model Based on the Mixed GCP

A *mixed graph* $G = (V, E, A)$ is a graph with vertex set V, edge set E, and arc set A. By definition, an edge $[x, y]$ is not oriented and an arc (x, y) is an oriented edge (from x to y). In the $MGCP$ (mixed graph coloring problem), the goal is to assign a color to every vertex while using a minimum number of colors and satisfying the incompatibility constraints (i.e., two adjacent vertices must get different colors). In addition, for every arc (x, y), the precedence constraint $col(x) < col(y)$ has to be respected (where $col(x)$ is the color assigned to x). For (P_3), there is a *conflict* between vertices x and y if one of the following conditions is true: (1) $y \in I(x)$ (i.e. x and y are incompatible) and $col(x) = col(y)$ (incompatibility violation); (2) $y \in P(x)$ and $col(x) \leq col(y)$ (precedence violation). In both cases, x and y are *conflicting vertices*. In case (1), the conflict occurs on edge $[x, y]$, and in case (2), it occurs on arc (x, y).

From the input data of problem (P_3), one can construct a mixed graph $G = (V, E, A)$ as follows: vertex j represents operation j; if $j' \in I(j)$, then edge $[j, j']$ is drawn to represent an incompatibility (at most one edge between two vertices); if $j'' \in P(j)$, then an arc (j'', j) is drawn to represent a precedence constraint. In addition, a color t can be associated with each time period t. Coloring G with k colors while trying to minimize the number of conflicts is equivalent to assigning a time period $t \in \{1, \ldots, k\}$ to each operation while trying to minimize the number of violations of incompatibility and precedence constraints.

4.3 Tabu Search

GCTS is derived from the tabu search approach proposed for the k-GCP in [2]. The search space is the set of partial but legal solutions of $(P_3^{(k)})$, and the objective function f to minimize is the number of operations without an associated time period. Formally, any solution s can be denoted by $s = (C_1, \ldots, C_k; OUT)$, where C_t is the set of operations performed at time period t (without the occurrence of any conflict), and $|OUT|$ has to be minimized (all the vertices

without a time period are in OUT). Note that if $|OUT| = 0$, it means that a feasible solution has been found with k periods, and the process is restarted with $k-1$ periods, and so on until no feasible solution is found. Then, the provided number of periods will be the last number for which a feasible solution has been found.

A move consists in assigning a time period t to an operation j belonging to OUT. If it creates conflicting operations (in C_t), their associated time period t is removed (i.e., such conflicting vertices are moved from C_t to OUT). When a time period t is assigned to an operation j, it is then tabu to remove t from j during tab (parameter) iterations. At each iteration, the best (according to function g defined below) neighbor solution s' of the current solution s is determined (ties are broken randomly), such that either s' is a non-tabu solution, or $f(s') < f^\star$, where $f^\star$ is the value of the best solution $s^\star$ encountered so far during the search. If operation j is removed from OUT when switching from s to s', it is forbidden to put j back into OUT during $tab(j) = R(u,v) + \gamma \cdot n_c$ iterations, where n_c is the number of conflicts in s, and function $R(u,v)$ is defined as in Subsection 2.3. Parameters u, v and γ are respectively tuned to 0, 9 and 0.6.

Let s be the current solution. Note that f may give the same value to several candidate neighbor solutions of s. At each iteration, in order to better discriminate the choice of a neighbor solution, another objective function g is used instead of f (thus, g is only used to evaluate candidate neighbor solutions). More precisely, a conflict can be due to an incompatibility constraint violation or to a precedence constraint violation. It was observed that it is better to give different weights to these two types of conflicts. Given a partial solution $s = (C_1, \ldots, C_k; OUT)$, an operation $j \in OUT$ and a time period $t \in \{1, \ldots, k\}$, two quantities are computed: (1) $A(j,t)$ is the set of incompatible operations which will be put in OUT if time period t is assigned to operation j; (2) $B(j,t)$ is the set of operations, involved in precedence constraint violations, which will be put in OUT if time period t is given to operation j. At each step of $GCTS$, the move which minimizes $g(j,t) = \alpha \cdot |A(j,t)| + \beta \cdot |B(j,t)|$ is performed, where α and β are parameters tuned to 4 and 1, respectively. With such an objective function g, it is very quick and accurate to evaluate a neighbor solution.

4.4 Results

The stopping condition of all (meta)heuristics is a time limit of $T = 3600$ seconds. The following methods GR, $GCTS$ and VNS are compared on instances derived from the well-known graph coloring benchmark instances [21]. Note that GR is restarted as long as T is not reached, and the best encountered solution is returned.

- GR: a greedy constructive algorithm derived from [4] and working as follows. At each step, select a vertex j and assign to it the smallest possible color without creating any conflict. If it is not possible, put j in OUT.
- $GCTS$: as described in Subsection 4.3.
- VNS: a variable neighborhood search [23], using $GCTS$ as intensification procedure.

Our algorithms were implemented in C++ and run on a computer with the following properties: Processor Intel Core2 Duo Processor E6700 (2.66GHz, 4MB Cache, 1066MHz FSB), RAM 2GB DDR2 667 ECC Dual Channel Memory (2x1GB). The results are presented in Table 3. The five first columns respectively indicate the following information: the name of the graph, the number n of vertices, the smallest number of colors $k^\star$ for which a legal $k^\star$-coloring was found by an algorithm or the chromatic number $\chi(G)$ if it is known, the edge density d, and the arc density $\hat{d}$. The last three columns respectively indicate the smallest number of colors for which a legal coloring was found by GR, $GCTS$, and VNS, with the number of successes among five runs in brackets. As expected, larger d and $\hat{d}$ values lead to a larger number of used colors. One can observe that $GCTS$ outperforms both GR and VNS.

Table 3. Results on the (P_3) instances

Graph	n	$k^\star$	d	$\hat{d}$	GR	$GCTS$	VNS
DSJC250.1	250	8	0.1	0.005	9 (5)	8 (5)	8 (5)
				0.1	30 (5)	30 (5)	30 (5)
DSJC250.5	250	28	0.5	0.005	36 (5)	30 (1)	31 (4)
				0.01	39 (5)	35 (5)	38 (1)
DSJC250.9	250	72	0.9	0.005	89 (5)	78 (5)	82 (3)
				0.01	95 (5)	91 (2)	97 (1)
DSJR500.1	500	12	0.03	0.005	12 (5)	12 (5)	12 (5)
				0.1	19 (5)	19 (5)	19 (5)
DSJR500.1c	500	85	0.97	0.005	183 (5)	187 (1)	186 (1)
				0.01	279 (5)	285 (3)	285 (4)
DSJR500.5	500	122	0.47	0.005	137 (5)	132 (5)	138 (1)
				0.01	146 (5)	149 (1)	148 (2)
le450_15c	450	15	0.16	0.005	24 (5)	18 (5)	18 (2)
				0.01	25 (5)	21 (5)	22 (4)
le450_15d	450	15	0.17	0.005	24 (5)	18 (5)	18 (2)
				0.01	25 (5)	20 (1)	22 (1)
le450_25c	450	25	0.17	0.005	29 (5)	28 (5)	27 (1)
				0.01	30 (5)	29 (5)	29 (4)
le450_25d	450	25	0.17	0.005	29 (5)	28 (5)	28 (5)
				0.01	30 (5)	29 (5)	29 (4)
flat300_20_0	300	20	0.47	0.005	40 (5)	27 (3)	27 (1)
				0.01	42 (5)	32 (1)	40 (3)
flat300_26_0	300	26	0.48	0.005	41 (5)	34 (4)	35 (1)
				0.01	42 (5)	38 (5)	42 (2)
flat300_28_0	300	28	0.48	0.005	40 (5)	35 (5)	35 (2)
				0.01	44 (5)	40 (1)	45 (1)

5 Conclusion

In this work, $GCTS$ is discussed (with a unified view), which is a tabu search approach relying on graph coloring models. It was showed that $GCTS$ was efficiently adapted to three project scheduling problems. This success mainly relies on four aspects:

- the use of graph coloring models to represent the considered problem and its solutions;
- the use of an auxiliary objective function instead of the provided one (e.g., for problem (P_3), fix k and minimize OUT, instead of minimizing k directly);
- the use of moves focusing on the costly operations (if costs have to be minimized) or on the removal of conflicts (if constraint violations are penalized);
- an efficient management of the tabu durations (e.g., depending on the quality of the performed moves).

This paper contributes to build bridges between the graph coloring and the project scheduling communities. An avenue of research consists in reducing the dimension of the project graph before triggering the solution methods (e.g., [20]).

References

1. Al-Anzi, F.S., Sotskov, Y.N., Allahverdi, A., Andreev, G.V.: Using Mixed Graph Coloring to Minimize Total Completion Time in Job Shop Scheduling. Applied Mathematics and Computation 182(2), 1137–1148 (2006)
2. Bloechliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & Operations Research 35, 960–975 (2008)
3. Bloechliger, I., Zufferey, N.: Multi-Coloring and Project-Scheduling with Incompatibility and Assignment Costs. Annals of Operations Research 211(1), 83–101 (2013)
4. Brélaz, D.: New Methods to Color Vertices of a Graph. Communications of the Association for Computing Machinery 22, 251–256 (1979)
5. Chiarandini, M., Stuetzle, T.: Stochastic local search algorithms for graph set T-colouring and frequency assignment. Constraints 12, 371–403 (2007)
6. Demeulemeester, E.L., Herroelen, W.S.: Project Scheduling: A Research Handbook. Kluwer Academic Publishers (2002)
7. Dorne, R., Hao, J.-K.: Meta-heuristics: Advances and trends in local search paradigms for optimization, chapter Tabu search for graph coloring, T-colorings and set T-colorings, pp. 77–92. Kluwer, Norwell (1998)
8. Furmańczyk, H., Kosowski, A., Żyliński, P.: Scheduling with precedence constraints: Mixed graph coloring in series-parallel graphs. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1001–1008. Springer, Heidelberg (2008)
9. Gandhi, R., Halldórsson, M.M., Kortsarz, G., Shachnai, H.: Improved bounds for sum multicoloring and scheduling dependent jobs with minsum criteria. In: Persiano, G., Solis-Oba, R. (eds.) WAOA 2004. LNCS, vol. 3351, pp. 68–82. Springer, Heidelberg (2005)
10. Garey, M., Johnson, D.S.: Computer and Intractability: a Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
11. Gendreau, M., Potvin, J.-Y.: *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol. 146. Springer, Heidelberg (2010)
12. Halldórsson, M.M., Kortsarz, G.: Multicoloring: Problems and techniques. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 25–41. Springer, Heidelberg (2004)
13. Hansen, P., Kuplinsky, J., de Werra, D.: Mixed Graph Coloring. Mathematical Methods of Operations Research 45, 145–169 (1997)

14. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing 39, 345–351 (1987)
15. Hertz, A., Schindl, D., Zufferey, N.: A solution method for a car fleet management problem with maintenance constraints. Journal of Heuristics 15(5), 425–450 (2009)
16. Icmeli, O., Erenguc, S.S., Zappe, C.J.: Project scheduling problems: A survey. International Journal of Operations & Production Management 13(11), 80–91 (1993)
17. Kerzner, H.: Project Management: A Systems Approach to Planning, Scheduling, and Controlling. Wiley (2003)
18. Kolisch, R., Padman, R.: An integrated survey of deterministic project scheduling. Omega 29(3), 249–272 (2001)
19. Lancaster, J., Ozbayrak, M.: Evolutionary algorithms applied to project scheduling problems – a survey of the state-of-the-art. International Journal of Production Research 45(2), 425–450 (2007)
20. Luyet, L., Varone, S., Zufferey, N.: An Ant Algorithm for the Steiner Tree Problem in Graphs. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 42–51. Springer, Heidelberg (2007)
21. Malaguti, E., Toth, P.: A survey on vertex coloring problems. International Transactions in Operational Research 17(1), 1–34 (2010)
22. Meuwly, F.-X., Ries, B., Zufferey, N.: Solution methods for a scheduling problem with incompatibility and precedence constraints. Algorithmic Operations Research 5(2), 75–85 (2010)
23. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research 24, 1097–1100 (1997)
24. Pinedo, M.: Scheduling: Theory, Algorithms, and Systemsmulti-coloring. Prentice Hall (2008)
25. Rochat, Y., Taillard, E.: Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167 (1995)
26. Sotskov, Y.N., Dolgui, A., Werner, F.: Mixed Graph Coloring for Unit-Time Job-Shop Scheduling. International Journal of Mathematical Algorithms 2, 289–323 (2001)
27. Zufferey, N.: Metaheuristics: some Principles for an Efficient Design. Computer Technology and Applications 3(6), 446–462 (2012)
28. Zufferey, N.: Graph Coloring and Job Scheduling: from Models to Powerful Tabu Search Solution Methods. In: Proceedings of the 14th International Workshop on Project Management and Scheduling (PMS 2014), Munich, Germany, March 31 – April 2 (2014)
29. Zufferey, N., Amstutz, P., Giaccari, P.: Graph colouring approaches for a satellite range scheduling problem. Journal of Scheduling 11(4), 263–277 (2008)
30. Zufferey, N., Labarthe, O., Schindl, D.: Heuristics for a project management problem with incompatibility and assignment costs. Computational Optimization and Applications 51, 1231–1252 (2012)