Annals of Operations Research
https://doi.org/10.1007/s10479-018-2977-x

S.I.: CLAIO 2016

An exact algorithm for the edge coloring by total labeling problem

Fabrizio Borghini[1] · **Isabel Méndez-Díaz**[2] · **Paula Zabala**[2]

Abstract
This paper addresses the edge coloring by total labeling graph problem. This is a labeling of the vertices and edges of a graph such that the weights (colors) of the edges, defined by the sum of its label and the labels of its two endpoints, determine a proper edge coloring of the graph. We propose two integer programming formulations and derive valid inequalities which are added as cutting planes on a Branch-and-Cut framework. In order to improve the efficiency of the algorithm, we also develop initial and primal heuristics. The algorithm is tested on random instances and the computational results show that it is very effective in comparison with CPLEX. It is displayed that it reduces both the CPU time (for solved instances) and the final percentage gap (for unsolved instances), and that it is capable of solving instances that are out of the reach of CPLEX.

Keywords Total labeling · Edge coloring · Graph coloring · Branch-and-Cut

Mathematics Subject Classification 05C15 · 90C57 · 90C10

1 Introduction

A labeling of a graph is an assignment of labels (usually integer numbers) to graph elements. The most common choices of graph elements are the set of all vertices (vertex labelings), the edge set (edge labelings) or the set of all vertices and edges (total labelings).

This research was partially supported by UBACYT Grant 2014-2017 20020130100467BA.

✉ Paula Zabala
pzabala@dc.uba.ar

Fabrizio Borghini
fabriborghini@gmail.com

Isabel Méndez-Díaz
imendez@dc.uba.ar

[1] Departamento de Computación, FCEN, Universidad de Buenos Aires, Buenos Aires, Argentina

[2] Departamento de Computación, FCEN, Instituto de Investigación en Ciencias de la Computación (ICC), CONICET-UBA , Universidad de Buenos Aires, Buenos Aires, Argentina

Published online: 25 July 2018

Typically, problems associated with labelings look for the smallest number of labels subject to some constraints which are required to be satisfied for the labels. For example, the most popular problems are the well-known vertex coloring problem (Jensen and Toth 1995), where a vertex labeling must not assign the same label to adjacent vertices, and the proper edge coloring problem (Jensen and Toth 1995), where no two adjacent edges share the same label.

The edge coloring by total labeling problem (ECTLP) is a mixing of the classic vertex coloring and the edge coloring problem that was introduced and studied by Brandt et al. (2010). Formally, given $G = (V, E)$ a simple graph with vertex set $V = \{1, \ldots, n\}$ and edge set $E \subseteq \{uv : u, v \in V, u \neq v\}$, an edge coloring total k-labeling is a mapping $f : V \cup E \to \{1, \ldots, k\}$ such that the weights (colors) of the edges defined by $w(uv) := f(u) + f(uv) + f(v)$ define a proper edge coloring of G. The smallest integer k for which there exists an edge coloring total k-labeling for a graph G is the edge coloring total labeling chromatic number of G, denoted by $\chi_t'(G)$.

The ECTLP is a relaxation of the edge-irregular total labeling problem introduced in Bača et al. (2007), where all edge colors are required to be different.

There are relatively very few works in the literature devoted to ECTLP, all of them concentrate on proposing lower and upper bounds of the chromatic number or study the problem on special classes of graphs. All the bounds are expressed in terms of Δ, the maximum degree of G. If $N(v)$ is the set of vertices adjacent to v, then $d(v) = |N(v)|$ is the vertex degree of v and Δ is the largest vertex degree of G.

It is straightforward to compute an upper bound on $\chi_t'(G)$ by considering a proper edge coloring of G alongside a constant vertex labeling (i.e. $f(v) = 1\ \forall v \in V$). These labels define an edge-coloring total labeling of G. Therefore, $\Delta + 1$ represents an upper bound resulting directly from Vizing's theorem (Misra and Gries 1992) for edge coloring.

Nevertheless, this bound can be improved. Consider $f : E \to \{1, \ldots, k\}$ a proper edge coloring of G with $\Delta + 1$ labels. V is partitioned in two sets, A and B, where all edges labeled with Δ and $\Delta + 1$ have one endpoint in A and the other endpoint in B. This is always possible as the subgraph formed by the edges labeled with Δ or $\Delta + 1$ is bipartite. Then, label 1 is assigned to all vertices in A and label Δ to all vertices in B. Labels $f(e)$ of the edges between vertices in A remain unchanged and the edges between vertices in B are labeled with $f(e) + 1$. Finally, the edges between vertices in A and B might be assigned Δ labels at the most. It can be easily observed that by assigning these labels to vertices and edges, an edge coloring by total labeling is defined.

On the other hand, if there is an edge-coloring total k-labeling of G, the possible color values of the edges incident to vertex v are $\{f(v) + 2, f(v) + 3, \ldots, f(v) + 2k\}$. If v has maximum degree, these labels should be enough to label all edges incident to v; therefore, $2k - 1$ must be at least Δ.

Based on these observations, in Brandt et al. (2010) present lower and upper bounds in terms of Δ, the maximum degree of G:

$$\left\lceil \frac{\Delta + 1}{2} \right\rceil \leq \chi_t'(G) \leq \Delta$$

In addition to this, in Brandt et al. (2010) different classes of graphs are analyzed and their bounds were tightened. For example, if F is a forest then the general lower bound is tight, i.e. $\chi_t'(F) = \lceil \frac{\Delta+1}{2} \rceil$.

Moreover, for the complete graph K_n, if $n \not\equiv 2 \pmod 4$, then $\chi_t'(K_n) = \lceil \frac{n}{2} \rceil$ and if $n \equiv 2 \pmod 4$, then $\chi_t'(K_n) \leq \frac{n}{2} + 1$. Finally, in Khennoufa et al. (2013) the authors determine χ_t' for generalized Petersen graphs and, in Seba and Khennoufa (2013), the chromatic number of 4-regular circulant graphs $C_n(1, k)$ is studied.

To the best of our knowledge, there are no previous works which propose algorithms, neither heuristics nor exact ones.

Given that many coloring problems have been successfully solved by integer linear programming [see for example Aardal et al. (2007), Burke et al. (2012), Coll et al. (2002), Cornaz et al. (2017), Malaguti et al. (2015)], the goal of this paper is to design an exact algorithm based on this approach.

The remainder of the paper is organized as follows. In Sect. 2, we present two models and their computational performances of the CPLEX default Branch-and-Cut algorithm are compared. Some families of valid inequalities are described in Sect. 3. Section 4 is devoted to describing heuristic algorithms for the problem. In Sect. 5, we show computational evidence which reflects the improvement in performance when the Branch-and-Cut algorithm uses valid inequalities as cutting planes as well as initial and primal heuristics. The paper closes with final remarks.

2 Integer programming models

It is well known that integer programming models give a natural tool to formulate coloring problems. As stated in the introduction, to the best of our knowledge no integer programming approach has been derived so far for ECTLP. Therefore, in this section we propose two integer programming formulations for the problem. The first one is based on general integer variables that represent the label assigned to a vertex or edge, while the second one comes from binary variables to indicate if a label is assigned to a vertex or edge.

2.1 Model 1

This model considers general integer variables $x_v \in \{1, \ldots, \Delta\}$ and $x_{uv} \in \{1, \ldots, \Delta\}$ $\forall uv \in E$, which represent the label assigned to each vertex and each edge, respectively. Notice that labels are restricted to $\{1, \ldots, \Delta\}$ due to the upper bound established in Brandt et al. (2010). Moreover, for each $j \in V$ and $u, v \in N(j)$ (set of adjacent vertices to j), with $u < v$ [1], let δ_{ujv} be a binary variable that values 1 if $w(uj) < w(jv)$ and 0 if $w(uj) > w(jv)$. Finally, let $z \in \{1, \ldots, \Delta\}$ be a general integer variable that values the maximum label used in a total labeling. The goal is to minimize z to obtain $\chi'_t(G)$.

Based on these variables, the following is an integer programming model:

$$\begin{aligned}
&\text{minimize } z \\
&\text{subject to} \\
&x_u \leq z && \forall u \in V && (1)\\
&x_{uv} \leq z && \forall uv \in E && (2)\\
&x_u + x_{uj} - x_{jv} - x_v \geq 1 - \delta_{ujv}(2\Delta - 1) && \forall j \in V, u, v \in N(j), u < v && (3)\\
&x_u + x_{uj} - x_{jv} - x_v \leq -1 + (1 - \delta_{ujv})(2\Delta - 1) && \forall j \in V, u, v \in N(j), u < v &&\\
&1 \leq x_u, x_{uv}, z \leq \Delta, \text{ with } x_u, x_{uv}, z \in \mathbb{Z} && \forall u \in V, uv \in E &&\\
&\delta_{ujv} \in \{0, 1\} && \forall j \in V, u, v \in N(j), u < v && (4)
\end{aligned}$$

Constraints (1) and (2) impose that every label is a lower bound for z. In this way, since we are minimizing variable z, the optimal value corresponds to $\chi'_t(G)$. Constraints (3) and (4)

[1] Vertices are ordered according to the standard order on the natural numbers.

define an edge coloring, asserting that two edges incident to the same vertex are assigned different colors, i.e. $w(uj) \neq w(jv)$ if $j \in V$ and $u, v \in N(j)$.

An optimal solution of the LP relaxation is given by $x_v^* = 1$ for all $v \in V$, $x_{uv}^* = 1$ for all $uv \in E$, variables δ^* with values $\frac{1}{2\Delta-1} \leq \delta_{ujv}^* \leq \frac{2\Delta-2}{2\Delta-1}$ for all $j \in V, u, v \in N(j)$ with $u < v$ and $z^* = 1$. This shows that the LP relaxation is very weak, which affects the Branch-and-Cut algorithm performance of the state of the art solvers based on this formulation. Strengthening with cutting planes is needed to make the model suitable for this approach.

2.2 Model 2

The second model is based on the multimatching formulation proposed in Lee and Leung (1993) for the edge coloring problem. For each $v \in V$ and $j = 1, \ldots, \Delta$, let us be binary variable x_{vj} that values 1 in case that vertex v is labeled with j. Additionally, variable z_{uvj} that values 1 if edge uv is labeled with j. Otherwise, it takes value 0. Moreover, we consider binary variable w_{uvk} that values 1 if the sum of the labels assigned to u, v and uv is k, i.e. $w(uv) = k$. Finally, the general integer variable y represents $\chi_t'(G)$. The following is an integer programming formulation based on these variables:

$$\text{minimize } y$$

$$\text{subject to}$$

$$\sum_{j=1}^{\Delta} x_{vj} = 1 \qquad \forall v \in V \qquad (5)$$

$$\sum_{j=1}^{\Delta} z_{uvj} = 1 \qquad \forall uv \in E \qquad (6)$$

$$w_{uvk} \geq x_{uj} + z_{uvr} + x_{vp} - 2 \qquad \forall uv \in E, j, r, p \in \{1, \ldots, \Delta\} \text{ with } k = j + r + p \qquad (7)$$

$$\sum_{\forall v \in N(u)} w_{uvk} \leq 1 \qquad \forall u \in V, d(u) > 1, k \in \{3, \ldots, 3\Delta\} \qquad (8)$$

$$y \geq \sum_{j=1}^{\Delta} j \; x_{vj} \qquad \forall v \in V \qquad (9)$$

$$y \geq \sum_{j=1}^{\Delta} j \; z_{uvj} \qquad \forall uv \in E$$

$$x_{vj} \in \{0, 1\} \qquad \forall v \in V, j \in \{1, \ldots, \Delta\}$$

$$z_{uvj}, w_{uvk} \in \{0, 1\} \qquad \forall uv \in E, k \in \{3, \ldots, 3\Delta\}, j \in \{1, \ldots, \Delta\}$$

$$1 \leq y \leq \Delta, \text{ with } y \in \mathbb{Z} \qquad (10)$$

The objective function is to minimize the value of variable y.

Constraints (5) and (6) assure that exactly one label is assigned to each vertex and edge, respectively. Constraints (7) impose value 1 to the color which is the sum of the labels assigned to u, v and uv. Constraints (8) assure that for each vertex, edges incident to it must not be colored with the same color. Finally, constraints (9) and (10) determine that y is greater than

or equal to the label used for every vertex or edge. Since we are minimizing variable y, the optimal value corresponds to $\chi'_t(G)$.

An optimal solution of the LP relaxation is given by

$x^*_{v1} = \frac{2\Delta+1}{3\Delta}$, $x^*_{v2} = \frac{\Delta-1}{3\Delta}$ and $x^*_{vj} = 0$ for all $v \in V$, $j \neq 1, 2$.
$z^*_{uv1} = \frac{2\Delta+1}{3\Delta}$, $z^*_{uv2} = \frac{\Delta-1}{3\Delta}$ and $z^*_{uvl} = 0$ for all $uv \in E$, $j \neq 1, 2$.
$w^*_{uv3} = \frac{1}{\Delta}$, $w^*_{uvk} = 0$ for all $uv \in E$, $k \neq 3$
$y^* = \frac{4\Delta-1}{3\Delta}$

Even though the LP relaxation optimal value of Model 2 is greater than the LP relaxation optimal value of Model 1, the lower bound provided is still very weak. It makes the model very hard to solve with a standard Branch-and-Cut algorithm and it is necessary to customize the algorithm by, for example, including cutting planes.

2.3 Comparisons of the above proposed models

The size of an integer programming model measured in numbers of constraints and variables is not directly related to the time required for a Branch-and-Cut algorithm to solve the problem. However, this is a first insight into the expected computational performance of a model.

In Table 1 we compare the number of variables and constraints of both models.

As Table 1 shows the size of Models 1 and 2 depends not only on the number of vertices (n) and edges (m) but also on the vertex degrees. It is not difficult to see that $\sum_{v \in V} \frac{d(v)(d(v)-1)}{2}$ is at most $m(\Delta-1)$. So, $m(\Delta-1)$ and $n+m+2m(\Delta-1)$ are upper bounds on the number of binary variables and on the number of constraints of Model 1, respectively. Then, even tough Model 1 has more general integer variables than Model 2, it has significantly less constraints and binary variables than Model 2. Hence, we can conclude that Model 1 is theoretically a better model than Model 2.

In order to show some deeper insight into the performance of the models, we run the Branch-and-Cut CPLEX default algorithm. Experiments were carried on an Intel i7 workstation with a CPU running at 3.4 GHz and 16 GB of RAM memory, over random instances of 15 vertices with different density percentages and 1 hour time limit. Each instance of $p\%$ of density is generated by considering a uniform probability p that two vertices are adjacent to each other.

In Table 2 we report on the average results obtained over 30 instances for low (less than 30%), medium (between 40 and 60%) and high (greater than 70%) density considered. Column **#Solved** refers to the number of instances that reach optimality within the time limit and column **Time** to the average run time (in seconds) over solved instances. Column **%Gap** shows the average final gap for unsolved instances. Table 2 shows that the higher the density, the more difficult the instance and the worse the performance of the models. For

Table 1 Model size

	Model 1	Model 2
General integer var	$n+m+1$	1
Binary integer var	$\sum_{v \in V} \frac{d(v)(d(v)-1)}{2}$	$n\Delta+m\Delta+m(3\Delta-2)$
Constraints	$n+m+\sum_{v \in V} d(v)(d(v)-1)$	$3n\Delta+2m+m\Delta^3$

Table 2 Branch-and-Cut CPLEX default

Density	Model 1			Model 2		
	#Solved	Time	%Gap	#Solved	Time	%Gap
Low	29	10.29	35	28	24.09	38.3
Medium	11	788.64	42.47	2	112.64	46.26
High	0	3600	67.58	0	3600	67.27

high density graphs, both models fail in reaching optimality for all instances within the time limit. The final gap is higher than 60%, mainly due to the difficulty to improve the lower bound. Low instances are the easiest and Model 1 has a considerable better performance in terms of both number of solved instances and average time required to reach optimality. For medium density graphs, Model 1 significantly exceeds the quantity of solved instances of Model 2. The average time is greater because it is taken over more instances, but if we restrict ourselves to the instances solved by Model 2, the average time is less than 1s, showing the superiority of Model 1.

In any case, the main message of the results is that a customization of the algorithm is needed to strengthen, among other things, bounds on the Branch-and-Cut enumeration tree. In the next section we concentrate on studying the underlying polytope associated with Model 1 to infer valid inequalities that could be useful as cutting planes to improve the overall performance.

3 Valid inequalities

In this section we focus on deriving families of valid inequalities for Model 1. The first ones correspond to inequalities based on the values of the labels. The second ones are derived considering variables δ that define an ordering among the edges incident to a vertex.

3.1 Lower and upper bounds

Colors are defined by the labels assigned to the vertices and edges and, since these labels take values between 1 and Δ, the color values are bounded below by 3 and above by 3Δ. Furthermore, the colors assigned to the edges incident to a vertex must be different because they must define an edge coloring.

Then, given $v \in V$, a lower and upper bound can be derived for the sum of the labels assigned to the neighborhood of v.

Suppose that $\{vu_1, vu_2, \ldots, vu_{d(v)}\} \subseteq E$ is the set of edges incident to v. As for edge-coloring all these edges should be assigned different colors, the smallest values of colors to be assigned to these edges are $\{3, 4, \ldots, d(v)+2\}$. Nevertheless, as the label in v is present in the colors of each edge, it is equivalent to saying that $\{2, 3, \ldots, d(v)+1\}$ are the smallest values of the sum of the labels in each edge vu_i and its corresponding endpoint u_i. Therefore, the sum of any labeling should necessarily be higher or equal to the sum of this set of values:

$$\sum_{u \in N(v)} x_u + \sum_{u \in N(v)} x_{vu} \geq 2 + 3 + \ldots + d(v) + 1 = \sum_{k=1}^{d(v)} (k+1) \qquad \text{(SumLB)}$$

$$= \frac{d(v)(d(v)+3)}{2}$$

Similarly, an upper bound can be applied to this sum. The following inequality takes into account the highest color values resulting from labeling in the case of edges incident to v. Since in the model the upper bound for vertex and edge labels is Δ, the maximum possible color values are $\{3\Delta, 3\Delta - 1, \ldots, 3\Delta - d(v) + 1\}$. As the label in v is present in all edge colors, it is equivalent to considering $\{2\Delta, 2\Delta, \ldots, 2\Delta - d(v) + 1\}$ as the highest values of the sum of the labels in each edge vu_i and its corresponding endpoint u_i. Therefore, in any case of labeling, the sum of all these labels should necessarily be lower than or equal to the sum of the maximum values:

$$\sum_{u \in N(v)} x_u + \sum_{u \in N(v)} x_{vu} \leq 2\Delta + (2\Delta - 1) + \cdots + (2\Delta - d(v) + 1) \qquad \text{(SumUB)}$$
$$= d(v)(2\Delta + 1) - \frac{d(v)(d(v) + 1)}{2}$$

Moreover, given a vertex v and an incident edge uv, such bounds could be adjusted if we knew how many edges incident to v result with a lower color and how many are assigned an upper color than the color assigned to uv.

The lowest color value is 3, but this lower bound is increased by 1 for each adjacent edge with a lower color value. Variables δ_{rvu} account for this information and they are useful to reflect the increase. Moreover, as the label in v is present in all incident edge color values, the argument can be applied to the sum of the incident edge label and its corresponding endpoint label. Take into account that in this case the lowest value is 2.

According to this, given $v \in V$, $r \in N(v)$, the following are valid inequalities:

$$x_{vr} + x_r \geq 2 + \sum_{\substack{u \in N(v) \\ u < r}} \delta_{uvr} + \sum_{\substack{u \in N(v) \\ r < u}} (1 - \delta_{rvu}) \qquad \text{(ColorLB)}$$

Note that the right-hand member is separated into two sums as a consequence of the definition of variables δ_{rvu} or δ_{uvr}, respectively.

Similarly, an upper bound is provided by considering adjacent edges with upper color values. The maximum color could be 2Δ but it decreases by 1 for each higher edge color value. As a result:

$$x_{vr} + x_r \leq 2\Delta - \sum_{\substack{u \in N(v) \\ u < r}} (1 - \delta_{uvr}) - \sum_{\substack{u \in N(v) \\ r < u}} \delta_{rvu} \qquad \text{(ColorUB)}$$

SumLB (SumUB) inequalities are implied by the addition of ColorLB (ColorUB respectively) inequalities corresponding to vertices adjacent to v.

Given that the number of ColorLB and ColorUB inequalities is $2m$, they were explicitly included in the model. Even when the number of SumLB and SumUB inequalities is n, they were not included in the model as they are implied by previous ones.

The addition of ColorLB and ColorUB inequalities in the model improves the optimal value of the initial LP relaxation. It shall be remembered that 1 is the optimal value irrespective of the instance. Nevertheless, by including these inequalities a better lower bound on the optimal value can be obtained

If v is a maximum degree vertex ($d(v) = \Delta$), then the inequality SumLB establishes that:

$$\sum_{u \in N(v)} x_{vu} + \sum_{u \in N(v)} x_u \geq \frac{\Delta^2 + 3\Delta}{2}$$

By definition, we know that $z \geq x_u, \forall u \in V$ and $z \geq x_{ur}, \forall ur \in E$, therefore

$$\Delta z + \Delta z \geq \sum_{u \in N(v)} x_{vu} + \sum_{u \in N(v)} x_u \geq \frac{\Delta^2 + 3\Delta}{2} \Longrightarrow$$
$$2\Delta z \geq \frac{\Delta(\Delta + 3)}{2} \Longrightarrow$$
$$z \geq \frac{\Delta + 3}{4}$$

This shows that the optimal value of the LP relaxation when SumLB o ColorLB inequalities are included will be at least $\frac{\Delta+3}{4}$.

It is worth mentioning that according with our computational experiments, the optimal LP relaxation achieves this lower bound in all cases.

Along the same lines, given two edges vu_1 and vu_2 that are incident to vertex v, the difference between their colors can be bounded below by considering the colors assigned to other edges incident to v. If S is a set of edges incident to v with color values between the colors assigned to vu_1 and to vu_2, then this difference is at least $|S| + 1$.

Formally, given $v \in V$, $u_1, u_2 \in N(v)$, $S \subset N(v) \setminus \{u_1, u_2\}$, the following is a valid inequality:

$$\begin{aligned}(x_{vu_1} + x_{u_1}) - (x_{vu_2} + x_{u_2}) &\geq |S| + 1 \\ &- (2\Delta + |S| - 1)\Big(\sum_{\substack{u \in S \\ u_1 < u}} \delta_{u_1 vu} + \sum_{\substack{u \in S \\ u < u_1}} (1 - \delta_{uvu_1})\Big) \\ &- (2\Delta + |S| - 1)\Big(\sum_{\substack{u \in S \\ u_2 < u}} (1 - \delta_{u_2 vu}) + \sum_{\substack{u \in S \\ u < u_2}} \delta_{uvu_2}\Big)\end{aligned} \tag{DifLB}$$

3.2 Cycles

We present below two inequality families that results from analyzing a lower bound on edge coloring in a cycle.

Consider a cycle $C = \{v_1v_2, v_2v_3, \ldots, v_{l-1}v_l, v_lv_1\}$. Any total labeling must define an edge coloring. It is well known that the minimum number of colors needed for an edge coloring is 2 in case l is even and 3 if l is odd. In the first case, it is achieved by alternating two colors along the cycle. In the second case, the procedure is also to alternate two colors except that an edge shall be assigned another color.

As we noted before, the colors resulting from any labeling (sum of edge and vertex labels) are greater or equal to 3. Then, the edge coloring that uses color 3 and 4 if l is even and 3, 4 and 5 if l is odd, brings a lower bound on the sum of color values used along a cycle.

This lower bound is $3\lfloor l/2 \rfloor + 4\lfloor l/2 \rfloor$ and $3\lfloor l/2 \rfloor + 4\lfloor l/2 \rfloor + 5$, respectively.

Based on this, the following are valid inequalities

$$\sum_{ur \in C} x_{ur} + 2\sum_{i=1}^{l} x_{v_i} \geq (3 + 4)\frac{l}{2} \qquad l \text{ even} \tag{CycEven}$$

$$\sum_{ur \in C} x_{ur} + 2\sum_{i=1}^{l} x_{v_i} \geq (3 + 4)\frac{l - 1}{2} + 5 \qquad l \text{ odd} \tag{CycOdd}$$

It is worth mentioning that the optimal LP relaxation value when these inequalities are included increases from 1 to $\frac{4}{3}$.

3.3 Linear ordering inequalities

Given a vertex v, variables δ_{uvr} define a linear ordering among its incident edges. Therefore, some inequalities proposed for this problem can be adapted to ECTLP.

The linear ordering problem is a well known problem which was deeply studied in the literature. In particular, in Grötschel et al. (1984) many valid inequality families were proposed. We consider two of them which, according to the reported computational experience, have had a good performance as cutting planes. The first one is the 3-dicycle inequality which prevents solutions that contain cycles.

Given $v, u, r, q \in V$ such that u, r, q are adjacent to v ($u < r < q$), the following inequalities are valid:

$$\delta_{uvr} + \delta_{rvq} \leq 1 + \delta_{uvq} \tag{3-dicycle-1}$$

$$\delta_{uvr} + \delta_{rvq} \geq \delta_{uvq} \tag{3-dicycle-2}$$

This indicates that if the color value in edge uv is lower than that in edge rv and the color value in edge rv is lower than that in qv, then the color value in uv is lower than that in qv.

The second family corresponds to *k-fence* inequalities. This type of inequalities was studied by Grötschel et al. (1984, 1985) and characterized as facet-defining for the polytope associated to the problem.

Given $v \in V$ and $U = \{u_1, \ldots, u_k\}$, $L = \{l_1, \ldots, l_k\} \subseteq N(v)$, $3 \leq k \leq \frac{|N(v)|}{2}$, $U \cap L = \emptyset$. The following is a valid inequality:

$$\sum_{\substack{i=1 \\ u_i < l_i}}^{k} \delta_{u_i v l_i} + \sum_{\substack{i=1 \\ u_i > l_i}}^{k} (1 - \delta_{l_i v u_i}) + \sum_{i=1}^{k} \left(\sum_{\substack{j=1 \\ j \neq i \\ l_i < u_j}}^{k} \delta_{l_i v u_j} + \sum_{\substack{j=1 \\ j \neq i \\ u_j < l_i}}^{k} (1 - \delta_{u_j v l_i}) \right) \leq k^2 - k + 1 \tag{Fence}$$

Due to the fact that obtaining this type of inequalities is highly difficult, and based on the computational results reported by Grotschel in [8], only $3 - fence$ inequalities are tested in the cutting planes algorithm.

4 Heuristics

4.1 Initial heuristics

Setting up an incumbent solution is the first step in a Branch-and-Cut algorithm. This solution establishes an upper bound for $\chi'_t(G)$ from the very beginning, with the objective of reducing the number of nodes in the search tree. Moreover, it makes it possible to reduce the number of variables and constraints in the model.

First, we describe a constructive algorithm to obtain an edge coloring total Δ-labeling. Then, we propose two heuristic approaches which aim to reduce the upper bound mentioned above.

4.1.1 Constructive algorithm

As we described in the introduction, a constructive proof of $\chi_t'(G) \leq \Delta$ is given in Brandt et al. (2010). Following this proof we implement a procedure that labels vertices and edges using Δ as the maximum value of the labeling.

The sketch of the procedure is shown in Algorithm 1.

Algorithm 1 Constructive algorithm

Require: $G = (V, E)$,
Step 1: Following the proposal in Misra and Gries (1992), a proper edge coloring $c : E \to \{1, 2, \ldots, \Delta+1\}$ of G is obtained.
Step 2: The subgraph determined by the edges with color Δ or $\Delta + 1$ is bipartite. Let us consider a partition of V, $A \cup B = V$, such that for all uv colored with Δ or $\Delta + 1$, then $u \in A$ and $v \in B$ (or vice versa).
Step 3: Assign label 1 to all $v \in A$ and label Δ to all $v \in B$.
Step 4: Assign label $c(e)$ to all $e = uv$ such that $u, v \in A$ and $c(e) + 1$ to all $e = uv$ such that $u, v \in B$.
Step 5: Labels assigned to the edges $e = uv$ with $u \in A$ and $v \in B$ are defined by a proper edge coloring. Since the subgraph determined by these edges is bipartite, we can assert that any edge coloring uses at most Δ colors.
Step 6: Color the edges by assigning the sum of the labels of its endpoints and its own label.

At the end, all edges and vertices are assigned labels less than or equal to Δ. The coloring of the edges satisfies that the edges with both endpoints in A are assigned colors between 3 and $\Delta + 1$, the edges with both endpoints in B are assigned colors between $\Delta + 2$ and 3Δ and other edges are colored with color values between $\Delta + 2$ and $2\Delta + 1$. It is easy to check that this is a proper edge coloring by total Δ-labeling.

4.2 Heuristic based on vertex coloring (HVC)

This heuristic is a two phase procedure. In the first phase a vertex labeling is found. Then, in the second phase, using the vertex labeling obtained as a starting point, labels to edges are assigned bearing in mind that the total labeling must define an edge coloring.

During the first phase, the classical sequential heuristic for vertex coloring is applied (Brelaz 1979). The algorithm takes each vertex in turn following a specific order and assigns the first color available, creating a new color when necessary. The choice of which vertex to color next is decided based on the maximal saturation degree, which is defined as the number of different colors already assigned to its adjacent vertices.

Since a solution with Δ labels is found with the constructive heuristic, HVC attempts to look for a solution using at most $\Delta - 1$ labels. Then if $\Delta - 1$ colors are not enough to color all vertices, the algorithm fails and does not return any solution.

If the first phase was successful in finding a vertex coloring, then the second phase attempts to label the edges following a greedy approach. By taking the edges in turn (in arbitrary order), to label edge e, the sum of the labels of its endpoints is obtained. Then, for each edge adjacent to e which has already been labeled, the difference between its label and this sum is obtained. These values determine the forbidden labels to edge e. The procedure chooses the smallest allowable label to assign to e. Once again, if $\Delta - 1$ labels are not enough to label the edges, the algorithm fails and does not return any solution.

The outline of the procedure is presented in Algorithm 2.

Algorithm 2 Heuristic based on vertex coloring

Require: $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$, $E = \{e_1, \ldots, e_m\}$
  $MaxLabelUsed = 1$
  FIRST PHASE: Vertex labeling
  $vertexLabel(v_1) = 1$
  for $j = 2$ **to** n **do**
    $v_j = nextVertextoColor$
    $vertexLabel(v_j) = min\{l : l \geq 1 \text{ and } vertexLabel(v_i) \neq l, \forall v_j v_i \in E\}$
    if $vertexLabel(v_j) \geq \Delta$ **then**
      return FAIL
    end if
    $MaxLabelUsed = max(MaxLabelUsed, vertexLabel(v_j))$
  end for

  SECOND PHASE: Edge labeling
  $edgeLabel(e_1) = 1$
  for $j = 2$ **to** m **do**
    $e_j = (u_j, v_j)$
    $p = vertexLabel(u_j) + vertexLabel(v_j)$
    $W = \{w : w = vertexLabel(s_i) + edgeLabel(e_i) + vertexLabel(u_j),$
      $\forall e_i = s_i u_j \in E, 1 \leq i \leq j-1\} \bigcup$
      $\{w : w = vertexLabel(v_j) + edgeLabel(e_i) + vertexLabel(r_i),$
      $\forall e_i = v_j r_i \in E, 1 \leq i \leq j-1\}$
    $edgeLabel(e_j) = min\{l : l \geq 1 \text{ and } l \neq w - p, \forall w \in W\}$
    if $edgeLabel(e_j) \geq \Delta$ **then**
      return FAIL
    end if
    $MaxLabelUsed = max(MaxLabelUsed, edgeLabel(e_j))$
  end for
  return $MaxLabelUsed$, $vertexLabel$ and $edgeLabel$

4.3 Heuristic based on an auxiliary graph (HAG)

When analyzing the HCV procedure and the experimental computational results, we observe that if two non adjacent vertices with a common neighbour are assigned the same color, then two different labels must be assigned to the adjacent edges. This increases the chances that the algorithm will fail and it would be more suitable if these vertices were assigned different colors.

In order to build a vertex labeling with this characteristic, the first step will look for a vertex labeling for an auxiliary graph X instead of finding a vertex coloring for G. Graph X is defined with the same set of vertices V, but there exists an edge between two vertices u and v if and only if there is a vertex j in V such that uj and $vj \in E$. Then, let $X = (V(X), E(X))$ be an auxiliary graph where $V(X) = V$ and $vj, uj \in E \iff uv \in E(X)$. In this way, the vertex labeling will not force phase 2 to assign different labels to adjacent edges.

Moreover, the first step of HVC looks for a vertex coloring with at most $\Delta - 1$ colors by using a greedy procedure. The algorithm could fail either because the chromatic number is greater than $\Delta - 1$ or because the greedy process gets stuck in the color assignment. In any case, the algorithm does not provide a solution. Actually, it is not a necessary condition for an edge coloring by total labeling to induce a vertex coloring; it would be useful to relax this condition and let the first phase continue even though the label assignment violates the neighborhood restriction.

Then, in case that the procedure finds infeasible to assign a label (among the available ones), it is allowed to assign an used label (*wildcard* color) in spite of being shared by a neighbour. In this way, we manage failures and increase the chances to achieve a total labeling. The *wildcard* colors are assigned in such a way that they are chosen evenly among the colors available, i.e. the sizes of the wildcard color classes differ in at most one. The goal is to avoid having many color repetitions and to prevent the second phase from being forced to need more diversity of colors.

Additionally, instead of restricting the first phase to $\Delta - 1$ labels and with the purpose of expanding the search space, we make an iterative procedure. Since $\left\lceil \frac{\Delta+1}{2} \right\rceil$ is a lower bound of $\chi_t'(G)$, we start looking for a solution using it as the maximum value of a labeling ($MaxLabel$). If the algorithm fails, $MaxLabel$ is increased by one and the procedure is repeated. We perform several trials such that in trial T, the procedure considers $MaxLabel = \left\lceil \frac{\Delta+1}{2} \right\rceil + T$.

Since we know that $\left\lceil \frac{\Delta+1}{2} \right\rceil \leq \chi_t'(G) \leq \Delta$, then it is reasonable to range T between 0 and $\Delta - 1 - \left\lceil \frac{\Delta+1}{2} \right\rceil$. However, preliminary computational experiments show that in most of the instances, the best solutions achieved by the algorithm are obtained when T ranges between $\left\lceil \frac{\Delta}{8} \right\rceil$ and $\left\lceil \frac{3\Delta}{8} \right\rceil$. Then, the algorithm limits T to these bounds to avoid spending time by exploring unlikely useful solutions. This scheme is shown in Algorithm 3.

In order to evaluate the quality of the solutions found by the heuristics, we experiment with graphs of 100 vertices, grouped by density in sets of 15 instances. In Table 3 we report the average results over 1000 independent executions with different random edge order.

Column **Time** refers to the average run time (in seconds), **AveSuccess** and σ-**Success** refer to the mean and standard deviation (in percentage terms) of the percentage of instances that the heuristic could improve the initial Δ-labeling, **AveImprov** and σ-**Improv** refer to the mean and standard deviation (in percentage terms) of the percentage of improvement and **AveGap** and σ-**Gap** refer to the mean and standard deviation (in percentage terms) of the percentage of gap between the solution and the lower bound provided in Brandt et al. (2010). We do not report standard deviation of run time since it is insignificant: the 1000 independent executions take very similar time.

Even though HAG takes more time than HVC, it is significantly more efficient. In all instances, HAG improves the initial solution, whereas HVC shows good performance only for low density graphs and fails many times in medium and high density graphs. Moreover,

Table 3 HCV versus HAG

Density		Time	AveSuccess	σSuccess	AveImprov	σImprov	AveGap	σGap
Low	HVC	0.09	87	7.3	10	1.9	75	1.8
	HAG	0.24	100	0	20	0.8	51	0.6
Medium	HVC	0.43	53	9.2	4	0.6	76	1.2
	HAG	2.11	100	0	24	0.5	48	0.7
High	HVC	1.02	40	5.2	3	0.4	90	0.9
	HAG	7.51	100	0	29	0.4	39	0.7

Algorithm 3 Heuristic based on an auxiliary graph

Require: $G = (V, E)$ and $X = (V, E(X))$
  $BestMaxLabelUsed = \Delta$
  for $T = \left\lceil \frac{3\Delta}{8} \right\rceil$ **to** $\left\lceil \frac{\Delta}{8} \right\rceil$ **do**
    $MaxLabelUsed = 1$
    FIRST PHASE: Vertex labeling of X
    $wildcard = 0$
    $MaxLabel = \left\lceil \frac{\Delta + 1}{2} \right\rceil + T$
    $vertexLabel(v_1) = 1$
    for $j = 2$ **to** n **do**
      $v_j = nextVertextoColor$
      $vertexLabel(v_j) = min\{l : l \geq 1 \text{ and } vertexLabel(v_i) \neq l, \forall v_j v_i \in E(X)\}$
      if $vertexLabel(v_j) > MaxLabel$ **then**
        $vertexLabel(vj) = wildcard + 1$
        $wildcard = (wildcard + 1)\%MaxLabel$
      end if
      $MaxLabelUsed = max(MaxLabelUsed, vertexLabel(v_j))$
    end for
    SECOND PHASE: Edge labeling as in previous algorithm
    if $BestMaxLabelUsed > MaxLabelUsed$ **then**
      $Bestvertexlabel = vertexLabel$
      $BestedgeLabel = edgeLabel$
      $BestMaxLabelUsed = MaxLabelUsed$
    end if
  end for
  if $BestMaxLabelUsed < \Delta$ **then**
    return $MaxLabelUsed$, $BestvertexLabel$ and $BestedgeLabel$
  else
    return FAIL
  end if

the average deviation with respect to the success for HVC is greater than the one for HAG, showing a performance that is more influenced by the order in which the edges are considered. Regarding percentage of improvement and percentage of gap, HAG is considerably better than HVC with larger improvements and lower gaps.

4.4 Primal heuristic

An important aspect in a Branch-and-Cut algorithm is the generation of feasible solutions with the aim of reducing the size of the search tree, bearing in mind that it should be achieved with an inexpensive computational effort.

After solving the LP relaxation at a node, a heuristic procedure can be executed. The procedure tries to use the information in the LP relaxation (fractional) solution aiming to improve the current best solution found so far. The proposed primal heuristic follows the same steps as HAG, but some decisions are made by taking advantage of the information provided by the fractional solution. From now on, we use tag * to refer to the values of the LP relaxation optimal solution.

The primal heuristic procedure is similar to the heuristic based on an auxiliary graph (HAG), but in step 1 of HAG, instead of choosing the minimum feasible label to vertex u, the label is determined by rounding x^*_u to the nearest integer.

Once the vertices have been labeled, the edges are labeled next. During the second phase of HAG, edges are considered in a non-specific order. Even when this order could be relevant, the initial heuristic does not provide us with the necessary information to apply a specific order for labeling.

Nevertheless, when applying primal heuristic, there exist optimal LP relaxation solutions where variable values δ^* *suggest* an order among edge colors. In some way, if for an edge uv, the value $\sum_{ur \in E, r \neq v} \delta^*_{vur}$ is small, then it could be reasonable to think that uv should be assigned a small label. This suggests that according to the information provided by δ variables, an increasing edge order in respect to this sum of δ values should be set in step 2 of HAG.

5 Computational experience

In this section we report on computational experiences with our Branch-and-Cut algorithm in Model 1. The code was implemented in C++ using the CPLEX 12.6 LP solver. We performed the experiments on a computer i7 3.4GHz and 16 GB of RAM memory.

CPU times are reported in seconds and a CPU time limit of 1800s is imposed. The algorithm is tested on random graphs of 15, 20, 30 and 40 vertices, where an edge between each pair of vertices is generated with an independent uniform probability p. We use 15 random graphs with low (p less than 30%), 15 with medium (p between 40 and 60%) and 15 with high (p greater than 70%) density.

In order to analize the performance of the heuristics and cutting planes, we consider four implementations.

The first implementation (BC-ECTLP) applies initial and primal heuristics and it also incorporates our cutting planes. Regarding CPLEX parameters, all of them are set to their default values except the dynamic search and the advanced presolve linear reduction. They must be disabled because user callback routines (primal heuristic and user cuts) are not compatible with them. In the second one (BC-ECTLPwP) we disable our primal heuristic, enabling CPLEX heuristics.

BC-CPLEX maintains the same values both for CPLEX and BC-ECTLP parameters, but does not incorporate our cutting planes or our primal heuristic. Finally, the last one (CPXDef) is the Branch-and-Cut CPLEX default setting configuration based on Model 1 with all parameters having their own default values.

In all implementations, $\left\lceil \frac{\Delta + 1}{2} \right\rceil$ is used as a lower bound of the value of the objective function.

Regarding cutting plane generation, after several computational tests, we chose a strategy that applies cutting plane generation at the root node. Lower and upper bound inequalities are explicitly included in the initial formulation. In each round, violated 3-dicycle, 3-cycle and DifLB are added as cutting planes. Regarding DifLB inequalities, we found that it is more efficient to restrict the search to $S = N(v) \backslash \{w_1, w_2\}$. Moreover, preliminary computational experience showed that the best option is to look for cycles of size 3 and not to consider fence inequalities. Note that, in all cases, there is a polynomial number of inequalities and the corresponding separation problems could be efficiently tackled through direct enumeration.

The initial heuristic is run once at the beginning and the primal heuristic is applied to each fractional solution of the search tree.

In Tables 4, 5, 6 and 7 we report average CPU time and number of nodes in the tree over solved instances and average gap over unsolved instances for $n = 15, 20, 30$ and 40,

Table 4 Computational results for $n = 15$

Density	Method	Time				Nodes	%Gap		
		Aveg	Best	Worst	#Win	Aveg	Aveg	Best	Worst
Low	BC-ECTLP	0.01	0.00	0.05	2	11	0.00	0.00	0.00
	BC-ECTLPwP	0.02	0.00	0.10	3	36	0.00	0.00	0.00
	BC-CPX	0.06	0.00	0.59	4	100	0.00	0.00	0.00
	CPXDef	0.03	0.00	0.26	2	89	0.00	0.00	0.00
Medium	BC-ECTLP	0.20	0.01	0.65	9	402	0.00	0.00	0.00
	BC-ECTLPwP	0.72 (14)	0.02	3.26	3	4751	16.67	16.67	16.67
	BC-CPX	34.84	0.03	515.80	0	269,294	0.00	0.00	0.00
	CPXDef	9.65	0.04	123.40	3	89,545	0.00	0.00	0.00
High	BC-ECTLP	72.79 (8)	0.44	571.14	3	146,372	11.90	11.11	12.50
	BC-ECTLPwP	50.03 (6)	2.28	184.93	1	262,890	12.72	11.11	20.00
	BC-CPX	147.26 (9)	0.36	1100.50	0	796,598	13.43	11.11	22.22
	CPXDef	343.14 (8)	0.28	1665.47	3	1,910,378	13.29	11.11	22.22

Table 5 Computational results for $n = 20$

Density	Method	Time				Nodes	%Gap		
		Aveg	Best	Worst	#Win	Aveg	Aveg	Best	Worst
Low	BC-ECTLP	0.10	0.00	0.70	4	118	0.00	0.00	0.00
	BC-ECTLPwP	0.39	0.00	3.61	4	3252	0.00	0.00	0.00
	BC-CPX	0.17	0.05	1.31	3	1069	0.00	0.00	0.00
	CPXDef	0.08	0.01	0.41	4	394	0.00	0.00	0.00
Medium	BC-ECTLP	3.17 (13)	0.29	10.12	10	9581	11.81	11.11	12.50
	BC-ECTLPwP	385.76 (11)	1.04	1675.91	0	1,129,283	12.04	11.11	12.50
	BC-CPX	102.01 (12)	0.25	541.34	2	371,855	12.04	11.11	12.50
	CPXDef	159.98 (11)	0.27	1036.10	1	975,478	11.18	10.00	12.50
High	BC-ECTLP	166.26 (1)	166.26	166.26	1	98,474	11.11	9.09	16.67
	BC-ECTLPwP	****	****	****	0	****	16.15	9.09	28.57
	BC-CPX	****	****	****	0	****	19.99	9.09	33.33
	CPXDef	****	****	****	0	****	19.99	9.09	33.33

Table 6 Computational results for $n = 30$

Density	Method	Time				Nodes	%Gap		
		Aveg	Best	Worst	#Win	Aveg	Aveg	Best	Worst
Low	BC-ECTLP	110.41	0.00	1316.32	9	192,054	0.00	0.00	0.00
	BC-ECTLPwP	4.66 (13)	0.00	22.36	0	12,813	12.50	12.50	12.50
	BC-CPX	7.61 (14)	0.00	65.72	1	21,376	12.50	12.50	12.50
	CPXDef	10.53 (14)	0.00	98.96	4	37,719	12.50	12.50	12.50
Medium	BC-ECTLP	146.38 (6)	10.24	495.89	5	57,586	9.73	7.69	14.28
	BC-ECTLPwP	62.77 (1)	62.77	62.77	0	84,569	16.15	9.09	28.57
	BC-CPX	35.32 (1)	35.32	35.32	0	59,100	19.31	10.00	28.57
	CPXDef	668.05 (3)	387.34	1194.34	2	967,371	25.47	10.00	47.83
High	BC-ECTLP	17.74 (1)	17.74	17.74	1	364	16.78	7.14	22.22
	BC-ECTLPwP	****	****	****	0	****	26.10	18.75	31.58
	BC-CPX	****	****	****	0	****	44.03	18.75	50.00
	CPXDef	****	****	****	0	****	48.23	45.83	50.00

Table 7 Computational results for $n = 40$

Density	Method	Time				Nodes	%Gap		
		Aveg	Best	Worst	#Win	Aveg	Aveg	Best	Worst
Low	BC-ECTLP	23.28 (13)	0.04	133.65	8	25,764	9.55	9.09	10.00
	BC-ECTLPwP	182.50 (10)	0.05	1579.83	1	276,778	11.19	9.09	16.67
	BC-CPX	74.71 (11)	0.06	274.48	3	88,159	11.72	9.09	16.67
	CPXDef	218.46 (13)	0.06	1437.85	3	204,313	10.80	9.09	12.50
Medium	BC-ECTLP	30.61 (3)	11.32	51.71	3	1044	14.45	8.33	21.43
	BC-ECTLPwP	****	****	****	0	****	26.95	13.33	35.29
	BC-CPX	****	****	****	0	****	42.45	13.33	50.00
	CPXDef	****	****	****	0	****	42.31	13.33	50.00
High	BC-ECTLP	****	****	****	****	****	18.96	4.76	26.92
	BC-ECTLPwP	****	****	****	****	****	28.33	23.08	34.62
	BC-CPX	****	****	****	****	****	49.29	48.57	50.00
	CPXDef	****	****	****	****	****	49.29	48.57	50.00

respectively. Values in parentheses show the number of instances solved to optimality within the CPU time limit. In order to provide a more detailed analysis, we also report the best and worst CPU time (for solved instances) and the best and worst final gap (for unsolved instances). Moreover, the number of instances where an algorithm outperforms the other is reported. Symbol **** is used to indicate that none of the corresponding instances could be solved within the CPU time limit imposed.

As expected, it can be observed that the higher the density, the more difficult the instance. Low density graphs are the easiest to solve. BC-ECTLP reaches optimality for all the instances up to 30 vertices, whereas the rest of the algorithms fail only for one or two instances. The difference in performance is more evident when the number of vertices increases. For $n = 40$, BC-ECTLP solves 13 instances, the same as CPXDef, but the average BC-ECTLP CPU time is 10% of CPXDef CPU time. BC-ECTLPwP and BC-CPX only solve 10 and 11 instances, respectively, and times are significantly higher than those in BC-ECTLP. In conclusion, CPU times and the number of solved instances show an evident outperformance of BC-ECTLP for low density instances.

In medium density, instances with $n = 20$ already seems to be hard. The last three implementations show more difficulties than BC-ECTLP to reach optimal solutions within the time limit. Moreover, the average time required by them on solved instances is significantly longer than the average BC-ECTLP time. BC-ECTLP success rate is 45% of all medium instances, CPXDef success rate is 10%, BC-ECTLPwP success rate is 5% and BC-CPX success rate is 3%. In instances where all algorithms fail, BC-ECTLP algorithm obtains significantly better gaps, achieving final gaps that are reduced by more than 65% in respect to CPXDef for $n = 40$.

High density instances are difficult, even for small sizes. In most of the cases, all algorithms fail to reach the optimal solution within the time limit. However, BC-ECTLP algorithm obtains significantly better average gaps. It is worth mentioning that the higher the size, the more difficult for CPXDef to improve the initial gap. For $n = 40$, the best CPXDef final gap is around 48% while BC-ECTLP obtains 4.76%. Furthermore, the worst BC-ECTLP final gap is about half of CPXDef final gap.

Usually, when a primal heuristic is applied, the required time to explore each node increases. Therefore, it makes it necessary to achieve a tradeoff between the computational time and the benefit that the heuristic brings. In the case of our heuristic, this time increased significantly. Nevertheless, a dramatic reduction in the number of tree nodes was observed in the Branch-and-Cut tree. The benefit of our primal heuristic is evident when comparing the results obtained through BC-ECTLP and BC-ECTLPwP; this proves to be essential to obtain good feasible solutions and to significantly prune the Branch-and-Cut tree. It should be taken into account that both implementations enable CPLEX heuristics, the difference only lies in that BC-ECTLP, as opposed to BC-ECTLPwP, uses also our primal heuristic.

To sum up, the main message of the table is that our algorithm dominates the Branch-and-Cut CPLEX default algorithm by the number of solved instances, the final gap and computational time.

6 Conclusion

We have proposed two integer programming models and a Branch-and-Cut algorithm for the edge coloring by total labeling problem. We have characterized several new valid inequalities for one of the models and developed initial and primal heuristics.

Although no polyhedral analysis was carried out, the efficiency of the inequalities has been demonstrated through computational experiments.

Our algorithm is competitive with state of the art generic Branch-and-Cut methods with respect to running time and quality of the solutions obtained. It is capable of solving instances that are out of the reach of CPLEX. We should notice that both, heuristic and cutting planes, have contributed to the success of BC-ECTLP. The results obtained by means of BC-ECTLPwP show that primal heuristic is crucial to finding good upper bounds and to pruning the tree in the Branch-and-Cut model. Consequently, running times and gaps obtained are substantially improved.

Regarding CPLEX parameters, no setting superiority is observed.

It is worth mentioning that the size of the instances considered is substantially smaller than regular instances for other coloring problems where graphs of hundred of vertices can be tackled in reasonable time. Based on the literature of the area, it is not surprising to observed that vertex and edge coloring variations present a wide range of computational difficulties. According to our experience, total labeling problems are one of the hardest types of labeling problems. We consider that our work is a promising practical approach to be pursued in practice for solving the edge coloring by total labeling problem. There is still further work to do to reach the state of the art of other coloring problems and to improve the algorithm. There is still place for new valid inequalities and other schemes to prune the search tree and speed up computational implementation.

Moreover, we would like to highlight that the integer programming formulations, as well as the valid inequalities, can be adapted to other edge coloring/labeling problems. Therefore, our work might prove to be an effective tool to deal with those problems for which there are very few or no proposals on exact algorithms in the literature. Bearing in mind that coloring/labeling problems do not only have theoretical interest but also many applications in practice, any advance in this direction is important to expand the state of the art in edge coloring/labeling algorithms.

References

Aardal, K., van Hoesel, S., Koster, A., Mannino, C., & Sassano, A. (2007). Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, *153*(1), 79–129.

Bača, M., Miller, M., & Ryan, J. (2007). On irregular total labellings. *Discrete Mathematics*, *307*(11), 1378–1388.

Brandt, S., Budajov, K., Rautenbach, D., & Stiebitz, M. (2010). Edge colouring by total labellings. *Discrete Mathematics*, *310*(2), 199–205.

Brelaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, *22*(4), 251–256.

Burke, E., Mareček, K., Parkes, A., & Rudová, H. (2012). A branch-and-cut procedure for the Udine Course Timetabling problem. *Annals of Operations Research*, *194*(1), 71–87.

Coll, P., Marenco, J., Méndez Díaz, I., & Zabala, P. (2002). Facets of the graph coloring polytope. *Annals of Operations Research*, *116*(1), 79–90.

Cornaz, D., Furini, F., & Malaguti, E. (2017). Solving vertex coloring problems as maximum weight stable set problems. *Discrete Applied Mathematics*, *217*, 151–162.

Grötschel, M., Jünger, M., & Reinelt, G. (1984). A cutting plane algorithm for the linear ordering problem. *Operations research*, *32*(6), 1195–1220.

Grötschel, M., Jünger, M., & Reinelt, G. (1985). Facets of the linear ordering polytope. *Mathematical Programming*, *33*(1), 43–60.

IBM. CPLEX Optimizer for z/OS. http://pic.dhe.ibm.com/infocenter/cplexzos/v12r6/index.jsp.

Jensen, T., & Toth, P. (1995). *Graph coloring problems*. Hoboken: Wiley.

Khennoufa, R., Seba, H., & Kheddouci, H. (2013). Edge coloring total k-labelling of generalized petersen graphs. *Information Processing Letters*, *113*(13), 489–494.

Lee, J., & Leung, J. (1993). A comparison of two edge-coloring formulations. *Operations Research Letters*, *13*(4), 215–223.

Malaguti, E., Méndez-Díaz, I., Miranda-Bront, J., & Zabala, P. (2015). A branch-and-price algorithm for the (k; c)-coloring problem. *Networks*, *65*(4), 353–366.

Misra, J., & Gries, D. (1992). A constructive proof of vizing's theorem. *Information Processing Letters*, *41*(3), 131–133.

Seba, H., & Khennoufa, R. (2013). Edge coloring by total labellings of 4-regular circulant graphs. *Electronic Notes in Discrete Mathematics*, *41*, 141–148.