A Branch-and-Cut Algorithm for the Equitable Coloring Problem Using a Formulation by Representatives

Laura Bahiense[*], Yuri Frota[†], Thiago F. Noronha[‡], Celso C. Ribeiro[§]

Abstract

An equitable k-coloring of a graph is defined by a partition of its vertices into k disjoint stable subsets, such that the difference between the cardinalities of any two subsets is at most one. The equitable coloring problem consists of finding the minimum value of k such that a given graph can be equitably k-colored. We present two new integer programming formulations based on representatives for the equitable coloring problem. We propose a primal constructive heuristic, branching strategies, and the first branch-and-cut algorithm in the literature of the equitable coloring problem. The computational experiments were carried out on randomly generated graphs, DIMACS graphs, and other graphs from the literature.

1 Introduction and motivation

Let $G = (V, E)$ be an undirected graph, where $V = \{1, \ldots, n\}$ is the set of vertices and E is the set of edges. An *equitable k-coloring of G* is a partition of V into k disjoint stable subsets such that the difference on the cardinalities of any two subsets is at most one. Each subset is associated with a color and called a *color set*. The Equitable Coloring Problem (ECP) consists of finding the minimum value of k such that there is an equitable k-coloring of G. This value is said to be the *equitable chromatic number* of G and is denoted by $\chi_=(G)$.

We notice that a graph may admit an equitable m-coloring, but not an equitable $(m+1)$-coloring. For example, the complete bipartite graph $K_{3,3}$ has an equitable 2-coloring, but it does not admit an equitable 3-coloring. This non-existence result can be extended to all complete bipartite graphs of the form $K_{2h+1,2h+1}$, for $h \geq 1$ [17].

The equitable coloring problem was first introduced by Meyer [20], motivated by a practical application to municipal garbage collection [24]. In this context, the vertices of the graph

[*]Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Produção, Rio de Janeiro, RJ 21945-970, Brazil, laura@pep.ufrj.br

[†]Universidade Federal Fluminense, Departamento de Ciência da Computação, Niterói, RJ 22410-240, Brazil, yuri@ic.uff.br

[‡]Universidade Federal de Minas Gerais, Departamento de Ciência da Computação, Belo Horizonte, MG 31270-010, Brazil, tfn@dcc.ufmg.br

[§]Universidade Federal Fluminense, Departamento de Ciência da Computação, Niterói, RJ 22410-240, Brazil, celso@ic.uff.br

represent garbage collection routes. A pair of vertices share an edge if the corresponding routes should not be run on the same day. It is desirable that the number of routes ran on each day be approximately the same. Therefore, the problem of assigning one of the six weekly working days to each route reduces to finding an equitable 6-coloring. Other applications arise from load balance in parallel memory systems [7], scheduling in communication systems [14], and partitioning and load balancing [2].

ECP was proved to be NP-hard in [11, 18]. Polynomial-time algorithms are known for split graphs [4] and trees [5]. Hajnal and Szemerédi [12] proved that every graph $G = (V, E)$ has an equitable $(\Delta + 1)$-coloring, where $\Delta = \max_{v \in V}\{d(v)\}$ and $d(v)$ denotes the degree of vertex $v \in V$. Kierstead and Kostochka [16] presented a shorter proof for this result and an approximation heuristic that attains this bound. In addition, they showed that every graph satisfying $d(u) + d(v) \leq 2k + 1$, for every edge $(u, v) \in E$, has an equitable $(k + 1)$-coloring.

A cut-and-branch algorithm was proposed in [1] for arbitrary graphs. It is based on the results of Hajnal and Szemerédi [12] and makes use of XPRESS Gomory and lifted cover cuts. The formulation used in this algorithm is the following:

$$\min \sum_{c=1}^{\Delta+1} w'_c \tag{1}$$

$$\sum_{c=1}^{\Delta+1} x'_{vc} = 1, \qquad \forall v \in V \tag{2}$$

$$x'_{vc} + x'_{uc} \leq w'_c, \qquad \forall (v, u) \in E, \quad \forall c \in \{1, \ldots, \Delta + 1\} \tag{3}$$

$$y'_c = \sum_{v=1}^{n} x'_{vc}, \qquad \forall c \in \{1, \ldots, \Delta + 1\} \tag{4}$$

$$y'_c - y'_\ell \leq 1 + M \cdot (2 - w'_c - w'_\ell), \qquad \forall c, \ell \in \{1, \ldots, \Delta + 1\} \tag{5}$$

$$y'_c - y'_\ell \geq -1 - M \cdot (2 - w'_c - w'_\ell), \qquad \forall c, \ell \in \{1, \ldots, \Delta + 1\} \tag{6}$$

$$x'_{vc} \in \{0, 1\}, \qquad \forall v \in V, \quad \forall c \in \{1, \ldots, \Delta + 1\} \tag{7}$$

$$w'_c \in \{0, 1\}, \qquad \forall c \in \{1, \ldots, \Delta + 1\} \tag{8}$$

$$y'_c \text{ integer}, \qquad \forall c \in \{1, \ldots, \Delta + 1\}, \tag{9}$$

where $x'_{vc} = 1$ if and only if color c is assigned to vertex v, $x'_{vc} = 0$ otherwise; $w'_c = 1$ if and only if $x'_{vc} = 1$ for some vertex v, $w'_c = 0$ otherwise, and y'_c are integer auxiliary variables. The objective function (1) counts the number of colors (or the number of color sets). Constraints (2) assert that each vertex must be assigned to exactly one color. Inequalities (3) enforce that adjacent vertices cannot share the same color. Equality (4) enforces that variable y'_c is equal to the number of vertices colored with color c. Inequalities (5) and (6) guarantee that the difference between the cardinalities of any two color sets is at most one, where M is a constant big enough to enforce $|y'_c - y'_\ell| \leq 1$ whenever both w'_c and w'_ℓ are positive. Constraints (7), (8) and (9) define integrality requirements on the variables. This model is weak, since the fractional solution where all x' variables are set to $1/(\Delta + 1)$, all w' variables are set to $2/(\Delta + 1)$ and all y' variables are set to $n/(\Delta + 1)$ is optimal for the initial relaxation of the model, which leads to a dual bound of 2.

Two polynomial-time heuristics for ECP are presented in [11]. A polyhedral approach for the equitable coloring problem was proposed in [19], but the largest instances exactly solved by that approach had as few as 35 nodes.

Campêlo et al. [6] proposed a 0-1 integer formulation for the graph coloring problem based on the idea of representative vertices. An asymmetric formulation and valid inequalities for the same problem were proposed in [3]. The formulations in [3, 6] have been extended by Frota et al. [10] to handle the partition coloring problem. In this paper, we explore the idea of a formulation by representatives to derive a branch-and-cut algorithm for equitable coloring. In the next section, we present combinatorial bounds to the cardinality of the color sets in any equitable coloring. These results are used by the integer programming formulations proposed in Section 3. Branch-and-cut algorithms for ECP are presented in Section 4. Computational results are reported in Section 5. Concluding remarks are drawn in the last section.

2 Bounds

We consider a graph $G = (V, E)$, with $|V| = n$, $|E| = m$, and $\Delta = \max_{v \in V}\{d(v)\}$, where $d(v)$ denotes the degree of node $v \in V$. The following result holds:

Theorem 1 (Hajnal and Szemerédi [12])**.** *Every graph G has an equitable $(\Delta+1)$-coloring.*

This theorem, together with the fact that $\chi_{=}(G) \geq 2$ for $E \neq \emptyset$, yields the following result:

Corollary 1. *If $E \neq \emptyset$, then $2 \leq \chi_{=}(G) \leq \Delta + 1$.*

Let $\underline{s}_k$ and $\overline{s}_k$ be, respectively, the minimum and maximum color set cardinalities in an equitable k-coloring of G. Furthermore, let w_k be the number of color sets with cardinality $\overline{s}_k$ in this equitable k-coloring. The following theorem holds:

Theorem 2. *For any equitable k-coloring of G, $\underline{s}_k = \lfloor \frac{n}{k} \rfloor$ and $\overline{s}_k = \lceil \frac{n}{k} \rceil$.*

Proof. The cardinalities of any two color sets in an equitable k-coloring differ by at most one. In consequence, $n = w_k \cdot \overline{s}_k + (k - w_k) \cdot \underline{s}_k$. If k divides n, then all color sets in an equitable k-coloring have the same cardinality and $\overline{s}_k = \underline{s}_k$. Then, $n = w_k \cdot \overline{s}_k + (k - w_k) \cdot \overline{s}_k = k \cdot \overline{s}_k$ holds only if $\overline{s}_k = \lceil \frac{n}{k} \rceil = \lfloor \frac{n}{k} \rfloor = \underline{s}_k$. Otherwise, if k does not divide n, we first prove that $\overline{s}_k = \lceil \frac{n}{k} \rceil$, and next that $\underline{s}_k = \lfloor \frac{n}{k} \rfloor$. Since there is an equitable k-coloring and k does not divide n, then $\overline{s}_k - \underline{s}_k = 1$ and $\lceil \frac{n}{k} \rceil - \lfloor \frac{n}{k} \rfloor = 1$. Assuming that $\overline{s}_k \neq \lceil \frac{n}{k} \rceil$, two cases have to be considered: $\overline{s}_k > \lceil \frac{n}{k} \rceil$ and $\overline{s}_k < \lceil \frac{n}{k} \rceil$. In the first case, there are w_k color sets of cardinality $\overline{s}_k > \lceil \frac{n}{k} \rceil$ and $k - w_k$ color sets of cardinality $\underline{s}_k = \overline{s}_k - 1 > \lceil \frac{n}{k} \rceil - 1 \geq \lceil \frac{n}{k} \rceil$. This leads to a contradiction, since in that case $n = w_k \cdot \overline{s}_k + (k - w_k) \cdot \underline{s}_k > w_k \cdot \lceil \frac{n}{k} \rceil + (k - w_k) \cdot \lceil \frac{n}{k} \rceil = k \cdot \lceil \frac{n}{k} \rceil > n$. Hence, $\overline{s}_k$ cannot be greater than $\lceil \frac{n}{k} \rceil$. In the second case, we have w_k color sets with cardinality $\overline{s}_k < \lceil \frac{n}{k} \rceil$. Therefore, $\overline{s}_k \leq \lceil \frac{n}{k} \rceil - 1 = \lfloor \frac{n}{k} \rfloor$ and $k - w_k$ color sets with cardinality $\underline{s}_k = \overline{s}_k - 1 < \lceil \frac{n}{k} \rceil - 1 = \lfloor \frac{n}{k} \rfloor$ exist. Once again, a contradiction would follow: $n = w_k \cdot \overline{s}_k + (k - w_k) \cdot \underline{s}_k < w_k \cdot \lfloor \frac{n}{k} \rfloor + (k - w_k) \cdot \lfloor \frac{n}{k} \rfloor = k \cdot \lfloor \frac{n}{k} \rfloor < n$. In consequence, $\overline{s}_k$ cannot be smaller than $\lceil \frac{n}{k} \rceil$. Therefore, $\overline{s}_k = \lceil \frac{n}{k} \rceil$. Since $\overline{s}_k - \underline{s}_k = 1$, then $\underline{s}_k = \overline{s}_k - 1 = \lceil \frac{n}{k} \rceil - 1 = \lfloor \frac{n}{k} \rfloor$, which completes the proof. □

Theorem 2 leads to the following bounds for the cardinality of the color sets in any equitable coloring:

Lemma 1. *Given an upper (resp. lower) bound U (resp. L) to $\chi_=(G)$, then $\lceil \frac{n}{U} \rceil$ (resp. $\lceil \frac{n}{L} \rceil$) is a lower (resp. upper) bound for $\overline{s}_k$ in any equitable k-coloring of G.*

Proof. From Theorem 2, $\overline{s}_{\chi_=(G)} = \left\lceil \frac{n}{\chi_=(G)} \right\rceil$. Since $\lceil \frac{n}{U} \rceil \leq \left\lceil \frac{n}{\chi_=(G)} \right\rceil \leq \lceil \frac{n}{L} \rceil$, then $\lceil \frac{n}{U} \rceil \leq \overline{s}_{\chi_=(G)} \leq \lceil \frac{n}{L} \rceil$. □

Corollary 1 and Lemma 1 yield the following result:

Corollary 2. *For any equitable $\chi_=(G)$-coloring of G, $\left\lceil \frac{n}{\Delta+1} \right\rceil \leq \overline{s}_{\chi_=(G)} \leq \lceil \frac{n}{2} \rceil$.*

3 Integer programming formulations

The formulation of the equitable coloring problem proposed in this section is based on choosing one vertex to be the representative of all vertices with the same color, instead of directly coloring all vertices. Therefore, each vertex is in either one of the following two states: (i) colored and representing all vertices colored with its color, or (ii) colored and represented by another vertex colored with the same color. Formulations by representatives have been successfully applied in the solution of other graph coloring problems, see [3, 6, 10].

3.1 Formulations

Let $A(u) = \{v \in V : (u,v) \notin E, v \neq u\}$ be the *anti-neighborhood* of a vertex $u \in V$ (i.e., the subset of vertices that are not adjacent to u). We also define $A'(u) = A(u) \cup \{u\}$, for any $u \in V$. Given a subset of vertices $V' \subseteq V$, we denote by $E[V']$ the subset of edges induced in the graph $G = (V,E)$ by V'. A vertex $v \in A(u)$ is said to be *isolated* in $A(u)$ if $E[A(u)] = E[A(u) \setminus \{v\}]$ (i.e., vertex v has no adjacent vertex in $A(u)$).

We define the binary variables x_{uv} for all $u \in V$ and for all $v \in A'(u)$, such that $x_{uv} = 1$ if and only if vertex u represents the color of vertex v; otherwise $x_{uv} = 0$. The equilibrium variable $w \in \mathbb{R}, w \geq 0$, indicates the cardinality of the maximum stable set of the equitable coloring (i.e., the cardinality of each stable set is either w or $w-1$). We also define L_w and U_w as integral lower and upper bounds for the value of w, respectively. The equitable coloring problem can be formulated as the following integer programming problem, in which the non-linear constraints (14) and (15) will be linearized later:

$$\min \sum_{u \in V} x_{uu} \tag{10}$$

subject to:

$$\sum_{v \in A'(u)} x_{vu} = 1, \qquad \forall u \in V \tag{11}$$

$$x_{uv} + x_{ub} \leq x_{uu}, \qquad \forall u \in V, \forall (v,b) \in E : v, b \in A(u) \tag{12}$$

$$x_{uv} \leq x_{uu}, \qquad \forall u \in V, \forall v \in A(u) : v \text{ is isolated in } A(u) \tag{13}$$

$$x_{uu} + \sum_{v \in A(u)} x_{uv} \leq w \cdot x_{uu}, \qquad \forall u \in V \tag{14}$$

$$x_{uu} + \sum_{v \in A(u)} x_{uv} \geq (w-1) \cdot x_{uu}, \qquad \forall u \in V \tag{15}$$

$$x_{uv} \in \{0,1\}, \qquad \forall u \in V, \forall v \in A'(u) \tag{16}$$

$$w \in \mathbb{R}, w \geq 0. \tag{17}$$

The above model is said to be a *formulation by representatives.* The objective function (10) counts the number of representative vertices, i.e., the number of colors (or the number of color sets). Constraints (11) enforce that each vertex $u \in V$ must be represented either by itself or by another vertex v in its anti-neighborhood. Inequalities (12) enforce that adjacent vertices have distinct representatives. Inequalities (12) together with constraints (13) ensure that a vertex can only be represented by a representative vertex. Inequalities (14) and (15) guarantee that the difference on the cardinalities of any two color sets is at most one. Constraints (16) and (17) define integrality and non-negativity requirements on the variables.

To break symmetries in the above formulation, we generalized the asymmetric formulation by representatives in [3]. We establish that a vertex $u \in V$ can only represent another vertex $v \in V$ if $u < v$. Therefore, the representative of a color set is the vertex with the smallest index among all those colored with this color. We define $A_>(u) = \{v \in A(u) : u < v\}$ as the *out-anti-neighborhood* of a vertex $u \in V$ (i.e., the vertices that cannot represent vertex u) and $A_<(u) = \{v \in A(u) : v < u\}$ as the *in-anti-neighborhood* of vertex $u \in V$ (i.e., the vertices that can represent vertex u, except himself). We also define $A'_>(u) = A_>(u) \cup \{u\}$ and $A'_<(u) = A_<(u) \cup \{u\}$, for any $u \in V$.

We also define $V^s = \{u \in V : A_<(u) = \emptyset\}$ as the set of vertices whose in-anti-neighborhoods in G are empty (i.e., the set of vertices that are always representatives). Since vertices in V^s are always representatives, then $x_{vv} = 1$ in any feasible solution, for any $v \in V^s$. Therefore, these variables may be removed from the symmetric (SF) formulation (10) to (17), which can then be rewritten as the asymmetric (AF) formulation (18) to (25) below:

$$\min \sum_{v \in V \setminus V^s} x_{vv} + \mid V^s \mid \tag{18}$$

subject to:

$$\sum_{v \in A'_<(u)} x_{vu} = 1, \qquad \forall u \in V \setminus V^s \tag{19}$$

$$x_{uv} + x_{ub} \leq \beta_u, \qquad \forall u \in V, \forall (v,b) \in E : v, b \in A_>(u) \tag{20}$$

$$x_{uv} \leq x_{uu}, \qquad \forall u \in V \setminus V^s, \forall v \in A_>(u) : v \text{ is isolated in } A_>(u) \tag{21}$$

$$\beta_u + \sum_{v \in A_>(u)} x_{uv} \leq w \cdot \beta_u, \qquad \forall u \in V \tag{22}$$

$$\beta_u + \sum_{v \in A_>(u)} x_{uv} \geq (w-1) \cdot \beta_u, \qquad \forall u \in V \tag{23}$$

$$x_{uv} \in \{0,1\} \qquad \forall u \in V, \forall v \in A'_>(u) \tag{24}$$

$$w \in \mathbb{R}, \tag{25}$$

where $\beta_u = 1$ if $u \in V^s$; otherwise $\beta_u = x_{uu}$.

Constraints (22) and (23) in the above formulation can be linearized and replaced by constraints (26) and (27) below:

$$\beta_u + \sum_{v \in A_>(u)} x_{uv} \leq w - L_w \cdot (1 - \beta_u), \qquad \forall u \in V \tag{26}$$

$$\beta_u + \sum_{v \in A_>(u)} x_{uv} \geq (w-1) - (U_w - 1) \cdot (1 - \beta_u), \qquad \forall u \in V \tag{27}$$

resulting in formulation LF$_1$ defined by the objective function (18) and constraints (19) to (21) and (24) to (27). The number of variables in this formulation is $\mid V \backslash V^s \mid + \overline{m} + 1$, where $\overline{m} = n \cdot (n-1)/2 - m$ is the number of edges in the complementary graph $\overline{G}$ of G.

An alternative approach to linearize constraints (22) and (23) consists of introducing new variables y_i, for every integer i in the interval $[L_w, U_w]$: $y_i = 1$ if the cardinality of the maximum stable set in the partition defining an equitable coloring of G is i; $y_i = 0$ otherwise. Therefore, w can be replaced by $\sum_{i=L_w}^{i=U_w} i \cdot y_i$ by adding constraint (28) to formulation AF:

$$\sum_{i=L_w}^{U_w} y_i = 1. \tag{28}$$

For every $u \in V$ and every integer $i \in [L_w, U_w]$, we also introduce new variables z_{ui} such that $z_{ui} = x_{uu} \cdot y_i$ for any integer solution of ECP, together with the following linear inequalities:

$$z_{ui} \leq y_i, \quad z_{ui} \leq x_{uu}, \quad z_{ui} \geq y_i + x_{uu} - 1, \quad \forall u \in V, \forall i \in [L_w, U_w]. \tag{29}$$

By substitution, constraints (22) can be rewritten as

$$\beta_u + \sum_{v \in A_>(u)} x_{uv} \leq \sum_{i=L_w}^{U_w} i \cdot z_{ui}, \qquad \forall u \in V. \tag{30}$$

Similarly, the same transformation can be applied to (23), resulting in

$$2 \cdot \beta_u + \sum_{v \in A_>(u)} x_{uv} \geq \sum_{i=L_w}^{U_w} i \cdot z_{ui}, \qquad \forall u \in V. \tag{31}$$

Therefore, the equitable coloring problem can be alternatively handled by the resulting formulation LF$_2$ defined by the objective function (18) and constraints (19) to (21), (24), and (28) to (31). The number of variables in this formulation is $|V \setminus V^s| + \overline{m} + \rho(1+|V|)$, where $\rho = U_w - L_w$. However, as $z_{ui} = y_i$ for all $u \in V^s$, z_{ui} can be replaced by y_i for all $u \in V^s$, which reduces the number of variables in formulation to LF$_2$ to $|V \setminus V^s| + \overline{m} + \rho(1+|V \setminus V^s|)$. Formulations LF$_1$ and LF$_2$ are evaluated and compared in Section 5.

3.2 Valid inequalities

To improve the linear relaxation bounds given by the previous formulations, we use the two families of valid inequalities originally proposed in [3, 6] for the graph coloring problem which remain valid for ECP. *Internal cuts* (32) give a bound to the minimum number of colors that are necessary to color any odd hole or anti-hole $H \subseteq V$:

$$\sum_{v \in H \setminus V^s} x_{vv} + \mid H \cap V^s \mid + \sum_{v \in H \setminus V^s, u \in A_<(v) \setminus H} x_{uv} \geq \chi(H), \tag{32}$$

where $\chi(H)$ is the chromatic number of the subgraph induced by H in G. Theorem 3 below was proved in [3, 6]:

Theorem 3. *If $H \subseteq V$ induces an odd hole or an odd anti-hole in G, then (32) is a valid inequality for formulations LF$_1$ and LF$_2$.*

External cuts (33) are used to bound the maximum number of vertices in a subset $K \subseteq A_>(u)$ with a particular structure (such as cliques, odd holes, and odd anti-holes) that can be represented by the same vertex $u \in V$. They are strengthened versions of inequalities (20). For any vertex $v \in K$, we define α_v as the maximum size of an independent set of the graph $G[K]$ induced in G by K that contains v, and $\alpha_K = \max_{v \in K}\{\alpha_v\}$ as the size of the maximum independent set of $G[K]$. The external cut induced by vertex u and the subset K of nodes is defined by the inequality

$$\sum_{v \in K} \frac{x_{uv}}{\alpha_v} \leq \beta_u. \tag{33}$$

Theorem 4 below was proved in [3, 6]:

Theorem 4. *If $K \subseteq A_>(u)$ is a non-empty set and $u \in V$, then (33) is a valid inequality for formulations LF$_1$ and LF$_2$.*

The next corollary follows from Theorem 4 and the fact that $\alpha_v = 1$ for any $v \in K$, where K is a clique of G:

Corollary 3. *If $K \subseteq A_>(u)$ is a clique and $u \in V$, then*

$$\sum_{v \in K} x_{uv} \leq \beta_u \tag{34}$$

is a valid inequality for formulations LF$_1$ and LF$_2$.

Theorem 4 is valid for any non empty set. Therefore, it is valid, in particular, for odd holes and odd anti-holes. In both cases, the value of α_v, for any $v \in H$, is equal to the size α_H of the maximum stable set of $G[H]$, where $\alpha_H = \lfloor |H|/2 \rfloor$ for odd holes and $\alpha_H = 2$ for odd anti-holes. Thus, the left-hand side of (33) can be rewritten as

$$\sum_{v \in H} \frac{x_{uv}}{\alpha_v} = \sum_{v \in H} \frac{x_{uv}}{\alpha_H} = \frac{\sum_{v \in H} x_{uv}}{\alpha_H},$$

for any $u \in V$ and for any $H \subseteq A_>(u)$. Consequently, the next corollary holds:

Corollary 4. *If $H \subseteq A_>(u)$ is an odd hole (or an odd anti-hole) and $u \in V$, then*

$$\sum_{v \in H} x_{uv} \leq \alpha_H \beta_u \qquad (35)$$

is a valid inequality for formulations LF_1 and LF_2.

4 Branch-and-cut

In this section, we describe two branch-and-cut algorithms, each of them based on one of the formulations LF$_1$ and LF$_2$ proposed in the previous section. Valid inequalities are progressively added to each subproblem of the search tree, which in most cases improves the linear relaxation bound. Tight lower and upper bounds to the number of color sets in the optimal solution of ECP, as well as efficient branching strategies, are instrumental to reduce the search effort. The main components of the two branch-and-cut algorithms are discussed in this section. They differ from each other by the integer programming formulation and the branching strategy.

4.1 Cut generation

The cutting plane procedure is based on the valid inequalities (32) to (35), whose separation consists basically of finding cliques, odd holes, and odd anti-holes in the graph. It follows the scheme suggested in the branch-and-cut algorithm proposed in [10] for solving the partition coloring problem. We use a GRASP heuristic for finding clique cuts and a modification of the Hoffman and Padberg heuristic [13] for finding odd holes and odd anti-holes cuts.

4.1.1 Separation of external clique cuts

Let $G^u = (V^u, E^u)$ be the subgraph induced in G by the out-anti-neighborhood $V^u = A_>(u)$ of a vertex $u \in V$. Furthermore, let $\bar{x}_{uv}$ be the optimal value of variable x_{uv} in the linear relaxation of formulation LF$_1$ or LF$_2$, for any $u \in V$ and any $v \in A'(u)$. For any $u \in V$ such that $\bar{x}_{uu} > 0$, the separation of an external clique cut consists of finding a subset of nodes $K \subseteq V^u$ such that $\sum_{v \in K} \bar{x}_{uv} > \beta_u$.

We developed a GRASP [8, 9, 21, 22] heuristic for finding cuts. The heuristic is an iterative procedure composed of two phases: a *construction phase* and a *local search phase*. The construction phase finds an initial solution that may be later improved by the local search phase.

This heuristic attempts to find a clique C with maximum weight $\sum_{v \in K} \bar{x}_{uv}$ for each vertex $u \in V$ such that $\bar{x}_{uu} > 0$. Its construction phase begins with an empty set and builds clique C, one vertex at a time. Let $C \subseteq V^u$ be a clique of G and $\delta(C) = \{r \in V^u \backslash C : C \cup \{r\} \text{ is a clique of } G\}$. At each iteration of the construction phase, one vertex $r \in \delta(C)$ is inserted into C with probability $\bar{x}_{ur} / (\sum_{r' \in \delta(C)} \bar{x}_{ur'})$ and the set $\delta(C)$ is updated. The procedure is repeated until $\delta(C) = \emptyset$.

There is no guarantee that the construction phase returns a locally optimal solution with respect to some neighborhood. Therefore, clique C may be improved by a local search procedure. The neighborhood $\gamma(C)$ is defined as the set of all cliques obtained by exchanging a vertex $v \in C$ with another vertex $u \in \delta(C \backslash \{v\})$. The method starts with the solution provided by the construction phase. It iteratively replaces the current solution by that with maximum weight within its neighborhood. The local search halts when no better solution is found in the neighborhood of the current solution.

The heuristic stops after $10 \cdot |V^u|$ iterations have been performed since the last time the best solution was updated. The $|V^u|$ heaviest cliques are selected and a cut is generated for each clique C such that $\sum_{v \in C} \bar{x}_{uv} > \beta_u$.

4.1.2 Separation of external odd hole cuts and external odd anti-hole cuts

For any $u \in V$ such that $\bar{x}_{uu} > 0$, the separation of external odd hole cuts consists of finding an odd hole or an odd anti-hole $H \in V^u$ such that $\sum_{v \in H} \bar{x}_{uv} > \alpha_H \beta_u$.

We developed a generalization of the Hoffman and Padberg [13] algorithm for finding violated odd hole inequalities in $G^u = (V^u, E^u)$. The same algorithm is applied to the complementary graph to find violated odd anti-hole inequalities.

First, the method performs a breadth-first search labeling of the vertices of graph $G^u = (V^u, E^u)$, starting from any root vertex r randomly chosen. A label h_v is assigned to each vertex $v \in V^u$. Figure 1 (a) illustrates a layered graph with its vertex labels. For any two adjacent vertices v_1 and v_2 with labels $h_{v_1} = h_{v_2} \geq 2$, if there exist two vertex-disjoint shortest paths p_{v_1} (from v_1 to r) and p_{v_2} (from v_2 to r), then there exists an odd cycle that contains vertices v_1, v_2, and r, as illustrated by Figure 1 (b).

Next, the algorithm assigns weights $t_{v_1 v_2} = 2 - \bar{x}_{uv_1} - \bar{x}_{uv_2}$ to every edge $(v_1, v_2) \in E^u$. Then, for every edge $(v_1, v_2) \in E^u$ with $h_{v_1} = h_{v_2} \geq 2$, the algorithm computes two vertex-disjoint shortest paths, one from v_1 to r and the other from v_2 to r. If both paths exist, an odd hole that can be used to generate a violated external cut is found. Otherwise, the algorithm continues from the next edge. This algorithm is applied $0.4 \cdot |V^u|$ times, starting from different root vertices.

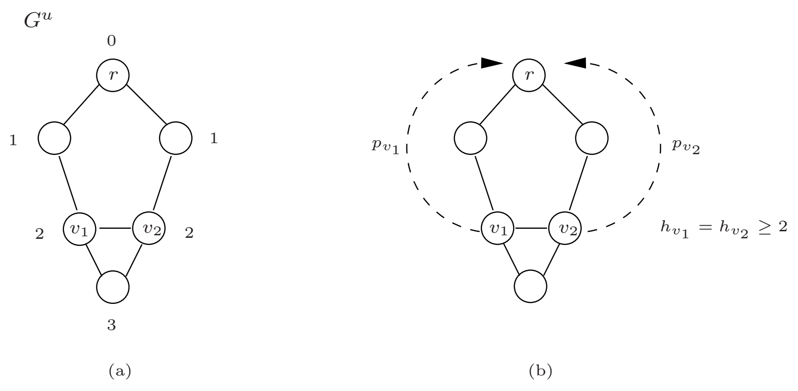

Figure 1: Layered graph with labels.

4.1.3 Separation of internal odd hole and internal odd anti-hole cuts

The separation of internal odd hole or odd anti-hole cuts consists of finding an odd hole or an odd anti-hole $H \in V$ such that

$$\sum_{v \in H \setminus V^s} \bar{x}_{vv} + |H \cap V^s| + \sum_{v \in H \setminus V^s, u \in A_<(v) \setminus H} \bar{x}_{uv} < \chi(G[H]).$$

The algorithm is similar to that developed for finding external odd hole cuts. There are two main differences with respect to the algorithm in Section 4.1.2. First, the weights applied to the edges of the layered graph are different. Second, the algorithm is applied to all vertices in G to find violated odd hole cuts. The same algorithm is applied to the complementary graph of G to find violated odd anti-hole inequalities. It starts by building a layered graph rooted at a randomly chosen vertex $r \in V$. Next, the algorithm assigns weights

$$t_{v_1 v_2} = \sum_{u \in A'_<(v_1) \setminus H} \bar{x}_{uv_1} + \sum_{u \in A'_<(v_2) \setminus H} \bar{x}_{uv_2}$$

to every edge $(v_1, v_2) \in E$. Then, for any edge $(v_1, v_2) \in E$ with $h_{v_1} = h_{v_2} \geq 2$, the algorithm calculates two vertex-disjoint shortest paths, one from v_1 to r and the other from v_2 to r. If both paths exist, an odd hole that can be used to generate a violated internal cut is found. Otherwise, the algorithm continues from the next edge. This algorithm is repeated $0.4 \cdot |V|$ times, starting from different root vertices.

4.2 Branching strategy

The branching strategy plays a major role in the success of a branch-and-cut algorithm. Branching on the x_{uv} variables, with $u \in V$ and $v \in A'(u)$, is not efficient because most of them are null in integral solutions. Therefore, our branching strategy is based on the cardinality variables w in formulation LF$_1$ and on the y_i variables in formulation LF$_2$. Branching on the x_{uv} variables starts only after all the w or y_i variables are integral.

Let $\bar{w}$ be the optimal value of variable w in the linear relaxation of LF$_1$. Two branches are generated if $\bar{w}$ is fractional: constraint $w \leq \lfloor \bar{w} \rfloor$ is added in the first branch, while constraint $w \geq \lceil \bar{w} \rceil$ is added in the second. If any of the y variables is fractional in the linear relaxation of LF$_2$, we branch on the variable y_i whose value in the linear relaxation is closest to 0.5 and two branches are generated: constraint $\sum_{j=L_w}^{j=i-1} y_j = 0$ is added in the first branch, while constraint $\sum_{j=i}^{j=U_w} y_j = 0$ is added in the second.

If none of the cardinality variables is fractional, then we branch on the variable x_{uu} whose value in the corresponding linear relaxation is closest to 0.5, with $u \in V$. If none of the latter is fractional, then we branch on the variable x_{uv} whose value in the corresponding linear relaxation is closest to 0.5, with $u \in V$ and $v \in A(u)$.

4.3 Upper bounds

We propose a tabu search heuristic to compute upper bounds for each subproblem of the branch-and-cut algorithms. The algorithm is based on the heuristic for the frequency assignment problem proposed in [23].

The initial solution sets the representative of each vertex $v \in V$ to be $argmax_{u \in A'_<(v)} x^*_{uv}$, where x^*_{uv} is the value of variable x_{uv} in the optimal solution of the linear relaxation of LF$_1$ or LF$_2$. This initial solution may violate constraints (20), as well as the equitable coloring constraints (22) and (23).

The objective function of the tabu search heuristic consists of minimizing the sum of four component costs c_1, c_2, c_3, and c_4. For every $u \in V$ and for every $v \in A'_>(u)$, let $\bar{x}_{uv} = 1$ if and only if vertex u represents the color of vertex v in the current solution of the tabu search; $\bar{x}_{uv} = 0$ otherwise. Furthermore, let $\bar{w} = \lceil \frac{|V|}{K} \rceil$ be the cardinality of a maximum stable set in a feasible solution with K colors (recall that the cardinality of each stable set is either w or $w-1$). The cost components c_1, c_2, c_3, and c_4 of the objective function can be calculated as:

$$c_1 = \sum_{v \in V \setminus V^s} \bar{x}_{vv} + \mid V^s \mid;$$

$$c_2 = \sum_{u \in V} \sum_{\substack{(v,b) \in E \\ v,b \in A_>(u)}} \max\{\bar{x}_{uv} + \bar{x}_{ub} - 1, 0\};$$

$$c_3 = \sum_{u \in V} \max\{\bar{w} - 1 - \beta_u - \sum_{v \in A_>(u)} \bar{x}_{uv}, 0\}; \text{ and}$$

$$c_4 = \sum_{u \in V} \max\{-\bar{w} + \beta_u + \sum_{v \in A_>(u)} \bar{x}_{uv}, 0\}.$$

The cost component c_1 counts the number of representative vertices (i.e., colors) in the solution defined by $\bar{x}$ and $\bar{w}$, while the other three components of the objective function measure the degree of infeasibility of this solution. Cost component c_2 measures the degree of infeasibility regarding constraint (20). It counts the number of adjacent vertices with the same representative, while components c_3 and c_4 measure the degree of infeasibility of the equitable coloring constraints (22) and (23), respectively. We notice that $c_2 = c_3 = c_4 = 0$ in any feasible solution.

The neighborhood N of any given solution consists of all solutions that can be obtained by replacing the representative $u \in A'_<(v)$ of a given vertex $v \in V$ by another vertex $b \in A'_<(v)$. For the sake of efficiency, we do not evaluate all $O(n^2)$ solutions in the neighborhood. Instead, we investigate a restricted neighborhood N' defined as follows. Given two parameters $p \in [0,1]$ and $q \in [0,1]$, we select at random a set $B \subseteq V$ with exactly $q \cdot |V|$ vertices. For each vertex $v \in B$, we evaluate the change in the cost function derived by replacing vertex v by each of those in a set $D \subseteq A'_<(v)$ with exactly $p \cdot |A'_<(v)|$ vertices, also selected at random. The values of p and q have been set, respectively, to 0.5 and 0.7 as in [23].

The tabu search procedure is implemented using a best-improvement local search strategy. At each iteration, the current solution is replaced by the best one in the restricted neighborhood N'. Therefore, the representative $u \in A'_<(v)$ of vertex $v \in V$ is replaced by another vertex $b \in A'_<(v)$. All solutions where u is the representative of vertex v are made tabu for $t \cdot \frac{\psi(u,v)}{\sigma(v)}$ iterations, where $\psi(u,v)$ is the total number of tabu search iterations where u was the representative of vertex v in the current solution, and $\sigma(v)$ is the number of iterations performed since the last time the representative of v has changed to u. The value of t was set to 100 as in [23].

A local search procedure based on the complete neighborhood N of the best known feasible solution is performed whenever the latter is updated. The tabu search resumes from the locally optimal solution obtained by this local search.

5 Computational experiments

Two branch-and-cut algorithms have been implemented and tested in the computational experiments. The first (B&C-LF_1) is based on formulation LF$_1$ (objective function (18) with constraints (19) to (21) and (24) to (27)), while the second (B&C-LF_2) is based on formulation LF$_2$ (objective function (18) with constraints (19) to (21), (24), and (28) to (31). The results obtained by these two new algorithms are compared directly with those provided by the branch-and-cut in [1], that was implemented using Gomory cuts and lifted cover cuts provided by the XPRESS solver.

We developed our own branch-and-cut framework and implemented it in C++. The search strategy implements a best first search criterion, based on the linear relaxation of formulations LF$_1$ and LF$_2$ with the external and internal cuts. The tailing off strategy adds the best cuts violated by at least 2%, with a maximum of n^2 external clique cuts, n external odd hole cuts and n internal odd hole cuts per iteration. The algorithm in [1] was also implemented in C++. Both algorithms have been compiled with version v3.41 of the Linux/GNU compiler. XPRESS version 2005-a was used exclusively as the linear programming solver. All experiments were performed on an 1.8 GHz AMD-Atlon machine with one Gbyte of RAM memory. The numerical experiments were carried out on 75 random instances generated in the same way as in [19], 20 graph coloring instances from the DIMACS challenge [15], and four Kneser graphs used in the computational experiments reported in [1].

We first investigate the effectiveness of the external and internal cuts. Table 1 displays the contribution of external and internal cuts to the linear relaxation of formulations LF$_1$ and LF$_2$ for the 20 graph coloring instances from the DIMACS challenge [15] and the four

Kneser graphs. The first column gives the name of each test instance, while the second gives its optimal value. The next five columns give the main statistics for the solution of the linear relaxation of formulation LF_1: the optimal value of the linear relaxation found by algorithm B&C-LF_1 without cuts, the computation time in seconds needed to compute the previous value, the optimal value of the linear relaxation found by algorithm B&C-LF_1^c with cuts, the total number of external and internal cuts added in the root node, and the computation time in seconds needed to compute this relaxation value. The last five columns give the same information for formulation LF_2 and algorithms B&C-LF_2 and B&C-LF_2^c.

We first observe that both formulations equally benefit from the external and internal cuts in terms of the optimal values of their linear relaxations: for all instances in this table, both formulations obtained the same lower bounds without the cuts and have been improved by the same amount with the cuts. The external and internal cuts improved the optimal values of the linear relaxations of LF_1 and LF_2 by approximately 12.45% on average, considering all 16 instances for which the lower bounds have been smaller than the optimal integral value. The cuts made it possible to find optimal integral lower bounds for three instances: queen.6_6, david, and $K_{9,4}$. Although both formulations lead to the same solution values, their running times and the number of cuts they add are different.

Instance	Optimal	B&C-LF_1	time	B&C-LF_1^c	cuts	time	B&C-LF_2	time	B&C-LF_2^c	cuts	time
	value		(s)			(s)		(s)			(s)
miles750	31	31.00	0	31.00	225	48	31.00	0	31.00	226	69
miles1000	42	42.00	0	42.00	223	52	42.00	0	42.00	292	42
miles1500	73	73.00	0	73.00	0	0	73.00	0	73.00	0	0
zeroin.i.1	49	49.00	1	49.00	0	15	49.00	1	49.00	0	13
zeroin.i.2	36	30.69	8	32.70	435	41	30.69	9	32.70	477	33
zeroin.i.3	36	31.43	7	33.32	327	35	31.43	5	33.32	345	39
queen.6_6	7	6.00	0	6.20	176	0	6.00	0	6.20	177	1
queen.7_7	7	7.00	0	7.00	232	1	7.00	0	7.00	148	1
queen.8_8	9	8.00	0	8.00	381	3	8.00	0	8.00	393	3
myciel3	4	2.67	0	3.00	4	0	2.67	0	3.00	5	0
myciel4	5	2.70	0	3.83	141	0	2.70	0	3.83	144	0
jean	10	10.00	0	10.00	0	0	10.00	0	10.00	0	0
anna	11	11.00	0	11.00	9	4	11.00	4	11.00	9	8
david	30	29.33	0	29.33	0	1	29.33	0	29.33	0	2
games120	9	9.00	1	9.00	115	29	9.00	0	9.00	132	23
$K_{5,2}$	3	2.50	0	2.50	2	0	2.50	0	2.50	2	0
$K_{7,2}$	6	3.50	0	3.50	18	0	3.50	0	3.50	19	0
$K_{7,3}$	3	2.24	0	2.92	249	1	2.24	1	2.92	276	1
$K_{9,4}$	3	2.00	1	3.00	5869	163	2.00	17	3.00	3572	120
1-FullIns-3	4	3.33	0	3.75	68	0	3.33	0	3.75	44	0
2-FullIns-3	5	4.25	0	4.73	269	1	4.25	0	4.73	311	2
3-FullIns-3	6	5.20	1	5.71	735	9	5.20	4	5.71	439	4
4-FullIns-3	7	6.17	0	6.71	1459	34	6.17	2	6.71	1855	31
5-FullIns-3	8	7.14	3	7.70	3921	121	7.14	1	7.70	3095	81

Table 1: Effectiveness of the internal and external cuts.

In the next experiment, we investigate the behavior of the tabu search heuristic. For each of the 20 graph coloring instances from the DIMACS challenge [15] and the four Kneser graphs, we provide information about the cost and computation time of the solutions obtained by tabu search starting from: (1) a solution with no vertices colored (i.e., from scratch); (2) a solution obtained by rounding the fractional variables in the optimal solution of the relaxation of formulation LF_1; and (3) a solution obtained by rounding the fractional variables in the optimal solution of the relaxation of formulation LF_2. In each case, we report on Table 2, the upper bound (UB) provided by the tabu search heuristic and the computation time (in seconds) needed to compute it. We observe that tabu search obtains better upper bounds and smaller computation times when starting from the linear relaxations of our formulations than when it starts from scratch.

Detailed comparative computational results are reported in Tables 3 to 6. The first three columns in each table display the name, the number of vertices, and the number of edges of each instance. The next four columns give lower (LB) and upper (UB) bounds for $\chi_=(G)$, the number of evaluated nodes in the branch-and-cut tree, and the CPU time (in seconds) taken by the algorithm in [1] to find the optimal. The two last groups of four columns give the same information for algorithms B&C-LF_1 and B&C-LF_2. A missing entry in any of the columns displaying CPU time indicates that the problem could not be solved within two hours of processing time by the corresponding algorithm. In this case, we give the lower and upper bounds, and the number of evaluated nodes at the time the algorithm was stopped.

Table 3 refers to benchmark DIMACS instances for graph coloring and four Kneser graph instances used in [1]. The lower bounds provided by the branch-and-cut algorithms based on formulations LF$_1$ and LF$_2$ are very close and always better than or equal to those provided by the branch-and-cut algorithm in [1]. Algorithm B&C-LF_1 solved 22 out of the 24 instances within two hours of processing time, while algorithm B&C-LF_2 solved all instances in Table 3. The branch-and-cut in [1] solved only 14 out of the 24 instances. The average relative gap $(UB - LB)/LB$ for the branch-and-cut algorithm in [1] is 1,662.1%, while the same gap for B&C-LF_1 and B&C-LF_2 is equal to 6.2% and 0.0%, respectively. The largest absolute gap $(UB - LB)$ for B&C-LF_1 and B&C-LF_2 over all 24 instances in Table 3 is equal to 26 and 0 colors, respectively, while the same value for the branch-and-cut algorithm in [1] corresponds to 204 colors (instance zeroin.i.3).

Table 4 shows the results obtained on 25 random instances with 50 nodes, generated in the same way as in [19]. An instance named rand_n_p_s has n vertices and a probability p for the existence of an edge between any two vertices. The last digit s is used to distinguish between instances with the same values of n and p. Once again, the lower bounds provided by the branch-and-cut algorithms based on formulations LF$_1$ and LF$_2$ are very close and always better than or equal to those provided by the branch-and-cut algorithm in [1]. Algorithms B&C-LF_1 and B&C-LF_2 solved, respectively, 21 and 24 out of the 25 instances within two hours of processing time, while the branch-and-cut in [1] managed to solve only seven instances. The average relative gap $(UB - LB)/LB$ for the branch-and-cut algorithm in [1] over the 25 instances is 17.4%, while the same value for B&C-LF_1 and B&C-LF_2 is much smaller and equal to 5.0% and 0.7%, respectively.

Tables 5 and 6 show the results for random instances with 60 and 70 nodes, respectively, once again generated in the same way as in [19]. As before, the lower bounds provided by the branch-and-cut algorithms based on formulations LF$_1$ and LF$_2$ are very close and always better than or equal to those provided by the branch-and-cut algorithm in [1]. The latter solved no instance with 60 and 70 nodes and edge density larger than 10% within two hours of processing time. Its average gap $(UB - LB)/LB$ over the instances with 60 and 70 nodes is 38.9% and 110.7%, respectively. The average gap of B&C-LF_1 is 6.1% and 7.7% over the instances with 60 and 70 nodes, respectively. Algorithm B&C-LF_2 performed even better: its average gap is equal to 2.6% and 6.7% over the instances with 60 and 70 nodes, respectively.

		Scratch		B&C-LF_1		B&C-LF_2	
Instance	Optimal value	UB	time (s)	UB	time (s)	UB	time (s)
miles750	31	35	13	35	3	35	3
miles1000	42	49	13	49	3	48	3
miles1500	73	77	13	73	4	73	3
zeroin.i.1	49	74	22	49	4	49	4
zeroin.i.2	36	95	22	81	5	84	5
zeroin.i.3	36	97	21	90	5	81	5
queen6_6	7	8	1	7	0	7	0
queen7_7	7	8	3	8	0	8	0
queen8_8	9	10	7	10	2	10	2
myciel3	4	4	0	4	0	4	0
myciel4	5	5	1	5	0	5	0
jean	10	10	3	10	0	10	0
anna	11	13	14	13	3	12	1
david	30	30	9	30	2	30	2
games120	9	11	6	11	1	9	0
$K_{5,2}$	3	4	0	4	0	3	0
$K_{7,2}$	6	6	0	6	0	6	0
$K_{7,3}$	3	5	0	5	0	5	1
$K_{9,4}$	3	6	2	3	0	3	0
1-FullIns-3	4	6	0	6	0	6	0
2-FullIns-3	5	8	1	8	0	8	0
3-FullIns-3	6	9	2	9	0	9	1
4-FullIns-3	7	11	5	11	1	11	0
5-FullIns-3	8	13	8	13	1	13	2

Table 2: Tabu search upper bounds for DIMACS and Kneser graphs.

			B&C [1]				B&C-LF_1				B&C-LF_2			
Graph	$\mid V \mid$	$\mid E \mid$	LB	UB	nodes	time	LB	UB	nodes	time	LB	UB	nodes	time
miles750	128	2113	3	58	1	-	31	31	5	124	31	31	6	171
miles1000	128	3216	3	67	130	-	42	42	10	214	42	42	13	267
miles1500	128	5198	2	86	1	-	73	73	1	13	73	73	1	13
zeroin.i.1	211	4100	2	211	1	-	49	49	3	79	49	49	1	50
zeroin.i.2	211	3541	2	211	1	-	35	61	97	-	36	36	23	510
zeroin.i.3	206	3540	2	206	2	-	35	61	101	-	36	36	28	491
queen.6_6	36	290	7	7	39447	941	7	7	1	2	7	7	1	1
queen.7_7	49	476	7	7	778	149	7	7	1	3	7	7	1	0
queen.8_8	64	728	8	10	4424	-	9	9	525	572	9	9	297	441
myciel3	11	20	4	4	19	0	4	4	5	0	4	4	7	0
myciel4	23	71	5	5	8867	12	5	5	255	4	5	5	237	5
jean	80	254	4	10	3453	-	10	10	1	3	10	10	1	4
anna	138	493	5	11	5913	-	11	11	1	18	11	11	2	26
david	87	406	7	30	8546	-	30	30	1	11	30	30	1	13
games120	120	638	9	9	11	39	9	9	1	35	9	9	1	30
$K_{5,2}$	10	15	3	3	1	0	3	3	1	0	3	3	1	0
$K_{7,2}$	21	105	6	6	40435	87	6	6	407	4	6	6	357	6
$K_{7,3}$	35	70	3	3	1	0	3	3	4	1	3	3	4	2
$K_{9,4}$	126	315	3	3	183	22	3	3	7	461	3	3	4	809
1-FullIns-3	30	100	4	4	73	3	4	4	10	2	4	4	34	2
2-FullIns-3	52	201	5	5	79	20	5	5	5	4	5	5	84	25
3-FullIns-3	80	346	6	6	21011	243	6	6	121	146	6	6	38	85
4-FullIns-3	114	541	7	7	5085	720	7	7	8	98	7	7	3	72
5-FullIns-3	154	792	8	8	24033	3088	8	8	77	1649	8	8	5	268

Table 3: Computational results for DIMACS and Kneser graphs.

The branch-and-cut algorithm in [1] performed better than the others only on the random instances with smaller edge densities (10%). This is due to the fact that the separation problem can be solved faster by the algorithm in [1], with the cuts generated being strong enough to prove optimality. Algorithms B&C-LF_1 and B&C-LF_2 performed best on instances with edge densities larger than 50%, because the larger is the graph density, the larger is the size of the cliques, odd holes, and odd anti-holes used to generate the internal and external cuts. The hardest instances for B&C-LF_1 and B&C-LF_2 were those with edge densities in the interval $[30\%, 50\%]$, because the cliques, odd holes, and odd anti-holes found for these instances were not large enough to generate effective internal and external cuts. Finally, it is important to emphasize that random instances with up to 70 nodes and edge densities ranging from 10% to 90% have been solved to optimality by the two new algorithms, while the largest solved to date by the algorithm in [19] had only 35 nodes.

The plot in Figure 2 displays in logarithmic scale the average relative gap $(UB - LB)/LB$ observed for B&C-LF_1, B&C-LF_2, and the branch-and-cut in [1] over all instances in each of the Tables 3 to 6. Algorithm B&C-LF_2 proposed in this work clearly outperformed the others.

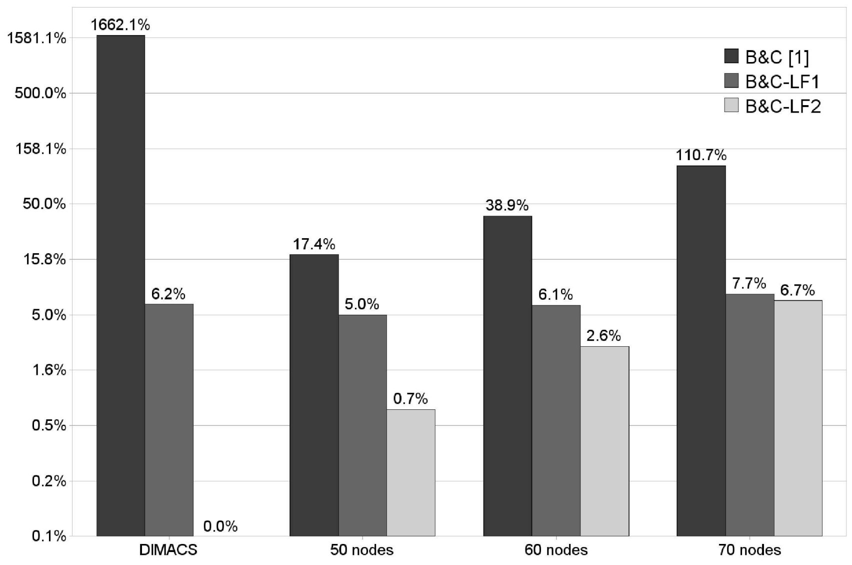

Figure 2: Average relative gaps observed for the branch-and-cut algorithm in [1], B&C-LF_1, and B&C-LF_2 over all instances in Tables 3 to 6.

The comparison with the cut-and-branch algorithm proposed in [19] is straightforward. The randomly generated instances solved by the latter had only 15 to 35 nodes, while algorithms B&C-LF_1 and B&C-LF_2 solved much larger instances with up to 70 nodes.

6 Concluding remarks

We proposed two branch-and-cut algorithms for the equitable graph coloring problem, based on two new integer programming formulations by representatives. Computational experiments have been carried out on 99 problem instances.

			B&C [1]				B&C-LF_1				B&C-LF_2			
Graph	$\mid V \mid$	$\mid E \mid$	LB	UB	nodes	time	LB	UB	nodes	time	LB	UB	nodes	time
rand_50_10_1	50	121	4	4	53	3	4	4	63	19	4	4	15	9
rand_50_10_2	50	103	3	3	17	3	3	3	12	4	3	3	8	8
rand_50_10_3	50	119	4	4	1	3	4	7	251945	-	4	4	2	1
rand_50_10_4	50	117	4	4	27	6	4	4	26	10	4	4	5	6
rand_50_10_5	50	100	3	3	16	1	3	3	5	3	3	3	5	4
rand_50_30_1	50	367	6	6	2831	235	6	6	200	61	6	6	1362	412
rand_50_30_2	50	359	6	6	2541	291	6	6	1451	441	6	6	112	53
rand_50_30_3	50	372	6	7	403054	-	6	7	84859	-	7	7	503	155
rand_50_30_4	50	357	6	6	6397	826	6	6	1248	339	6	6	744	231
rand_50_30_5	50	381	6	7	432545	-	6	7	62308	-	6	7	110739	-
rand_50_50_1	50	583	7	9	13720	-	9	9	25	15	9	9	25	17
rand_50_50_2	50	608	8	9	53726	-	9	9	81	44	9	9	1	9
rand_50_50_3	50	602	8	10	29278	-	9	10	36451	-	10	10	1035	273
rand_50_50_4	50	616	8	11	73951	-	10	10	151	45	10	10	187	56
rand_50_50_5	50	606	8	10	36049	-	9	9	1	8	9	9	1	7
rand_50_70_1	50	831	11	15	65046	-	13	13	161	42	13	13	161	42
rand_50_70_2	50	843	11	14	16119	-	14	14	29	12	14	14	29	12
rand_50_70_3	50	859	11	14	11825	-	14	14	31	14	14	14	1	7
rand_50_70_4	50	852	11	15	10142	-	14	14	11	11	14	14	11	11
rand_50_70_5	50	863	11	15	11307	-	14	14	1	8	14	14	1	8
rand_50_90_1	50	1105	21	26	7204	-	26	26	1	8	26	26	1	8
rand_50_90_2	50	1072	19	21	13177	-	21	21	1	8	21	21	1	8
rand_50_90_3	50	1075	20	26	8632	-	26	26	1	8	26	26	1	8
rand_50_90_4	50	1095	21	26	15903	-	26	26	1	8	26	26	1	8
rand_50_90_5	50	1101	20	21	16721	-	21	21	3	8	21	21	4	8

Table 4: Computational results for random graphs with 50 nodes.

			B&C [1]				B&C-LF_1				B&C-LF_2			
Graph	$\mid V \mid$	$\mid E \mid$	LB	UB	nodes	time	LB	UB	nodes	time	LB	UB	nodes	time
rand_60_10_1	60	166	4	4	35	6	4	4	16	22	4	4	20	38
rand_60_10_2	60	148	4	4	29	6	4	4	5	7	4	4	11	15
rand_60_10_3	60	150	4	4	13	5	4	4	7	14	4	4	8	16
rand_60_10_4	60	173	4	4	25	8	4	4	17	18	4	4	15	23
rand_60_10_5	60	152	4	4	57	9	4	4	6	8	4	4	9	14
rand_60_30_1	60	529	6	7	82426	-	7	8	25516	-	7	8	4368	-
rand_60_30_2	60	519	6	7	76495	-	6	8	23159	-	7	8	3945	-
rand_60_30_3	60	535	6	7	152382	-	7	8	25155	-	7	7	41	61
rand_60_30_4	60	509	6	7	206105	-	7	8	27251	-	7	7	137	111
rand_60_30_5	60	506	6	7	223443	-	6	8	26962	-	7	8	3873	-
rand_60_50_1	60	880	8	12	8112	-	11	11	337	187	11	11	351	214
rand_60_50_2	60	853	8	11	5178	-	10	10	1194	750	10	11	13501	-
rand_60_50_3	60	858	8	13	38413	-	10	10	21	40	10	10	5	15
rand_60_50_4	60	849	9	13	14105	-	10	11	19511	-	10	11	36708	-
rand_60_50_5	60	903	8	13	5941	-	10	12	18145	-	11	11	657	388
rand_60_70_1	60	1239	12	25	4826	-	16	16	25	15	16	16	33	19
rand_60_70_2	60	1240	12	18	3401	-	16	16	41	22	16	16	61	28
rand_60_70_3	60	1240	13	20	5966	-	16	17	18257	-	16	16	771	252
rand_60_70_4	60	1209	12	17	6153	-	16	16	75	27	16	16	63	28
rand_60_70_5	60	1261	12	22	2320	-	16	16	169	55	16	16	71	28
rand_60_90_1	60	1559	22	25	3823	-	24	24	26	13	24	24	1	9
rand_60_90_2	60	1561	18	31	456	-	31	31	1	9	31	31	1	9
rand_60_90_3	60	1571	19	26	858	-	25	25	1	9	25	25	1	9
rand_60_90_4	60	1583	22	37	4081	-	25	25	1	9	25	25	2	9
rand_60_90_5	60	1606	23	38	1699	-	26	26	1	9	26	26	1	9

Table 5: Computational results for random graphs with 60 nodes.

Random instances with up to 70 nodes have been solved to optimality by the new algorithms, while the largest instances solved to date by the cut-and-branch algorithm in [19] had only 35 nodes.

The results obtained by the new branch-and-cut algorithms were far superior to those given by the previously existing branch-and-cut approach. The average relative gap $(UB - LB)/LB$ observed for the branch-and-cut algorithm in [1] over all test instances was equal to 443.4%, while for B&C-LF_1 and B&C-LF_2 it was equal to only 7.4% and 2.4%, respectively. The largest absolute gap $(UB - LB)$ for algorithms B&C-LF_1 and B&C-LF_2 over the 75 random instances corresponded to five and two colors, respectively, while the same gap for the branch-and-cut algorithm in [1] was of 52 colors for instance rand_70_90_4. The numerical results have shown that algorithm B&C-LF_2 clearly outperformed B&C-LF_1 and previously existing approaches.

References

[1] L. Bahiense, S. Jurkiewicz, A. Lozano, M. Pimenta, C. Waga, and C. Valladares. An integer programming approach to equitable coloring problems. In *Proceedings of the XXXIX Brazilian Simposium on Operations Research*, volume 1, pages 1795–1801, Fortaleza, 2007.

[2] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling Computer and Manufacturing Processes.* Springer, Berlin, 2001.

[3] M. Campêlo, V. Campos, and R. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Electronic Notes in Discrete Mathematics*, 19:337–343, 2005.

[4] B.L. Chen, M.T. Ko, and K.W. Lih. Equitable and m-bounded coloring of split graphs. *Lecture Notes in Computer Science*, 1120:1–5, 1996.

[5] B.L. Chen and K.W. Lih. Equitable coloring of trees. *Journal of Combinatorial Theory – Series B*, 61:83–87, 1994.

[6] R.C. Corrêa, M. Campêlo, and Y.A.M. Frota. Cliques, holes and the vertex coloring polytope. *Information Processing Letters*, 89:159–164, 2004.

[7] S. K. Das, I. Finocchi, and R. Petreschi. Conflict-free star-access in parallel memory systems. *Journal of Parallel and Distributed Computing*, 66:1431–1441, 2006.

[8] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP - Part I: Algorithms. *International Transactions in Operational Research*, 16:1–14, 2009.

[9] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP - Part II: Applications. *International Transactions in Operational Research*, 16:131–172, 2009.

[10] Y. Frota, N. Maculan, T.F. Noronha, and C.C. Ribeiro. A branch-and-cut algorithm for partition coloring. *Networks*, 55:194–204, 2010.

[11] H. Furmanczyk and M. Kubale. The complexity of equitable vertex coloring of graphs. *Journal of Applied Computer Science*, 13:95–107, 2005.

[12] A. Hajnal and E. Szemerdi. Proof of a conjecture of P. Erdös. In P. Erdös, A. Rényi, and V.T. Sós, editors, *Combinatorial Theory and its Application*, pages 601–623, London, 1970. North-Holland.

[13] K.L. Hoffman and M. Padberg. Solving airline crew scheduling problems. *Management Science*, 39:657–682, 1993.

[14] S. Irani and V. Leung. Scheduling with conflicts and applications to traffic signal control. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 85–94, Atlanta, 1996.

[15] D.S. Johnson and M.A. Trick. *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, 1996.

[16] H. A. Kierstead and A. V. Kostochka. An ore-type theorem on equitable coloring. *Journal of Combinatorial Theory Series B*, 98:226–234, 2008.

[17] A.V. Kostochka, M.J. Pelsmajer, and D.B. West. A list analogue of equitable coloring. *Journal of Graph Theory*, 44:166–177, 2003.

[18] M. Kubale. *Graph Colorings*. American Mathematical Society, Providence, 2004.

[19] I. Méndez-Díaz, G. Nasini, and D. Severin. A polyhedral approach for the graph equitable coloring problem. In *Proceedings of the VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, Buenos Aires, 2008.

[20] W. Meyer. Equitable coloring. *American Mathematical Monthly*, 80:143–149, 1973.

[21] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer, 2003.

[22] M.G.C. Resende and C.C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as real problem solvers*, pages 29–63. Springer, 2005.

[23] S. Touhami. *Optimization Problems in Cellular Networks*. PhD thesis, John Molson School of Business, Concordia University, Montreal, 2004.

[24] A. Tucker. Perfect graphs and an application to optimizing municipal services. *SIAM Review*, 15:585–590, 1973.

			B&C [1]				B&C-LF_1				B&C-LF_2			
Graph	$\mid V \mid$	$\mid E \mid$	LB	UB	nodes	time	LB	UB	nodes	time	LB	UB	nodes	time
rand_70_10_1	70	209	4	4	37	11	4	9	35436	-	4	4	100	184
rand_70_10_2	70	205	4	4	42	11	4	4	24	23	4	4	22	31
rand_70_10_3	70	223	4	4	45	12	4	4	63	145	4	4	64	156
rand_70_10_4	70	224	4	4	22	18	4	4	18	21	4	4	10	26
rand_70_10_5	70	223	4	4	65	16	4	4	138	127	4	4	89	149
rand_70_30_1	70	688	6	8	19423	-	7	9	16648	-	7	9	4355	-
rand_70_30_2	70	717	6	8	13526	-	7	9	11435	-	7	8	4436	-
rand_70_30_3	70	686	6	8	39821	-	7	8	9983	-	7	8	4066	-
rand_70_30_4	70	695	6	8	26613	-	7	9	11563	-	7	9	3948	-
rand_70_30_5	70	719	6	8	62214	-	7	9	12261	-	7	9	600	-
rand_70_50_1	70	1189	8	15	2369	-	11	12	10923	-	11	12	8665	-
rand_70_50_2	70	1176	8	15	5549	-	10	11	3344	-	11	12	7563	-
rand_70_50_3	70	1210	8	14	1794	-	11	12	12197	-	11	12	6443	-
rand_70_50_4	70	1136	7	37	5263	-	10	12	6861	-	10	11	5371	-
rand_70_50_5	70	1202	9	14	3629	-	11	12	2800	-	11	12	1203	-
rand_70_70_1	70	1675	9	20	1	-	17	17	161	94	17	17	301	166
rand_70_70_2	70	1710	11	22	381	-	17	17	322	140	17	17	749	332
rand_70_70_3	70	1698	13	29	1169	-	18	18	869	458	18	18	52	38
rand_70_70_4	70	1677	14	26	1540	-	17	17	328	143	17	17	163	79
rand_70_70_5	70	1629	13	37	1889	-	17	17	2039	826	17	17	2127	749
rand_70_90_1	70	2154	11	32	1	-	29	29	101	30	29	29	1	10
rand_70_90_2	70	2153	21	38	30	-	28	28	1	10	28	28	1	8
rand_70_90_3	70	2168	16	33	1	-	29	29	14	12	29	29	28	14
rand_70_90_4	70	2160	12	64	1	-	29	29	26	14	29	29	16	12
rand_70_90_5	70	2159	7	30	1	-	27	27	1	10	27	27	1	8

Table 6: Computational results for random graphs with 70 nodes.