A branch-and-price algorithm for the robust graph coloring problem

Claudia Archetti[(1)] *Nicola Bianchessi*[(2)] *Alain Hertz*[(3)]

[(1)] *Department of Quantitative Methods*
University of Brescia, Brescia, Italy
[(2)] *Department of Information Engineering*
University of Brescia, Brescia, Italy
[(3)] *Department of Mathematics and Industrial Engineering*
École Polytechnique and GERAD, Montréal, Canada

December 14, 2011

Abstract

Given a graph G, an integer k, and a cost c_{uv} associated with all pairs uv of non-adjacent vertices in G, the robust graph coloring problem is to assign a color in $\{1, \cdots, k\}$ to every vertex of G so that no edge has both endpoints with the same color, and the total cost of the pairs of vertices having the same color is minimum. We propose a branch and price algorithm for the solution of this problem. The pricing problem consists in finding a stable set of minimum total weight, and we propose both an exact and a heuristic algorithm for its solution. Computational experiments are reported for randomly generated and benchmark graph coloring instances.

Keywords: graph coloring, robust solution, branch-and-price algorithm, tabu search.

1 Introduction

The classical graph coloring problem is to assign a color to every vertex of a graph G so that no edge in G has both endpoints with the same color. Task assignment problems can be modeled as graph coloring problems, where each task is associated with a vertex, edges link vertices corresponding to overlapping tasks, and each color represents a person to be assigned to a set of tasks. The constraint imposing that no edge can have both endpoints with the same color is then equivalent to requiring that no two tasks overlapping in time are performed by the same person.

In practice, tasks are often subject to delays and it can therefore happen that two tasks assigned to the same person finally overlap, which makes the assignment infeasible. Task managers are therefore interested in determining a robust assignment that is not too much sensible to delays. For this purpose, one can for example associate a positive integer to each pair of non-overlapping tasks, to indicate the estimated probability that these two tasks will overlap in case of delays. Such a weighting procedure is illustrated in [8] for a crew assignment problem where crew teams have to be assigned to round-trip domestic flight routes from Hong Kong to other mainland China cities. Given two non-overlapping round trips T_i and T_j such that T_i precedes T_j, one can associate a weight $\omega_{ij} = \frac{g}{d-t+\alpha}$ to the pair $\{T_i, T_j\}$, where t is the return time of T_i, d is the departure time of T_j, g is the ground transfer time, and α is a constant. The problem is then to find a task assignment that minimizes the total weight of the pairs $\{T_i, T_j\}$ of round trips assigned to a same crew team.

Using graph theoretical terms, one can associate a positive weight to each pair of non-adjacent vertices, and then, rather than minimizing the number of colors needed to color the vertices of the graph, one can decide to use a fixed number of colors and try to minimize the total weight of the pairs of vertices having the same color. This is the Robust Graph Coloring Problem (RGCP) introduced by Ramírez-Rodríguez in his doctoral dissertation [12]. Applications of this extension of the classical graph coloring problem are mentioned in [14], and range from the examination timetabling problem to cluster analysis. A mathematical formulation of the RGCP as a binary linear programming problem is proposed in [14, 8] and the same authors also propose genetic algorithms to solve the problem. Another genetic algorithm, as well as a simulated annealing and a tabu search for the RGCP are described in [10] and a local search procedure is proposed in [6].

In this paper we propose a branch-and-price algorithm for the RGCP. A formal definition of the RGCP in given in the next section while the proposed exact algorithm is described in Section 3. Experimental results are reported in Section 4.

2 Problem definition

We are given an undirected graph $G = (V, E)$, where V is a finite set of vertices and E is a finite set of edges. Let $\bar{G}$ be the complement of G, i.e., $\bar{G} = (V, \bar{E})$, with $\bar{E} = \{uv : u, v \in V, u \neq v, uv \notin E\}$. A set S of vertices in G is *stable* if no edge in G has both endpoints in S, i.e. $uv \in \bar{E}$ for all $u, v \in S$. Given an integer k, a *k-coloring* of G is a partition of V into k stable sets, each stable set of the partition being called a *color class*. The *chromatic number* $\chi(G)$ of G is the smallest integer k such that there exists a k-coloring of G.

Assume that a non negative cost c_{uv} is associated to each pair $uv \in \bar{E}$. The Robust Graph Coloring Problem (RGCP) is the problem of finding a k-coloring of G for a given $k \geq \chi(G)$, such that the total cost of the pairs uv with u and v in a same color class is

minimized. For illustration, a graph G, its complement $\bar{G}$, an optimal robust 3-coloring with total cost 4, and an optimal robust 4-coloring with total cost 1 are represented on Figure 1.

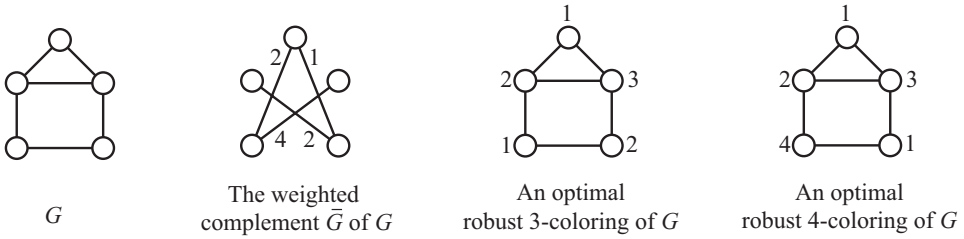

Figure 1. Illustration of optimal robust colorings.

3 A branch-and-price algorithm

In [14], Yáñez and Ramírez describe a binary linear programming model for the solution of the RGCP. They use the following binary variables:

- $x_{ui} = 1$ if color i is assigned to vertex u, 0 otherwise;
- $y_{uv} = 1$ if vertices u and v belong to the same color class, 0 otherwise.

The optimal solution of the RGCP can then be determined by solving the following problem:

$$\min \sum_{uv \in \bar{E}} c_{uv} y_{uv} \tag{1}$$

$$\sum_{i=1}^{k} x_{ui} = 1 \qquad \forall u \in V \tag{2}$$

$$x_{ui} + x_{vi} \leq 1 \qquad \forall uv \in E \quad \forall i \in \{1, \cdots, k\} \tag{3}$$

$$x_{ui} + x_{vi} - 1 \leq y_{uv} \qquad \forall uv \in \bar{E} \quad \forall i \in \{1, \cdots, k\} \tag{4}$$

$$x_{ui} \in \{0, 1\} \qquad \forall u \in V \quad \forall i \in \{1, \cdots, k\} \tag{5}$$

$$y_{uv}\{0, 1\} \qquad \forall uv \in \bar{E} \tag{6}$$

Constraints (2) impose a color in $\{1, \cdots, k\}$ on each vertex, constraints (3) ensure that each color class is a stable set and constraints (4) link the x variables with the y variables. As shown in [14], this model is not able to solve problems with more than 15 vertices.

We propose a set covering formulation for the RGCP. We denote by $\mathcal{S}$ the set of all stable sets in G, and for a stable set $s \in \mathcal{S}$, we define

- $c_s = \sum_{u,v \in s} c_{uv}$
- σ_s = binary variable equal to 1 if s is chosen as a color class, 0 otherwise.

The RGCP is then to find the optimal solution of the following set covering model:

$$\min \sum_{s \in S} c_s \sigma_s \tag{7}$$

$$\sum_{s \in \mathcal{S}: u \in s} \sigma_s \geq 1 \qquad \forall u \in V \tag{8}$$

$$\sum_{s \in \mathcal{S}} \sigma_s = k \tag{9}$$

$$\sigma_s \in \{0, 1\} \qquad \forall s \in \mathcal{S} \tag{10}$$

The objective function (7) minimizes the total cost of the k-coloring. Constraints (8) state that each vertex has to belong to a color class while constraint (9) imposes the use of exactly k color classes.

The number of stable sets in $\mathcal{S}$ is typically exponential in the number of vertices. Thus, model (7)–(10) is solved by means of a *branch-and-price* algorithm [1, 2], that is a branch-and-bound algorithm where, at each node of the tree, the continuous relaxation of (7)-(10), augmented by branching constraints, is solved by means of column generation. From now on we call the problem (7)–(10), eventually augmented by branching constraints, Master Problem (MP). Its continuous relaxation will be referred to as Linear Master Problem (LMP).

In order to solve the LMP by means of column generation, we consider an initial restricted linear master problem (RLMP) defined over a subset of stable sets $\mathcal{S}' \subseteq \mathcal{S}$. Then, when the RLMP is solved to optimality, the dual solution of the RLMP can be used to search for new stable sets with negative reduced cost. This is done by solving a particular subproblem, also called *pricing problem*. If such a stable set exists, the corresponding variable is added to the RLMP and the process is iterated starting from the new RLMP. Otherwise, the optimal solution of the current RLMP is also optimal for the LMP. In order to accelerate the solution of the LMP, at each iteration of the column generation algorithm, heuristic solution methods can be applied to the pricing problem before solving it to optimality. If the heuristic algorithms succeed in finding negative reduced cost columns, then these columns are added to the RLMP and the exact pricing solver is not called. Otherwise, the pricing problem is solved to optimality.

At a given node of the branch-and-bound tree, the optimal LMP solution may not satisfy integrality constraints. In this case, branching has to be performed.

We now describe in detail the main components of the branch-and-price approach.

3.1 Pricing problem.

The pricing problem consists in finding a stable set s with the most negative reduced cost. More formally, let us consider the following binary variables:

- $x_u = 1$ if vertex u belongs to the stable set s, 0 otherwise;
- $y_{uv} = 1$ if both vertices u and v belong to the stable set s, 0 otherwise.

At the root node of the branch-and-bound tree, the pricing problem is the following:

$$\min \quad \bar{c}^* = \sum_{uv \in \bar{E}} c_{uv} y_{uv} - \sum_{u \in V} \pi_u x_u - \beta \tag{11}$$

$$x_u + x_v \leq 1 \qquad \forall uv \in E \tag{12}$$

$$x_u + x_v - 1 \leq y_{uv} \qquad \forall uv \in \bar{E} \tag{13}$$

$$x_u \in \{0, 1\} \qquad \forall u \in V \tag{14}$$

$$y_{uv}\{0, 1\} \qquad \forall uv \in \bar{E} \tag{15}$$

where:

- π_u is the dual cost associated with constraint (8) for vertex u;
- β is the dual cost associated with constraint (9).

Constraints (12) impose that the chosen vertices belong to a stable set, while constraints (13) link variables y with variables x. Constraints (12) and (13) play the same role as constraints (3) and (4) of the model proposed in [14].

At the following nodes of the branch-and-bound tree, the pricing problem is modified by taking into account the branching constraints, as explained in the next section.

3.2 Branching rules.

When the optimal solution of the LMP is fractional, we use the standard branching rules proposed in [13] and implemented also in [11] for the solution of the graph coloring problem. More precisely, given two vertices u, v with $uv \in \bar{E}$, we force u and v to be in different stable sets on one branch (which corresponds to setting $x_u + x_v \leq 1$ in the pricing problem), and to be in the same stable set on the other branch (which corresponds to setting $x_u = x_v$ in the pricing problem). The pair u, v is chosen as follows: for every $uv \in \bar{E}$ we compute $\mu_{uv} = \sum_{s \in \mathcal{S}: u,v \in s} \sigma^*_s$, where σ^* is the optimal solution of the LMP; we then branch on the pair u, v with the most fractional value μ_{uv} (ties are broken at random).

3.3 Pricing problem solution.

We propose an exact and a heuristic algorithm for the solution of the pricing problem. The exact algorithm consists in solving model (11)–(15), eventually augmented by branching constraints, through a commercial solver.

On the heuristic side, we devised a tabu search algorithm since it has been shown to be very effective in finding optimal stable sets [3, 5]. We first build a new graph $\widetilde{G}$ from the original graph G in order to take the branching constraints into account. This is done as follows: if two vertices u and v have to be in different color classes, we add an edge between u and v; if u and v must receive the same color, we merge them into a single vertex adjacent to all vertices adjacent to u and/or v in G.

The solution space explored by the tabu search is the set $\widetilde{\mathcal{S}}$ of all stable sets in $\widetilde{G}$, starting from an initial stable set $s \in \widetilde{\mathcal{S}}$, and iteratively moving from a solution to a neighbour solution, until a stopping criterion is met. Given a stable set s, the neighborhood $N(s)$ of s is constructed as follows. For every vertex $u \notin s$, we first add u to s and remove all vertices in s adjacent to u in order to get a stable set. We then remove additional vertices not adjacent to u if such removals strictly decrease the objective (11). We thus get a stable set $s_u \in \widetilde{\mathcal{S}}$, and $N(s)$ is defined as $\{s_u : u \notin s\}$.

When a vertex is removed from a solution s, it is tabu to insert it back for the next $L + \lambda\theta\sqrt{|V|}$ iterations, where θ is randomly generated with a uniform distribution in the interval $[0, 1]$, while L and λ are parameters of the tabu search. The tabu status of a solution $s' \in N(s)$ is removed if the best solution found so far has a positive reduced cost while s' has a negative reduced cost. The best non tabu solution $s' \in N(s)$ is chosen as the new current solution. All solutions with a strictly negative reduced cost found by the tabu search algorithm are stored in a pool $\mathcal{P}$ of solutions which is then passed to the column generation algorithm as a set of variables with negative reduced cost.

The tabu search needs an initial solution to start with. In our case, we consider the list $\mathcal{L}$ of all stable sets s corresponding to the σ_s variables with 0 reduced cost in the optimal solution of the current RLMP. The tabu search starts with the first stable set in $\mathcal{L}$. After $nmax'$ iterations without improvement of the best solution found so far or $tmax'$ seconds, the tabu search is stopped and restarted with the next stable set in $\mathcal{L}$. The heuristic algorithm is stopped when one of the following conditions holds:

- all stable sets in $\mathcal{L}$ have been considered;
- $nmax''$ stable sets with a strictly negative reduced cost have been found;
- $tmax''$ seconds are elapsed.

3.4 Column generation.

The set $\mathcal{S}'$ used to initialize the RLMP is constructed as follows. At the root of the branch-and-bound tree, we first generate k stable sets $s_1, \cdots, s_k$. Each s_i is initialized to

the empty set, and the vertices of G are then considered, one after the other, for possible inclusion in a set s_i. If every set s_i $(1 \leq i \leq k)$ contains at least one vertex adjacent to the current considered vertex v, we move to the next vertex (i..e., vertex v is not inserted in a s_i); otherwise, v is added to the stable set s_i with smallest index i and such that there is no vertex adjacent to v in s_i. We thus get at most k non-empty stable sets which define the initial set $\mathcal{S}'$. Note that it may happen that $\bigcup_{i=1}^{k} s_i \neq V$, which means that the RLMP has no feasible solution if restricted to this subest $\mathcal{S}'$. For this reason, at each node of the branch-and-bound tree, we add a high cost dummy column in $\mathcal{S}'$. It represents a fictitious color class containing all vertices of G. This column is kept in the model until feasibility is reached. Also, at each node of the branch-and-bound tree, all columns (i.e. stable sets) generated so far which are feasible with respect to branching constraints are inserted in the RLMP.

Once the RLMP has been solved to optimality, we consider the pricing problem in order to find new negative reduced cost columns. If the optimal solution of the pricing problem has a positive value, it means that the optimal solution of the current RLMP is also optimal for the LMP. Algorithm 1 summarizes the column generation algorithm.

Algorithm 1 Column generation algorithm for the solution of the LMP

Step 1. Initialize the RLMP with the subset of stable sets $\mathcal{S}' \subseteq \mathcal{S}$.
Step 2. Solve the RLMP.
Step 3. Apply the heuristic algorithm to solve the pricing problem. Let $\mathcal{P}$ be the set of negative reduced cost columns found.
if $\mathcal{P} \neq \emptyset$ **then**
  Step 4. $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathcal{P}$.
  Step 5. Go to Step 2.
else
  Step 6. Apply the exact algorithm to determine the stable set $\hat{s}$ associated with the optimal solution of the pricing problem.
  if $\bar{c}^*_{\hat{s}} < 0$ **then**
    Step 7. $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{\hat{s}\}$.
    Step 8. Go to Step 2.
  else
    Step 9. Stop: the optimal solution of the current RLMP is optimal for the LMP.
  end if
end if

As shown in [9], each time the pricing problem is solved to optimality, a valid lower bound $\underline{z}^*_{LMP}$ on the optimal value of the LMP can be computed as

$$\underline{z}^*_{LMP} = z^*_{RLMP} + k\bar{c}^*,$$

where z^*_{RLMP} is the optimal value of the current RLMP and $\bar{c}^*$ is the value of the optimal solution of the pricing problem. This allows to stop the column generation algorithm and prune the node of the branch-and-bound tree as soon as $\underline{z}^*_{LMP}$ is greater than or equal to the value of the best incumbent feasible solution.

4 Computational experiments

To test our branch-and-price algorithm, we have considered two sets of instances that can be found at http://www-c.eco.unibs.it/~archetti/rgcp.zip. The first set of instances was generated with the same technique as in [14]. The number n of vertices is a multiple of 10 and varies from 10 to 150, and the graph density (i.e., the probability that an edge exists between two arbitrary vertices) is 0.5. The number k of colors is set equal to $\lceil \frac{n}{3} \rceil$ and $\lceil \frac{n}{3} \rceil + 1$. The costs associated to each pair of non-adjacent vertices are real numbers randomly generated according to a uniform distribution in the interval $]0, 1[$. For each combination of number of nodes and number of colors we have generated 5 instances, for a total of 150 instances.

The instances in the second set are derived from the list of DIMACS benchmark graph coloring instances with up to 150 vertices. For a detailed description of these instances, the reader can refer to [7] or http://mat.gsia.cmu.edu/COLOR04. In these instances, each vertex u has a label $\ell(u) \in \{1, \cdots, |V|\}$ and we set $c_{uv} = \ell(u)\ell(v)$ for every pair uv of non-adjacent vertices. The number k of colors is set equal to $\lceil \frac{3}{2}\bar{k} \rceil$ and $\lceil 2\bar{k} \rceil$, where $\bar{k}$ is either the chromatic number, when it is known, or the best known upper bound on the chromatic number, as reported in [4].

The branch-and-price algorithm was implemented in C++ and all experiments were carried out on an AMD Athlon 64 X2 Dual Core Processor 5600+, 2.89 GHz machine with 3.37 GB of RAM. CPLEX 12.2.0.2 was used to solve the RLMP and the pricing problem to optimality. The overall execution time limit for each instance was set to 2 hours.

According to a preliminary testing phase we have set the parameters of the tabu search algorithm as follows: the parameters L and λ that define the tabu tenure have been set to 5 and 0.5, respectively; for the termination criteria, we have set $nmax' = 50$, $nmax'' = 300$, $tmax' = 60$ seconds, and $tmax'' = 300$ seconds.

The results of the experiments on the first set of instances are summarized in Table 1. The first three columns indicate the name of the instance, the number of vertices and the number k of colors used in the experiments. The next two columns provide information on the root node solution: the optimal value of the LMP appears under column labelled ($\underline{z}^*(\%)$), and is expressed as a percentage of the best lower bound (column 6) obtained at the end of the branch-and-price algorithm; the time (in seconds) required to compute it is reported in the next column. The last four columns give the best lower and upper bounds obtained at the end of the branch-and-price algorithm (columns $\underline{z}^*$ and $\overline{z}^*$, respectively),

the percentage gap between these two values (column gap), and, in case of a null gap, the time (in seconds) required to prove optimality (column t). An italic entry in the first column means that the corresponding instance is not feasible.

As can be seen in Table 1, the branch-and-price algorithm is able to solve systematically all instances with up to 80 vertices and solves all instances except 3 with up to 100 vertices. The largest instance solved to optimality within the time limit of two hours has 130 vertices. In all cases where a feasible solution is found, the final gap with the lower bound is lower than 0.40%. Moreover, the root node relaxation provides an extremely good lower bound since it is always greater than 97% of the final lower bound, except in one case, for a graph with 10 vertices. To the best of our knowledge, the only exact algorithm proposed in the literature for the solution of the RGCP is based on formulation (1)–(6) [14]. The authors have performed experiments on instances similar to those in our first set, and it was shown that only instances with up to 15 vertices can be solved with such a formulation within half an hour of computation. We can therefore conclude that the branch-and-price algorithm based on the new formulation (7)–(10) outperforms the previous exact solution approach proposed in [14].

To better analyze the behavior of our branch-and-price algorithm, we report in Table 2 the results obtained with the second set of instances. As can be observed in the second column, the instances are ordered according to their number of vertices. Columns 3-9 refer to the experiments with a number $k = \lceil \frac{3}{2}\bar{k} \rceil$ of colors, while columns 10-16 contain the results for $k = \lceil 2\bar{k} \rceil$. A '-' means that no result is available either because the time limit has been reached or because CPLEX went out of memory. Note that instance miles1500 has 128 vertices and its chromatic number is known as being equal to 73. Hence, by fixing $k = \lceil 2\bar{k} \rceil = 156$, we get a number of colors that is larger than the number of vertices, and an optimal solution can then easily be obtained by assigning a different color to each vertex. For this reason, we do not report any result under columns 10-16 for miles1500.

It clearly appears that this second set of instances is more challenging than the first one. Also, instances with $k = \lceil \frac{3}{2}\bar{k} \rceil$ colors are harder to solve than the ones with $k = \lceil 2\bar{k} \rceil$ colors. Indeed, the smallest instance not solved to optimality with $k = \lceil \frac{3}{2}\bar{k} \rceil$ has 37 vertices, while this number raises up to 52 with $k = \lceil 2\bar{k} \rceil$. Also, on instances with more than 50 nodes, our algorithm is not able to solve the LMP at the root node for 26 of the 32 instances when using a number $k = \lceil \frac{3}{2}\bar{k} \rceil$ of colors, while this number reduces to 22 for $k = \lceil 2\bar{k} \rceil$. The largest instances that could be solved to optimality are DSJC125.9, miles1000 and miles1500. While these instances have 125 or 128 vertices, they are easy to solve since the best known upper bound on the chromatic number is larger than 40% of the number of vertices, which means that by setting $k = \lceil \frac{3}{2}\bar{k} \rceil$, most color classes will have 1 or 2 vertices. For comparison, the smallest instance which could not be solved to optimality has 37 vertices and a chromatic number equal to 4, which means that the average number of vertices in each color class raises up to more than 6 when using $k = \lceil \frac{3}{2}\bar{k} \rceil = 6$ colors.

5 Conclusions

In this paper we propose a set covering formulation with an exponential number of variables for the robust graph coloring problem. The model is solved through a column generation algorithm embedded in a branch-and-bound scheme, giving rise to a branch-and-price solution approach. In order to decrease the solution time, a tabu search algorithm is proposed to solve the pricing problem. The exact pricing solver is called only when the tabu search is not able to find any negative reduced cost column. Computational experiments on a first set of randomly generated instances show that the branch-and-price is extremely robust in solving instances with up to 130 vertices, thus outperforming the previous exact approach which solved similar instances with at most 15 vertices. A second set of experiments performed on benchmark graph coloring instances has clearly shown that the difficulty of an instance not only depends on the number of vertices, but also on the average size of the color classes. While an instance with 128 vertices could be solved to optimality when using 63 colors, the optimal solution of a much smaller instance with 37 vertices could not be obtained with $k = 6$ colors.

Table 1: Results for the first set of instances

Instance	n	k	Branch-and-price					
			root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
rcp10.1	10	4	100.00	0.093	1.747	1.747	0	0.093
rcp10.2	*10*	*4*	-	-	-	-	-	-
rcp10.3	10	4	100.00	0.094	3.612	3.612	0	0.094
rcp10.4	*10*	*4*	-	-	-	-	-	-
rcp10.5	10	4	100.00	0.125	1.733	1.733	0	0.125
rcp10.1	10	5	100.00	0.031	1.929	1.929	0	0.031
rcp10.2	10	5	100.00	0.031	0.997	0.997	0	0.031
rcp10.3	10	5	96.94	0.031	1.796	1.796	0	0.078
rcp10.4	10	5	100.00	0.031	1.599	1.599	0	0.031
rcp10.5	10	5	100.00	0.125	2.659	2.659	0	0.125
rcp20.1	20	7	98.01	0.484	5.931	5.931	0	1.984
rcp20.2	20	7	99.88	0.421	6.290	6.290	0	1.000
rcp20.3	20	7	100.00	0.281	5.020	5.020	0	0.281
rcp20.4	20	7	99.17	0.265	5.914	5.914	0	2.390
rcp20.5	20	7	100.00	0.922	5.645	5.645	0	0.922
rcp20.1	20	8	99.18	0.375	4.838	4.838	0	1.531
rcp20.2	20	8	100.00	0.390	3.184	3.184	0	0.390
rcp20.3	20	8	100.00	0.234	2.169	2.169	0	0.250
rcp20.4	20	8	98.79	0.296	3.669	3.669	0	0.968
rcp20.5	20	8	97.54	0.453	3.815	3.815	0	1.500
rcp30.1	30	10	100.00	1.031	6.091	6.091	0	1.031
rcp30.2	30	10	100.00	0.578	7.407	7.407	0	0.578
rcp30.3	30	10	97.94	1.515	8.483	8.483	0	10.671
rcp30.4	30	10	100.00	1.000	6.570	6.570	0	1.000
rcp30.5	30	10	98.47	1.265	6.737	6.737	0	7.656
rcp30.1	30	11	100.00	1.031	5.378	5.378	0	1.031
rcp30.2	30	11	97.67	0.969	5.108	5.108	0	4.172
rcp30.3	30	11	100.00	0.531	4.897	4.897	0	0.547
rcp30.4	30	11	98.48	0.562	5.068	5.068	0	4.265
rcp30.5	30	11	100.00	0.985	4.897	4.897	0	0.985
rcp40.1	40	14	99.92	2.188	5.970	5.970	0	5.547
rcp40.2	40	14	99.96	4.172	6.242	6.242	0	7.235
rcp40.3	40	14	100.00	1.453	7.657	7.657	0	1.453
rcp40.4	40	14	99.41	1.671	6.146	6.146	0	6.093
rcp40.5	40	14	99.57	1.953	6.185	6.185	0	5.610

Table 1: Results for the first set of instances

Instance	n	k	Branch-and-price					
			root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
rcp40.1	40	15	100.00	2.328	5.367	5.367	0	2.328
rcp40.2	40	15	99.51	1.453	4.819	4.819	0	3.578
rcp40.3	40	15	99.29	1.297	5.224	5.224	0	11.500
rcp40.4	40	15	99.04	2.672	5.544	5.544	0	10.735
rcp40.5	40	15	99.25	2.250	4.939	4.939	0	10.531
rcp50.1	50	17	99.94	9.156	7.687	7.687	0	15.062
rcp50.2	50	17	98.65	6.656	7.364	7.364	0	43.485
rcp50.3	50	17	100.00	2.578	6.852	6.852	0	2.578
rcp50.4	50	17	98.67	6.860	7.923	7.923	0	34.094
rcp50.5	50	17	97.09	4.468	8.073	8.073	0	111.750
rcp50.1	50	18	99.06	5.281	6.777	6.777	0	29.968
rcp50.2	50	18	99.26	6.438	6.678	6.678	0	23.547
rcp50.3	50	18	98.62	2.890	5.529	5.529	0	25.828
rcp50.4	50	18	100.00	5.485	6.812	6.812	0	5.485
rcp50.5	50	18	99.68	14.781	6.466	6.466	0	32.422
rcp60.1	60	20	99.38	9.313	9.123	9.123	0	94.375
rcp60.2	60	20	98.09	15.656	8.910	8.910	0	157.093
rcp60.3	60	20	99.18	6.485	8.485	8.485	0	32.047
rcp60.4	60	20	98.89	6.141	7.810	7.810	0	95.578
rcp60.5	60	20	99.21	5.703	8.406	8.406	0	34.968
rcp60.1	60	21	100.00	5.250	7.112	7.112	0	5.250
rcp60.2	60	21	99.33	4.859	8.550	8.550	0	104.437
rcp60.3	60	21	97.63	7.046	8.109	8.109	0	466.968
rcp60.4	60	21	100.00	10.594	7.725	7.725	0	10.594
rcp60.5	60	21	98.54	6.546	7.770	7.770	0	96.046
rcp70.1	70	24	99.39	10.641	9.106	9.106	0	94.032
rcp70.2	70	24	98.03	21.516	7.753	7.753	0	748.328
rcp70.3	70	24	100.00	23.328	8.001	8.001	0	23.328
rcp70.4	70	24	98.40	12.515	8.823	8.823	0	235.468
rcp70.5	70	24	98.78	8.250	7.976	7.976	0	319.968
rcp70.1	70	25	99.44	36.750	6.743	6.743	0	93.891
rcp70.2	70	25	100.00	6.687	6.973	6.973	0	6.687
rcp70.3	70	25	98.57	16.516	6.874	6.874	0	160.828
rcp70.4	70	25	98.71	6.672	7.963	7.963	0	219.610
rcp70.5	70	25	98.65	27.750	7.853	7.853	0	265.953

Table 1: Results for the first set of instances

Instance	n	k	Branch-and-price					
			root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
rcp80.1	80	27	98.16	63.031	10.078	10.078	0	1745.484
rcp80.2	80	27	98.37	31.984	10.260	10.260	0	1498.781
rcp80.3	80	27	98.50	41.390	9.126	9.126	0	621.875
rcp80.4	80	27	98.61	89.281	9.233	9.233	0	1347.063
rcp80.5	80	27	97.84	63.953	10.143	10.143	0	4259.890
rcp80.1	80	28	99.32	36.938	7.769	7.769	0	215.328
rcp80.2	80	28	99.27	56.235	9.272	9.272	0	365.844
rcp80.3	80	28	98.70	38.906	8.564	8.564	0	482.000
rcp80.4	80	28	99.71	15.593	7.621	7.621	0	130.281
rcp80.5	80	28	99.45	50.125	7.404	7.404	0	246.735
rcp90.1	90	30	98.03	67.625	10.785	10.795	0.09	
rcp90.2	90	30	98.54	49.313	10.316	10.316	0	1208.860
rcp90.3	90	30	98.24	93.844	10.530	10.530	0	2641.531
rcp90.4	90	30	98.99	48.891	10.113	10.113	0	2134.297
rcp90.5	90	30	98.63	60.796	9.481	9.481	0	2252.515
rcp90.1	90	31	98.39	80.141	9.263	9.263	0	4183.656
rcp90.2	90	31	98.92	84.344	8.924	8.924	0	1695.750
rcp90.3	90	31	98.91	80.656	8.966	8.966	0	540.469
rcp90.4	90	31	98.50	85.359	8.949	8.949	0	1598.468
rcp90.5	90	31	98.39	49.813	7.667	7.667	0	1493.953
rcp100.1	100	34	99.07	70.094	9.758	9.758	0	3486.766
rcp100.2	100	34	99.73	134.625	8.990	8.990	0	289.125
rcp100.3	100	34	98.54	166.250	9.500	9.500	0	6331.562
rcp100.4	100	34	97.95	61.813	10.000	-	-	
rcp100.5	100	34	98.70	184.359	9.858	9.858	0	1806.359
rcp100.1	100	35	98.69	117.641	8.479	8.479	0	3356.438
rcp100.2	100	35	98.77	123.907	9.897	9.897	0	2772.016
rcp100.3	100	35	98.45	122.140	8.805	8.805	0	4105.922
rcp100.4	100	35	97.99	114.469	8.210	8.219	0.10	
rcp100.5	100	35	98.11	177.172	8.105	8.105	0	5854.172
rcp110.1	110	37	98.10	165.422	10.188	-	-	
rcp110.2	110	37	98.80	160.172	9.816	9.820	0.04	
rcp110.3	110	37	99.11	169.610	10.619	10.619	0	2923.079
rcp110.4	110	37	98.94	100.359	10.247	10.247	0	3103.625
rcp110.5	110	37	99.12	250.547	10.365	10.365	0	2248.141

Table 1: Results for the first set of instances

Instance	n	k	Branch-and-price					
			root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
rcp110.1	110	38	98.53	168.797	9.260	9.260	0	7040.312
rcp110.2	110	38	99.96	402.703	9.533	9.533	0	551.609
rcp110.3	110	38	98.46	334.219	9.079	-	-	
rcp110.4	110	38	98.29	401.641	8.285	-	-	
rcp110.5	110	38	98.18	154.610	9.614	9.649	0.37	
rcp120.1	120	40	98.82	404.329	10.343	10.343	0	5544.297
rcp120.2	120	40	98.59	374.219	11.078	-	-	
rcp120.3	120	40	98.88	250.968	10.819	10.819	0	5727.906
rcp120.4	120	40	99.39	374.922	10.886	10.886	0	2490.188
rcp120.5	120	40	98.85	380.547	11.933	11.933	0	4553.188
rcp120.1	120	41	99.13	338.016	10.184	10.184	0	4780.656
rcp120.2	120	41	98.75	236.969	10.661	-	-	
rcp120.3	120	41	98.99	327.704	9.430	-	-	
rcp120.4	120	41	99.76	221.188	9.387	9.387	0	1431.954
rcp120.5	120	41	98.74	256.562	9.700	-	-	
rcp130.1	130	44	98.96	448.593	11.599	-	-	
rcp130.2	130	44	98.88	627.375	10.883	-	-	
rcp130.3	130	44	98.68	187.578	11.122	-	-	
rcp130.4	130	44	99.24	482.656	10.863	-	-	
rcp130.5	130	44	99.13	697.593	9.876	9.896	0.21	
rcp130.1	130	45	99.06	559.625	9.767	-	-	
rcp130.2	130	45	99.13	670.594	9.686	9.686	0	5556.641
rcp130.3	130	45	99.12	345.063	10.133	-	-	
rcp130.4	130	45	99.41	841.032	9.370	9.370	0	3060.079
rcp130.5	130	45	98.74	441.235	9.415	-	-	
rcp140.1	140	47	99.40	1078.954	11.643	-	-	
rcp140.2	140	47	99.20	421.343	11.357	-	-	
rcp140.3	140	47	99.17	622.375	12.046	-	-	
rcp140.4	140	47	98.99	740.860	11.138	-	-	
rcp140.5	140	47	99.21	223.734	10.497	-	-	
rcp140.1	140	48	98.98	752.438	9.549	-	-	
rcp140.2	140	48	99.17	1016.407	10.637	-	-	
rcp140.3	140	48	98.95	296.750	10.526	-	-	
rcp140.4	140	48	99.20	503.828	10.344	-	-	
rcp140.5	140	48	98.70	215.328	9.346	9.356	0.10	

Table 1: Results for the first set of instances

Instance	n	k	Branch-and-price					
			root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
rcp150.1	150	50	99.43	560.875	11.387	-	-	
rcp150.2	150	50	99.11	888.672	9.975	-	-	
rcp150.3	150	50	99.28	813.844	10.436	-	-	
rcp150.4	150	50	99.52	1368.454	12.577	-	-	
rcp150.5	150	50	99.30	1313.641	11.235	-	-	
rcp150.1	150	51	99.29	534.063	11.238	-	-	
rcp150.2	150	51	99.43	1152.109	11.556	-	-	
rcp150.3	150	51	99.12	578.922	10.626	-	-	
rcp150.4	150	51	98.97	967.766	11.149	-	-	
rcp150.5	150	51	99.52	967.688	10.593	-	-	

Table 2: Results for the second set of instances

Instance	n	Branch-and-price with $k = \lceil \frac{3}{2}\bar{k} \rceil$							Branch-and-price with $k = \lceil 2\bar{k} \rceil$						
		k	root node		final solution				k	root node		final solution			
			$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t		$\underline{z}^*(\%)$	t	$\underline{z}^*$	$\overline{z}^*$	gap	t
myciel3	11	6	100.00	0.156	110	110	0	0.156	8	100.00	0.031	28	28	0	0.031
myciel4	23	8	100.00	2.078	2600	2600	0	2.078	10	100.00	0.843	1648	1648	0	0.843
queen5_5	25	8	99.99	1.531	3850	3850	0	2.984	10	100.00	0.985	2521	2521	0	0.985
1-FullIns_3	30	6	100.00	20.515	13292	13292	0	100.656	8	100.00	7.046	8787	8787	0	95.375
queen6_6	36	11	100.00	2.281	12061	12061	0	4.156	14	100.00	3.281	7741	7741	0	3.281
2-Insertions_3	37	6	100.00	211.343	32397	-	-		8	100.00	35.953	22101	22101	0	2075.906
myciel5	47	9	100.00	375.437	52829	-	-		12	100.00	146.250	35156	35156	0	4066.594
queen7_7	49	11	100.00	33.891	48002	48002	0	2785.547	14	100.00	13.719	33383	33383	0	98.500
2-FullIns_3	52	8	-	-	-	-	-	-	10	100.00	595.421	70830	-	-	
3-Insertions_3	56	6	-	-	-	-	-	-	8	-	-	-	-	-	-
queen8_8	64	14	100.00	79.688	109796	-	-		18	100.00	39.171	75460	-	-	
1-Insertions_4	67	8	-	-	-	-	-	-	10	-	-	-	-	-	-
huck	74	17	-	-	-	-	-	-	22	100.00	1834.250	106103	-	-	
4-Insertions_3	79	6	-	-	-	-	-	-	8	-	-	-	-	-	-
3-FullIns_3	80	9	-	-	-	-	-	-	12	-	-	-	-	-	-
jean	80	15	-	-	-	-	-	-	20	-	-	-	-	-	-
queen9_9	81	15	100.00	616.250	277416	-	-		20	100.00	293.672	185506	-	-	
david	87	17	-	-	-	-	-	-	22	-	-	-	-	-	-
mug88_1	88	6	-	-	-	-	-	-	8	-	-	-	-	-	-
mug88_25	88	6	-	-	-	-	-	-	8	-	-	-	-	-	-
1-FullIns_4	93	8	-	-	-	-	-	-	10	-	-	-	-	-	-
myciel6	95	11	-	-	-	-	-	-	14	-	-	-	-	-	-
queen8_12	96	18	-	-	-	-	-	-	24	100.00	894.188	301864	-	-	
mug100_1	100	6	-	-	-	-	-	-	8	-	-	-	-	-	-
mug100_25	100	6	-	-	-	-	-	-	8	-	-	-	-	-	-
queen10_10	100	17	-	-	-	-	-	-	22	100.00	2469.906	410430	-	-	
4-FullIns_3	114	11	-	-	-	-	-	-	14	-	-	-	-	-	-
games120	120	14	-	-	-	-	-	-	18	-	-	-	-	-	-
queen11_11	121	17	-	-	-	-	-	-	22	-	-	-	-	-	-
DSJC125.1	125	8	-	-	-	-	-	-	10	-	-	-	-	-	-
DSJC125.5	125	26	100.00	6848.516	863172	-	-		34	100.00	1540.734	582561	-	-	
DSJC125.9	125	66	99.98	54.250	142721	142721	0	212.781	88	99.99	57.860	36351	36351	0	114.157
miles1000	128	63	100.00	2597.281	187394	187394	0	2597.281	84	100.00	1068.438	58768	58768	0	1068.438
miles1500	128	110	100.00	46.484	4284	4284	0	46.484							
miles250	128	12	-	-	-	-	-	-	16	-	-	-	-	-	-
miles500	128	30	-	-	-	-	-	-	40	-	-	-	-	-	-
miles750	128	47	-	-	-	-	-	-	62	-	-	-	-	-	-
anna	138	17	-	-	-	-	-	-	22	-	-	-	-	-	-
queen12_12	144	18	-	-	-	-	-	-	24	-	-	-	-	-	-
2-Insertions_4	149	6	-	-	-	-	-	-	8	-	-	-	-	-	-

References

[1] Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh M.W.F., Vance P.H. (1998), Branch-and-Price. Column Generation for Solving Huge Integer Programs. *Operations Research* 46, 316-329.

[2] Desaulniers, G., Desrosiers, J., Solomon, M. (2005), Column Generation, Springer.

[3] Friden, C., Hertz, A., de Werra, D. (1989), Stabulus: a technique for finding stable sets in large graphs with tabu search, *Computing* 42, 35-44.

[4] Galinier, P., Hertz, A., Zufferey, N. (2008), An Adaptive Memory Algorithm for the k-Colouring Problem, *Discrete Applied Mathematics* 156, 267-279.

[5] Gendreau, M., Soriano, P., Salvail, L. (1993), Solving the maximum clique problem using a tabu search approach, *Annals of Operations Research* 41, 385-403.

[6] Guo, S., Kong, Y., Lim, A., Wang, F. (2004), A neighborhood based on improvement graph for robust graph coloring problem, *Lecture Notes in Artificial Intelligence*, G.I. Webb and Xinghuo Yu (Eds.) vol. 3339, pp. 636-645, Springer-Verlag Berlin Heidelberg.

[7] Johnson, D.S., Trick, M.A. (1996), Cliques, Coloring, and Satisfiability, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 26, pp. 653-657, American Mathematical Society.

[8] Kong, Y., Wang, F., Lim, A., Guo, S. (2003), A new hybrid genetic algorithm for the robust graph coloring problem, *Lecture Notes in Artificial Intelligence*, T. D. Gedeon and L. C. C. Fung, Eds., vol. 2903. pp. 125-136, Springer-Verlag Berlin Heidelberg.

[9] Lasdon, L. S. (1970), Optimization Theory for Large Systems, Macmillan Company, New York.

[10] Lim, A., Wang, F. (2004), Metaheuristics for Robust Graph Coloring Problem, Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 514-518.

[11] Mehrotra, A., Trick, M.A. (1996), A column generation approach for graph coloring, *INFORMS Journal on Computing* 8, 344-354.

[12] Ramírez-Rodríguez, J. (2001), Extensiones del Problema de Coloracíon de Grafos, *Doctoral dissertation*, Universidad Complutense de Madrid, Madrid, Spain.

[13] Ryan, D.M., Foster, B.A. (2003), On the capacitated vehicle routing problem, *Mathematical Programmming* 94, 343-359.

[14] Yáñez, J., Ramírez, J. (2003), The robust coloring problem, *European Journal of Operational Research* 148, 546-558.