Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 156 (2008) 159–179
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

A cutting plane algorithm for graph coloring☆

Isabel Méndez-Díaz, Paula Zabala

Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina

Received 20 May 2004; received in revised form 6 December 2005; accepted 18 July 2006
Available online 18 April 2007

Abstract

We present an approach based on integer programming formulations of the graph coloring problem. Our goal is to develop models that remove some symmetrical solutions obtained by color permutations. We study the problem from a polyhedral point of view and determine some families of facets of the 0/1-polytope associated with one of these integer programming formulations. The theoretical results described here are used to design an efficient Cutting Plane algorithm.

Keywords: Graph coloring; Integer programming; Facets of polyhedra; Cutting plane algorithms

1. Introduction

The graph coloring problem (GCP) is perhaps one of the most well-known problems in graph theory. This problem arises in many applications such as scheduling, timetabling, electronic bandwidth allocation and sequencing. Despite this practical importance, there are relatively few methods available for solving the problem exactly [2,8,9,13,14].

Given a graph $G = (V, E)$ with n vertices and m edges, a *coloring* of G is an assignment of colors to each vertex such that the endpoints of any edge have different colors. A (k)-coloring of G is a coloring that uses k colors. The *chromatic number* of G is the smallest number of colors needed to color G and is denoted by $\chi(G)$. The coloring problem is to determine $\chi(G)$ and to find a coloring of G that uses $\chi(G)$ colors.

The coloring problem is known to be NP-hard for arbitrary graphs [7], while it is polynomially solvable for special classes of graphs, for instance perfect graphs. Even though there is little hope of finding a polynomial time algorithm for arbitrary graphs, this observation does not necessarily mean that it is impossible to devise algorithms that are reasonably fast and that can be used successfully in practice.

Like many optimization problems on graphs, the graph coloring problem can be formulated as a linear integer programming problem. In formulating discrete optimization problems, it is not only important to have a correct mathematical model, but also to have a well-structured model that can be solved effectively. Very often, there exists a natural symmetry inherent to the problem itself that, if propagated to the model, can hopelessly mire a cutting plane algorithm.

Since colors in GCP are indistinguishable, many symmetrical colorings typically exist for the same given number of colors. If feasible solutions of an integer programming GCP formulation also suffer from that symmetry drawback,

☆ This research was partially supported by UBACYT Grant X212, PICT Grant 11-09112 and CNPq PROSUL 4900333/04-4.
E-mail addresses: imendez@dc.uba.ar (I. Méndez-Díaz), pzabala@dc.uba.ar (P. Zabala).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.07.010

a cutting plane algorithm tends to behave poorly. The main reason for that is the fact that many iterations in the cutting procedure have the same optimal value. Bearing this in mind, we propose an approach based on integer programming formulations that reduces the number of symmetric feasible solutions in order to mitigate the effects of symmetry and to have a more tractable computational model.

Cutting plane algorithms are an important tool to deal with linear integer programming problems [15]. The main idea is to consider the linear relaxation and try to strengthen it by adding violated strong valid inequalities. The algorithm can use general cuts that do not take advantages of the structure, or specially developed ones that exploit the properties of the problem. In this paper we present a cutting plane algorithm for GCP. We strengthen the polyhedra associated with the proposed integer programming models with strong valid inequalities which we prove are facet-defining in many instances. The algorithm is tested on random graphs and on a set of test problems studied in the literature. Initial results were presented in [10].

The remainder of the paper is organized as follows. In Section 2, we present three models. Some families of facets for one of them are described in Section 3. Section 4 shows implementation details of our cutting plane algorithm. Experiments with different cuts combinations are reported in Section 5, as well as a comparison between the proposed models. In Section 6, we report our computational experience with the cutting plane algorithm on DIMACS benchmark (http://mat.gsia.cmu.edu/COLOR02) and finally the conclusions.

We expect the reader to be familiar with the polyhedral theory. See [11] for the background material needed.

We close this section by introducing all the notation and definitions used throughout the paper.

Let $G = (V, E)$ be a graph and $V' \subset V$. $G[V'] = (V', E')$ is the induced subgraph of G by V' with $E' = \{\{u, v\} : \{u, v\} \in E \text{ and } u, v \in V'\}$. $V' \subset V$ is a clique in G if $\forall u, v \in V', \{u, v\} \in E$. $V' \subset V$ is a stable set or independent set in G if $\forall u, v \in V', \{u, v\} \notin E$. A clique (stable set) K in G is maximal if there is no clique (stable set) $K' \neq K$ in G with $K \subset K'$. The stability number of G, $\alpha(G)$, is the maximum size of an independent set in G. A clique partition of the graph G is a partition $(K_1, \ldots, K_k)$ of V such that K_i is a clique in G for $i = 1, \ldots, k$. A sequence $v_1, \ldots, v_k$ of pairwise distinct vertices is a path in G if $\{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\} \in E$. A path is a cycle if in addition $\{v_1, v_k\} \in E$. A hole is a chordless cycle. The neighborhood of v is $N(v) = \{u : u \in V \text{ and } \{u, v\} \in E\}$. A graph G is bipartite if $\chi(G) \leqslant 2$.

2. Integer programming formulations

The problem of finding a minimum coloring in a graph can be formulated in many ways. In order to present integer programming formulations, we use x_{ij} to denote binary variables, with $i \in V$ and $1 \leqslant j \leqslant n$, where $x_{ij} = 1$ if color j is assigned to vertex i and $x_{ij} = 0$ otherwise. We also define n binary variables w_j for $j = 1, \ldots, n$, that indicate whether color j is used in some node, i.e., $w_j = 1$ if $x_{ij} = 1$ for some vertex i.

The following is a classical integer programming formulation:

$$\min \quad \sum_{j=1}^{n} w_j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in V, \tag{1}$$

$$x_{ij} + x_{kj} \leqslant w_j \quad \forall \{i, k\} \in E, \ 1 \leqslant j \leqslant n, \tag{2}$$

$$x_{ij} \in \{0, 1\} \ \forall i \in V, \ 1 \leqslant j \leqslant n \quad w_j \in \{0, 1\} \ 1 \leqslant j \leqslant n.$$

Constraints (1) assert that each vertex must receive exactly one color, and constraints (2) guarantee that every pair of adjacent vertices will not be assigned the same color and that $w_j = 1$ when some vertex has color j.

In [3], we present some facet-defining inequalities for the polytope associated with this formulation, $\mathscr{SCP}$. However, this is a symmetric formulation because all color permutations yield feasible solutions with the same objective function value, thus it is very difficult to use in practice and it is able to solve fairly small instances. The drawback arises from the indistinguishability of the variables. To avoid the symmetry of the classical model, next we propose three new models that eliminate equivalent solutions with different criteria.

2.1. Color order model

Given a (k)-coloring, any choice of k colors from $\{1, \ldots, n\}$ gives a feasible solution of $\mathscr{SCP}$ and they are all equivalent solutions. In order to eliminate some of these solutions we impose that color j can be assigned to a vertex provided color $j-1$ has already been assigned. In this way, all symmetrical (k)-coloring using colors with label greater than k are not considered. To characterize these feasible solutions, it is enough to add the following sets of constraints:

$$w_j \leqslant \sum_{i \in V} x_{ij} \quad \forall 1 \leqslant j \leqslant n, \tag{3}$$

$$w_j \geqslant w_{j+1} \quad \forall 1 \leqslant j \leqslant n-1. \tag{4}$$

Let $\mathscr{CP} = conv(\mathscr{SCP} \cap \{(x, w) : (x, w) \text{ satisfies (3) and (4)}\})$.

The reduction of feasible solutions from $\mathscr{SCP}$ is very significant.

2.2. Independent set order model 1

Given a (k)-coloring, in $\mathscr{CP}$ there are still equivalent solutions arising from permutations of the first k colors. To eliminate some of these solutions, the next model is more restricted than the first one and imposes that the number of vertices colored by j must be greater or equal than the number of vertices colored by $j+1$. In order to consider only these solutions, we add to $\mathscr{SCP}$ the following inequalities:

$$w_j \leqslant \sum_{i \in V} x_{ij} \quad \forall 1 \leqslant j \leqslant n, \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} \geqslant \sum_{i=1}^{n} x_{ij+1} \quad \forall 1 \leqslant j \leqslant n-1. \tag{6}$$

Let $\mathscr{CP}1 = conv(\mathscr{SCP} \cap \{(x, w) : (x, w) \text{ satisfies (5) and (6)}\})$.

The polytope $\mathscr{CP}1$ is included in $\mathscr{CP}$. Given a coloring, the larger the amount of independent sets with different cardinal, the greater the elimination of symmetrical solutions.

2.3. Independent set order model 2

Given a partitioning into independent sets, permutations of colors between independent sets with same cardinal give symmetrical solutions for the above model.

In order to eliminate these solutions the following model considers a unique assignment of colors for each partitioning into independent sets. We sort the associated independent sets by the minimum label of the vertices belonging to each set and we only consider the coloring that assigns color j to the jth independent set. All other permutations that define the same coloring do not lead to a feasible solution.

The constraints to eliminate equivalent colorings from $\mathscr{SCP}$ are

$$x_{ij} = 0 \quad \forall j \geqslant i+1, \tag{7}$$

$$x_{ij} \leqslant \sum_{k=j-1}^{i-1} x_{kj-1} \quad \forall i \in V \backslash \{1\} \ \forall 2 \leqslant j \leqslant i-1. \tag{8}$$

Let $\mathscr{CP}2 = conv(\mathscr{SCP} \cap \{(x, w) : (x, w) \text{ satisfies (7) and (8)}\})$.

This model eliminates completely the symmetry that arises from color indistinguishability.

The three models eliminate symmetrical solutions with different criteria. It should be natural to prefer the model with the smallest number of equivalent solutions. However, it is necessary to take into account other factors, such as number of variables, number of constraints, performance of LP solver on the associated LP-relaxation, etc.

To compare and evaluate the models, in Section 5 we experiment with a cutting plane algorithm and we analyze the lower bound quality and the CPU time to achieve it.

3. The coloring polytope

The aim of this section is to analyze polyhedral properties of the formulations introduced above.

From a polyhedral point of view, $\mathscr{CP}1$ presents difficulties to be characterized since it depends on some properties of the graph. For example, if any optimal coloring of G assigns the same color to two vertices at most, all feasible solutions of $\mathscr{CP}1$ satisfy $\sum_{i=1}^{n} x_{i2} = 2 - w_{n-1}$. Otherwise, if there is at least one $(\chi(G))$-coloring such that $\sum_{i=1}^{n} x_{i1} \geqslant 3$, then this equation is not satisfied by all feasible solutions.

$\mathscr{CP}2$ is even more difficult to be characterized. For example, let $G = (V, E)$ be a graph with $|V| = 7$. Consider $v, v', u, w \in V$ such that $K = V \backslash \{v, v'\}$ is a clique, $N(v) = \{u\}$ and $N(v') = \{w\}$. If the graph is labeled such that $v_1 = v$, $v_7 = v'$, $v_4 = u$ and $v_6 = w$, the dimension of $\mathscr{CP}2$ is 25 and it has 42 facet-defining inequalities. However, if $v_1 = v$, $v_2 = v'$, $v_3 = u$ and $v_7 = w$, the dimension is 22 and there are 243 facet-defining inequalities.

Since $\mathscr{CP}1$ and $\mathscr{CP}2$ are both included in $\mathscr{CP}$, a polyhedral study of $\mathscr{CP}$ also provides valid inequalities for them. The study of $\mathscr{CP}$ provides useful information since we observed that in many instances, some of the facet-defining inequalities of $\mathscr{CP}$ are also facet-defining of $\mathscr{CP}1$ and $\mathscr{CP}2$.

3.1. Basic properties

We assume w.l.o.g. that G has neither universal vertices nor isolated vertices. It implies that $2 \leqslant \chi(G) < |V|$.

First, we want to find a minimal equation system for $\mathscr{CP}$ and to determine its dimension.

Proposition 1. *The dimension of $\mathscr{CP}$ is $n^2 - \chi(G) - 1$ and a minimal equation system is defined by*

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in V,$$
$$w_j = 1 \quad 1 \leqslant j \leqslant \chi(G),$$
$$\sum_{i=1}^{n} x_{in} = w_n.$$

Proof. We must show that (a) every feasible solution satisfies these constraints, (b) they are mutually independent, and (c) there are $n^2 - \chi(G)$ affinely independent feasible solutions. Condition (a) arises from the definition of the model, and it is not difficult to verify (b), since they act over disjoint sets of variables. To see (c), consider the colorings given in the following table, where each row corresponds to a coloring and each column to colors. W.l.o.g we suppose that v_{n-1} and v_n are not adjacent.

- *Set* 0: Any (n)-coloring. We denote v_i the vertex colored by i, for $i = 1, \ldots, n$.
- For $i = 1, \ldots, n-2$. *Set* i: For $j = 1, \ldots, n-1$ and $i \neq j$, from the above coloring, we define (n)-colorings by switching colors j, n and i. We assign color j to v_n, color i to v_j and color n to v_i.
  In this way, we construct $(n-1)(n-2)$ colorings.
- *Set* $n-1$: For $j = 1, \ldots, n-1$, we consider the (n)-coloring that assigns color j to v_n, color n to v_j and color i to v_i for $i \neq j$, $i \neq n$. There are $n-1$ colorings.
- *Set* n: We consider the $(n-1)$-coloring that assigns color i to v_i for $i = 1, \ldots, n-2$, and color $n-1$ to v_{n-1} and v_n. From this $(n-1)$-coloring, define the $(n-1)$-coloring by switching color i and color $n-1$. If we apply this procedure for all $i = 1, \ldots, n-2$, we construct $n-2$ colorings.

	color 1	color 2	color 3	...	color i	...	color $n-1$	color n
Set 0	v_1	v_2	v_3	...	v_i	...	v_{n-1}	v_n
Set 1	v_n	v_1	v_3	...	v_i	...	v_{n-1}	v_2
	v_n	v_2	v_1	...	v_i	...	v_{n-1}	v_3
	$\vdots$							

	color 1	color 2	color 3	...	color i	...	color $n-1$	color n
	v_n	v_2	v_3	...	v_1	...	v_{n-1}	v_i
	$\vdots$							
	v_n	v_2	v_3	...	v_i	...	v_1	v_{n-1}
	v_n	v_2	v_3	...	v_i	...	v_{n-1}	v_1
Set 2	v_2	v_n	v_3	...	v_i	...	v_{n-1}	v_1
	v_1	v_n	v_2	...	v_i	...	v_{n-1}	v_3
	$\vdots$							
	v_1	v_n	v_3	...	v_2	...	v_{n-1}	v_i
	$\vdots$							
	v_1	v_n	v_3	...	v_i	...	v_2	v_{n-1}
	v_1	v_n	v_3	...	v_i	...	v_{n-1}	v_2
Set i	v_i	v_2	v_3	...	v_n	...	v_{n-1}	v_1
	v_1	v_i	v_3	...	v_n	...	v_{n-1}	v_2
	$\vdots$							
	v_1	v_2	v_3	...	v_n	...	v_i	v_{n-1}
	v_1	v_2	v_3	...	v_n	...	v_{n-1}	v_i
Set $n-1$	v_{n-1}	v_2	v_3	...	v_i	...	v_n	v_1
	v_1	v_{n-1}	v_3	...	v_i	...	v_n	v_2
	$\vdots$							
	v_1	v_2	v_3	...	v_{n-1}	...	v_n	v_i
	$\vdots$							
	v_1	v_2	v_3	...	v_i	...	v_n	v_{n-1}
Set n	v_{n-1}, v_n	v_2	v_3	...	v_i	...	v_1	–
	v_1	v_{n-1}, v_n	v_3	...	v_i	...	v_2	–
	$\vdots$							
	v_1	v_2	v_3	...	v_{n-1}, v_n	...	v_i	–
	v_1	v_2	v_3	...	v_i	...	v_{n-1}, v_n	–

Besides that, take one (j)-coloring for each $j = \chi(G), \ldots, n-2$.

Let (X^{C^i}, W^{C^i}) for $i = 1, \ldots, n^2 - \chi(G)$ be the associated feasible solution to each coloring. To see they are affinely independent, consider a linear combination

$$\sum_{i=1}^{n^2-\chi(G)} \alpha_i (X^{C^i}, W^{C^i}) = 0 \quad \text{such that} \quad \sum_{i=1}^{n^2-\chi(G)} \alpha_i = 0.$$

We have to see that $\alpha_i = 0$ for $i = 1, \ldots, n^2 - \chi(G)$.

The first (n)-coloring is the unique coloring that assigns color n to v_n, then $\alpha_1 = 0$. There is one (j)-coloring for each $j = \chi(G), \ldots, n-2$, then it is easy to get $\alpha_i = 0$ for these solutions.

The first $(n-3)$ colorings from each *Set j* for $j = 1, \ldots, n-2$, are unique in the color assigned to v_j. It implies that $\alpha_i = 0$ for them.

If we consider the set of colorings with a chosen variable equal to 1, we obtain the following implications:

(1) $w_{n-1} = 1$ and $w_n = 0 \Rightarrow \sum_{j=1}^{n-1} \alpha_{(n-2)n+j+2} = 0$.
(2) $x_{v_{n-1}n} = 1 \Rightarrow \sum_{j=1}^{n-2} \alpha_{jn-j} + \alpha_{(n-3)n+(n-1)+3} = 0$.
(3) $x_{v_{n-1}j} = 1 \Rightarrow \alpha_{(n-3)n+j+3} + \alpha_{(n-2)n+j+2} = 0 \ \forall j = 1, \ldots, n-2$.
(4) $x_{v_j n-1} = 1 \Rightarrow \alpha_{jn-j} + \alpha_{(n-2)n+j+2} = 0 \ \forall j = 1, \ldots, n-2$.
(5) $x_{v_{n-1}n-1} = 1 \Rightarrow \sum_{j=1}^{n-2} \alpha_{jn-j+1} + \alpha_{(n-2)n+(n-1)+2} = 0$.
(6) $x_{v_n j} = 1 \Rightarrow \alpha_{jn-j} + \alpha_{jn-j+1} + \alpha_{(n-2)n+j+2} = 0 \ \forall j = 1, \ldots, n-2$.
(7) $x_{v_i n} = 1 \Rightarrow \alpha_{jn-j+1} + \alpha_{(n-3)n+j+3} = 0 \ \forall j = 1, \ldots, n-2$.

From (4) and (6) it follows that $\alpha_{jn-j+1} = 0$. Then, from (7) we obtain $\alpha_{(n-3)n+j+3} = 0 \ \forall j = 1, \ldots, n-2$ and (5) implies $\alpha_{(n-2)n+(n-1)+2} = 0$. Replacing in (3), we get $\alpha_{(n-2)n+j+2} = 0 \ \forall j = 1, \ldots, n-2$ and from (4) we conclude $\alpha_{jn-j} = 0 \ \forall j = 1, \ldots, n-2$. Finally, (2) gives $\alpha_{(n-3)n+(n-1)+3} = 0$.

Therefore, we have $n^2 - \chi(G)$ affinely independent feasible solutions. Then, these constraints define a minimal equation system for $\mathscr{CP}$. □

Now we characterize inequalities from the original formulation that define facets.

Proposition 2. *Let $i_0 \in V$, the following holds*:

(a) *Every non-negativity constraint $x_{i_0 n} \geqslant 0$ defines a facet of $\mathscr{CP}$.*
(b) *For $j_0 = 1, \ldots n-1$, if $G - \{i_0\}$ is not a clique then $x_{i_0 j_0} \geqslant 0$ defines a facet of $\mathscr{CP}$.*

Proof.

(a) Consider one (n)-coloring that assigns color n to vertex i_0 and use the same procedure used in Proposition 1 to generate feasible solutions. Now, excluding the first one, we obtain $n^2 - \chi(G) - 1$ colorings lying on the face defined by the inequality.
(b) Since $G - \{i_0\}$ is not a clique, there are two non-adjacent vertices in $G - \{i_0\}$. For notational convenience, we denote them by v_n and v_{n-1}. Consider one (n)-coloring that does not assign color j_0 to vertex i_0. By following the above technique to generate colorings, and excluding the (n)-coloring that assigns j_0 to i_0, we obtain the necessary feasible solutions to proof that $x_{i_0 j_0} \geqslant 0$ is a facet. □

Proposition 3. *Let j_0 be any color such that $\chi(G) \leqslant j_0 \leqslant n-2$, then the constraint $w_{j_0} \geqslant w_{j_0+1}$ defines a facet of $\mathscr{CP}$.*

Proof. Consider the colorings from Proposition 1 except the (j_0)-coloring. They lie on the face $w_{j_0} = w_{j_0+1}$, then this is a facet of $\mathscr{CP}$. □

Completing the facets defined by the model inequalities, we state the following result.

Proposition 4. *The inequality $w_{j_0} \leqslant \sum_{i \in V} x_{i j_0}$ defines a facet of $\mathscr{CP}$ for all $j_0 = 1, \ldots, n-1$ iff there is some $(\chi(G))$-coloring lying on the face.*

Proof. Suppose that no $(\chi(G))$-coloring lies on the face. Then any feasible solution on the face satisfies $w_{\chi(G)+1} = 1$ and this is a contradiction.

Now, to see the inequality is a facet, it is enough to consider the set of colorings from Proposition 1, exclude the $(n-1)$-coloring that assigns color j_0 to v_{n-1} and v_n, and add for all $j = \chi(G), ..., n-2$ any (j)-coloring where color j_0 is assigned to a unique vertex. □

3.2. Independent set inequalities

Substructures of a graph give rise to valid inequalities for the coloring polytope of the whole graph.

We know that the set of vertices that are assigned the same color is an independent set, then its size is lower than or equal to the stability number of the graph. If we consider any subgraph $G' = (V', E')$, adapting that property to G' is quite straightforward and thus

$$\sum_{v \in V'} x_{vj_0} \leqslant \alpha(G') w_{j_0}$$

is valid for all $j_0 = 1, \ldots, n$ where $\alpha(G')$ is the stability number of G'.

Any $(n - \alpha(G') + 1)$-coloring that assigns color j_0 to a maximum independent set of G' satisfies it as an equality, then it must be a proper face of $\mathscr{CP}$. However, if $\alpha(G') \geqslant 2$, it is not a facet since no (n)-coloring satisfies it at equality. We can strengthen it by considering the color ordering to obtain

$$\sum_{v \in V'} x_{vj_0} + \sum_{j=n-\alpha(G')+1}^{n} \sum_{v \in V} x_{vj} \leqslant \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}.$$

Any $(n - \alpha(G') + r)$-coloring, where $r \geqslant 1$, has $\alpha(G') - r$ vertices sharing the color with other vertices. Therefore, there must be at most $\alpha(G') + 1$ vertices colored with $r + 1$ different colors, and this indicates that the inequality is valid.

We call it *Independent Set* inequality. The following proposition establishes when it is a facet-defining inequality.

Proposition 5. *Let* $V' \subset V$, $G' = G[V']$ *such that* $\alpha(G') < \alpha(G)$, $j_0 \leqslant n - \alpha(G')$ *and the valid* Independent Set *inequality*

$$\sum_{v \in V'} x_{vj_0} + \sum_{j=n-\alpha(G')+1}^{n} \sum_{v \in V} x_{vj} \leqslant \alpha(G') w_{j_0} + w_{n-\alpha(G')+1}.$$

If

- *There is an independent set of size* $\alpha(G') + 1$ *in* $G[V' \cup \{v\}]$ *for all* $v \in V \backslash V'$.
- *There exists* I, *a maximum independent set of* G', *such that* $V \backslash I$ *is not a clique*.
- *A* $(\chi(G))$*-coloring lies on the face*.

Then this is a facet-defining inequality.

Proof. Let F be the proper face of $\mathscr{CP}$ defined by the *Independent Set* inequality. Suppose that there is an inequality $\lambda^X X + \lambda^W W \leqslant \lambda_0$ valid with respect to $\mathscr{CP}$ such that $F \subseteq \mathscr{CP}\{(X, W) : \lambda^X X + \lambda^W W = \lambda_0\}$. To prove the proposition, we have to show that $(\lambda^X, \lambda^W, \lambda_0)$ can be written as a combination of the minimal equation system and the *Independent Set* inequality. We proceed to prove different cases that allow us to gather information about the coefficients of λ^X, λ^W and λ_0.

We have to prove:

(a) $\lambda^X_{vj} = \lambda^X_{vn} + \lambda^W_n \ \forall v \in V \ \forall n - \alpha(G') + 1 \leqslant j \leqslant n - 1$.
(b) $\lambda^X_{vj_0} = \lambda^X_{vn} + \lambda^W_n \ \forall v \in V'$.
(c) $\lambda^X_{vj_0} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1} \ \forall v \in V \backslash V'$.
(d) $\lambda^X_{vj} = \lambda^X_{vn-\alpha(G')+1} + \lambda^W_{n-\alpha(G')+1} \ \forall v \in V \ \forall j \neq j_0$ and $j \leqslant n - \alpha(G')$.
(e) If $j_0 \geqslant \chi(G) + 1$ then $\lambda^W_{j_0} = \alpha(G') \lambda^W_{n-\alpha(G')+1}$.
(f) $\lambda^W_j = 0$ for $j = \chi(G) + 1, \ldots, n - 1$ and $j \neq j_0, n - \alpha(G') + 1$.

To establish their validity, we construct pairs $C^1 = (X^1, W^1)$ and $C^2 = (X^2, W^2)$ of feasible solutions lying on F that differ in the components that enable us to derive each case. Each feasible solution has X's and W's components. For notational convenience, we note $C(v) = j$ to indicate that $x_{vj} = 1$. The W's components follow directly from X's.

For the sake of simplicity, the colorings are specified on some subset of vertices, and ensuring that there are enough colors to assign to the rest of the graph.

(a) Let $j, n-\alpha(G')+1 \leqslant j \leqslant n-1$ and $v_1 \in V\backslash V'$, $v_2 \in V$ non-adjacent vertices. Consider C^1, an $(n-1)$-coloring, such that $C^1(v_1)=C^1(v_2)=j$ and $C^1(v')=j_0$ for some $v' \in V'$. From this coloring, we build an (n)-coloring by assigning color n to v_1. Since both colorings are lying on F, we obtain $\lambda^X_{v_1 j}=\lambda^X_{v_1 n}+\lambda^W_n$. Now, consider $v \in V$, $v \neq v_1$ and an (n)-coloring such that $C(v_1)=j$, $C(v)=n$ and $C(v')=j_0$ for some $v' \in V'$. By interchanging colors j and n, we obtain $\lambda^X_{vj}+\lambda^X_{v_1 n}=\lambda^X_{vn}+\lambda^X_{v_1 j}$ and from it follows $\lambda^X_{vj}=\lambda^X_{vn}+\lambda^W_n$.

(b) Consider $v_1 \in V'$, $v_2 \in V\backslash V'$ non-adjacent vertices. Consider C^1, an $(n-1)$-coloring, such that $C^1(v_1)=C^1(v_2)=j_0$ and $C^1(v')=j_0$. From this coloring, we build an (n)-coloring by assigning color n to v_1. Since both colorings are lying on F, we obtain $\lambda^X_{v_1 j_0}=\lambda^X_{v_1 n}+\lambda^W_n$. Now, consider $v \in V'$, $v \neq v_1$ and an (n)-coloring such that $C(v_1)=j_0$, $C(v)=n$. By interchanging colors j_0 and n follows $\lambda^X_{vj_0}=\lambda^X_{vn}+\lambda^W_n$.

(c) Let $v \in V\backslash V'$. Under hypotheses, exists C^1, $(n-\alpha(G'))$-coloring, on F such that $C^1(v)=j_0$ and $\alpha(G')$vertices in V' are colored with j_0. If we change to $n-\alpha(G')+1$ the color of v, we have $\lambda^X_{vj_0}=\lambda^X_{vn-\alpha(G')+1}+\lambda^W_{n-\alpha(G')+1}$.

(d) Consider j, $j \leqslant n-\alpha(G')$, $j \neq j_0$. Under hypotheses, exists $I \subset V'$, maximal independent set such that $V\backslash I$ is not a clique, then there are $v_1, v_2 \in V\backslash I$ non-adjacent vertices. Let C^1, $(n-\alpha(G'))$-coloring, that assigns color j_0 to all vertices in I and $C^1(v_1)=C^1(v_2)=j$. Now, if we change to $n-\alpha(G')+1$ the color of v_1, we conclude that $\lambda^X_{v_1 j}=\lambda^X_{v_1 n-\alpha(G')+1}+\lambda^W_{n-\alpha(G')+1}$. Consider $v \in V$ and an (n)-coloring such that $C(v_1)=n-\alpha(G')+1$ and $C(v)=j$. By interchanging colors j and $n-\alpha(G')+1$ follows $\lambda^X_{vj}=\lambda^X_{vn-\alpha(G')+1}+\lambda^W_{n-\alpha(G')+1}$.

(e) Assume that $j_0 \geqslant \chi(G)+1$. Let $I=\{z_1,\ldots,z_{\alpha(G')}\}$, $I \subset V'$, a maximal independent set. Consider C^1, (j_0-1)-coloring, and C^2, (j_0)-coloring, such that $C^2(z_i)=j_0 \forall i=1,\ldots,\alpha(G')$. Let us denote by c^1_v and c^2_v the colors of v assigned by C^1 and C^2, respectively. Since both colorings lie on F, it follows that

$$\sum_{v\in V\backslash I}\lambda^X_{vc^1_v}+\sum_{i=1}^{\alpha(G')}\lambda^X_{z_i c^1_{z_i}}=\sum_{v\in V\backslash I}\lambda^X_{vc^2_v}+\sum_{i=1}^{\alpha(G')}\lambda^X_{z_i j_0}+\lambda^W_{j_0}.$$

But $\lambda^X_{vc^1_v}=\lambda^X_{vc^2_v}$ for all $v \in V\backslash I$ because $c^1_v, c^2_v \leqslant n-\alpha(G')-1$. Moreover, we prove that $\lambda^X_{z_i c^1_{z_i}}=\lambda^X_{z_i n-\alpha(G')+1}+\lambda^W_{n-\alpha(G')+1}=\lambda^X_{z_i j_0}+\lambda^W_{n-\alpha(G')+1}$. Combining these equalities, we conclude that $\lambda^W_{j_0}=\alpha(G')\lambda^W_{n-\alpha(G')+1}$.

(f) Let j, $j \neq j_0, n-\alpha(G')+1$ and $\chi(G)+1, \leqslant j \leqslant \ldots, n-1$. Consider the color r such that $r=j-1$ if $j-1 \neq j_0$, $r=j-2$ otherwise. We can assume that there is C^1, $(j-1)$-coloring, such that color r is assigned to more than one vertex. From this coloring we build C^2, (j)-coloring, by assigning color j to one of the vertices colored by r. Since $r \neq j_0, n-\alpha(G')+1$, we know that $\lambda^X_{vr}=\lambda^X_{rj}$. Then, we conclude that $\lambda^W_j=0$. □

This inequality becomes useful if we consider substructures with known stability number. That is the case for the following propositions.

Proposition 6. *Let K be a maximal clique. The* Clique *inequality*

$$\sum_{v\in K} x_{vj_0}-w_{j_0}\leqslant 0$$

is a facet-defining inequality of $\mathscr{CP}$.

Proof. It is enough to verify that $V'=K$ satisfies the hypotheses of the above proposition.

(1) Since K is a maximal clique, every $v \in V\backslash K$ has a non-adjacent vertex v' in K. Then $G[K\cup\{v\}]$ has an independent set of size 2.
(2) K is maximal clique, then there are $v_1, v_2 \in K$ such that v_1 has a non-adjacent vertex in $V\backslash K$. This implies that $V\backslash\{v_2\}$ is not a clique.
(3) Any coloring that assigns color j_0 to one vertex to K satisfies the inequality at equality. □

Proposition 7. *Let $C_k = \{v_1, \ldots, v_k\}$ be a hole of size k, $k > 3$. The* Hole *inequality*

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{j=n-\lfloor k/2 \rfloor+1}^{n} \sum_{v \in V} x_{vj} \leqslant \lfloor k/2 \rfloor w_{j_0} + w_{n-\lfloor k/2 \rfloor+1}$$

with $j_0 \leqslant n - \lfloor k/2 \rfloor$ is a facet-defining inequality of $\mathscr{CP}$ if:

- *For all $v \in V \backslash C_k$, there is an independent set of size $\lfloor k/2 \rfloor + 1$ in $G[C_k \cup \{v\}]$.*
- *A $(\chi(G))$-coloring lies on the face.*

Proof. We only have to check the first condition of Proposition 5. Since C_k is a hole of size greater than 3, we can assert that there is $I \subset C_k$, an independent set such that $V \backslash I$ is not a clique. □

The following results on an *antihole* and a *path* run along the same arguments as the previous result.

Proposition 8. *Let $\bar{C}_k = \{v_1, \ldots, v_k\}$ be an antihole of size k. The* Antihole *inequality*

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{v \in V} x_{vn-1} \leqslant 2w_{j_0} + w_{n-1} - w_n$$

with $j_0 \leqslant n - 2$ is a facet-defining inequality of $\mathscr{CP}$ if:

- *For all $v \in V \backslash \bar{C}_k$, there is an independent set of size 3 in $G[\bar{C}_k \cup \{v\}]$.*
- *A $(\chi(G))$-coloring lies on the face.*

Proposition 9. *Let $P_k = \{v_1, \ldots, v_k\}$ a* path *of size k. The* Path *inequality*

$$\sum_{i=1}^{k} x_{v_i j_0} + \sum_{j=n-\lceil k/2 \rceil+1}^{n} \sum_{v \in V} x_{vj} \leqslant \lceil k/2 \rceil w_{j_0} + w_{n-\lceil k/2 \rceil+1}$$

is a facet-defining inequality of $\mathscr{CP}$ if:

- *For all $v \in V \backslash P_k$, there is an independent set of size $\lceil k/2 \rceil + 1$ in $G[\bar{C}_k \cup \{v\}]$.*
- *A $(\chi(G))$-coloring lies on the face.*

3.3. Vertex inequalities

For any feasible solution of the polytope $\mathscr{CP}$, constraints (3) and (4) impose that if color j_0 is not assigned to some vertex, colors with label greater than j_0 are not assigned either. Moreover, no vertex uses more than one color. Both observations are put together in the following result.

Proposition 10. *Given $i_0 \in V$ and $1 \leqslant j_0 \leqslant n$, then the* Block Color *inequality $\sum_{j=j_0}^{n} x_{i_0 j} \leqslant w_{j_0}$ is a valid inequality for $\mathscr{CP}$. If $\chi(G) + 1 \leqslant j_0 \leqslant n - 2$, then this is facet-defining inequality of $\mathscr{CP}$.*

Proof. Consider the following feasible colorings:

(1) Any (n)-coloring that assigns color j_0 to i_0. We call v_i to the vertex colored by color i, for $i = 1, \ldots, n$. So, $i_0 = v_{j_0}$.
(2) Let j, $1 \leqslant j \leqslant n-1$, $j \neq j_0$, $i \neq j$ and $i \leqslant n-1$. From the above coloring, we define the (n)-coloring that assigns color j to v_n, color i to v_j and color n to v_i. So, we construct $(n-2)(n-2)$ colorings.
(3) For $j_0 + 1 \leqslant i \leqslant n-1$, from the first (n)-coloring, we define the (n)-coloring that assigns color j_0 to v_n, color i to v_{j_0} and color n tov_i. So, we construct $(n-1-j_0)$ colorings.
(4) For $j = 1, \ldots, n-1$, consider the n-coloring that assigns color j to v_n, color n to v_j and color i to $v_i \forall i \neq j, i \neq n$. There are $n-1$ colorings.

(5) W.l.o.g, suppose v_{n-1} and v_n are not adjacent. We consider the $(n-1)$-coloring that assigns color i to $v_i \forall i = 1, \ldots, n-2$, and color $n-1$ to v_{n-1} and v_n.
(6) For $i = 1, \ldots, n-2$, we define the $(n-1)$-coloring by switching color i and color $n-1$ from the above $(n-1)$-coloring. With this procedure we construct $n-2$ colorings.
(7) Any (j)-coloring, for each $j = \chi(G), \ldots, n-2$.
(8) Take the (j_0-1)-coloring. If v_{j_0} is colored with c, then switch color c with color j. Making this for all $j=1, \ldots, j_0-1$, $j \neq c$, we generate $j_0 - 2$ colorings.

It is easy to see that these $n^2 - \chi(G) - 1$ feasible solutions are affinely independent. Then, the valid inequality is a facet of $\mathscr{CP}$. □

Let $v \in V$, $N(v)$ the neighborhood of v and $\delta(v) = |N(v)|$. By combining $x_{vj} + x_{kj} \leqslant w_j$ for all $k \in N(v)$ we obtain the valid *Neighborhood* inequality

$$\sum_{k \in N(v)} x_{kj} + \delta(v) x_{vj} \leqslant \delta(v) w_j.$$

If r is the size of a maximum independent set in $N(v)$, no more than r vertices can be colored with the same color, then the above inequality can be strengthened and

$$\sum_{k \in N(v)} x_{kj} + r x_{vj} \leqslant r w_j$$

is a valid inequality stronger than the previous one. This is not a facet-defining inequality but it becomes very useful to *improve* the LP-relaxation. The original model has mn constraints $x_{ij} + x_{kj} \leqslant w_j$ and this size is difficult to handle for large and dense graphs. We replace the constraints in the original model with these new constraints. Despite this replacement relaxes the polytope, the computational experience shows it works better than the original formulation (see Section 4.1 for details).

3.4. Multicolor inequalities

Valid inequalities can be constructed by taking non-negative linear combinations of a linear inequality description of the set of solutions. In this way, we obtain new valid *weak* inequalities that are dominated by the originals. Nevertheless, we can obtain *stronger* valid inequalities by applying a strengthening procedure based on the combinatorial implications and integrality properties of the feasible solutions. Next we present some valid inequalities derived from the original constraints $x_{uj} + x_{vj} \leqslant w_j$.

3.4.1. Multicolor hole

Let $C_k = \{v_1, \ldots, v_k\}$ be a hole of size k and $\{j_1, \ldots, j_{k-1}\}$ a subset of colors such that $j_1 > j_i \ \forall i = 2, \ldots, k-1$. Consider the following $k-2$ valid inequalities:

$$x_{v_2 j_2} + x_{v_3 j_2} \leqslant w_{j_2},$$

$$x_{v_3 j_3} + x_{v_4 j_3} \leqslant w_{j_3},$$

$$\vdots$$

$$x_{v_{k-1} j_{k-1}} + x_{v_k j_{k-1}} \leqslant w_{j_{k-1}}$$

and

$$x_{v_k j_1} + x_{v_1 j_1} + x_{v_2 j_1} \leqslant 2 w_{j_1}.$$

Adding these inequalities, we obtain

$$x_{v_1 j_1} + \sum_{i=2}^{k-1} (x_{v_i j_{i-1}} + x_{v_i j_i}) + x_{v_k j_{k-1}} + x_{v_k j_1} \leqslant 2w_{j_1} + \sum_{i=2}^{k-1} w_{j_i}.$$

Any feasible solution that binds the last added inequality fixes the color to v_1, v_2 and v_k and satisfies $\sum_{i=2}^{k-1} w_{j_i} = k-2$ because $j_1 > j_i \ \forall i = 2, \ldots, k-1$. The other $k-3$ vertices can contribute at most with $k-3$. Then, the coefficient of w_{j_1} can be reduced to 1. This is a proper face of $\mathscr{CP}$ and it is a facet-defining inequality under the following conditions:

- $\forall v \in V \backslash C_k$, v has at least one non-adjacent vertex in $\{v_1, v_2, v_k\}$.
- A $(\chi(G))$-coloring lies on the face.

The proof is technically not more complicated than the previous one and runs along the same arguments.

3.4.2. Multicolor clique

Let $K = \{v_1, \ldots, v_p\}$ be a clique of size p, k such that $p \leqslant k \leqslant n-1$, and $Col = \{j_1, \ldots, j_{p-1}\} \subset \{1, \ldots, k-1\}$. We consider the following clique inequalities:

$$\sum_{i=1}^{p} x_{v_i j} \leqslant w_j \quad \forall j \in Col.$$

Adding them and considering that any coloring needs p colors to assign to K, we obtain the valid inequality

$$\sum_{i=1}^{p} \sum_{j=k}^{n} x_{v_i j} + \sum_{i=1}^{p} \sum_{j \in Col} x_{v_i j} \leqslant w_k + \sum_{j \in Col} w_j.$$

If $V \backslash K$ is not a clique and $\chi(G) + 1 \leqslant k \leqslant n-2$, it can be proved that it is a facet-defining inequality.

3.4.3. Multicolor path

Let $K_1, K_2, \ldots, K_r$ be cliques such that $K_i \cap K_j = \emptyset$ if $j \neq i-1, i+1$ and $|K_i \cap K_{i+1}| \leqslant 1$. Consider the colors $c_1, \ldots, c_r, c_{j_0}$ with $c_j < c_{j_0} \leqslant n-1 \forall j = 1, \ldots, r$. Combining *Clique* and *Block Color* inequalities, we obtain

$$\sum_{v \in K_1} x_{vc_1} + \sum_{v \in K_2} x_{vc_2} + \cdots + \sum_{v \in K_r} x_{vc_r} + \sum_{j=c_{j_0}}^{n} \sum_{v \in \bigcup_{i=1}^{r} K_i} x_{vj} \leqslant \sum_{i=1}^{r} w_{c_i} + R w_{c_{j_0}},$$

where $R = |\bigcup_{i=1}^{r} K_i|$.

Since $c_{j_0} > c_j \ \forall j = 1, \ldots, r$, the coefficient of $w_{c_{j_0}}$ can be reduced to $R-r$. Then, the following is a stronger valid inequality:

$$\sum_{v \in K_1} x_{vc_1} + \sum_{v \in K_2} x_{vc_2} + \cdots + \sum_{v \in K_r} x_{vc_r} + \sum_{j=c_{j_0}}^{n} \sum_{v \in \bigcup_{i=1}^{r} K_i} x_{vj} \leqslant \sum_{i=1}^{r} w_{c_i} + (R-r) w_{c_{j_0}}.$$

It is a proper face of $\mathscr{CP}$. In case that $K_i = \{v_i, v_{i+1}\}$ for $i = 1, \ldots, r$, we have

$$x_{v_1 c_1} + \sum_{i=2}^{k-1} x_{v_i c_{i-1}} + x_{v_i c_i} + x_{v_k c_{k-1}} + \sum_{j=c_{j_0}}^{n} \sum_{i=1}^{k} x_{v_i j} \leqslant \sum_{i=1}^{k-1} w_{c_i} + w_{c_{j_0}}.$$

We call it *Multicolor Path* inequality. This is a facet-defining inequality under the following conditions:

- $V \backslash \{v_1, \ldots, v_k\}$ is not a clique.
- If $v \in K_i$, there is a $(c_{j_0} - 1)$-coloring such that the assigned color to v is not c_i.
- A $(\chi(G))$-coloring lies on the face.

4. Implementation details of a cutting plane algorithm

In this section we describe the design and implementation of our cutting plane algorithm.

4.1. LP-relaxation

The first step in the development of a cutting plane algorithm is the definition of an initial LP-relaxation. We find that the linear relaxation of $\mathscr{CP}$ has too many adjacency constraints $x_{uj} + x_{vj} \leqslant w_j$ making its resolution too slow, mainly for medium and high density graphs.

We analyze several alternatives to reduce formulation size.

First, we consider the possibility of finding an edge clique cover, i.e., a set C of cliques such that for every edge $\{u, v\} \in E$, there is some clique $K \in C$ such that $u, v \in K$. In this way, adjacency constraints may be replaced by a clique inequality defined by each clique $K \in C$ and each color $j = 1, \ldots, n$. We experiment with several heuristics to find an edge clique cover. However, computational experiments do not show a significant solution time reduction for linear relaxations.

The second alternative we try is a weak LP-relaxation dropping all the adjacency constraints. We expected the addition of clique inequalities during the cutting plane algorithm somehow made up for the relaxation quality loss. Once more, computational experiments show it is not a good choice. The lower bound increase due to cutting plane addition do not prove to be very effective.

Finally, the initial relaxation showing the best behavior is the one resulting from replacing adjacency constraints by a weak version of the *Neighborhood* inequalities. The coefficient r of this inequality is replaced by the cardinal of a clique partition of $N(v)$ that we find by a greedy heuristic. As we mentioned above, this replacement relaxes the polytope but allows us to handle the model for larger graphs. It proves to be very convenient, since it shows a good balance between CPU time, memory requirements and lower bound increase.

4.2. Upper and lower bounds

Good upper as well as lower bounds on the value of the optimal solution are very important to keep the linear program reasonably sized. A lower bound is obtained by finding a maximal clique with a greedy heuristic. All the variables related to the vertices of the clique are fixed in the model by considering that the first n_cli (clique size) colors are assigned to each vertex. We apply the well-known DSATUR heuristic [2] to find a feasible solution. This solution gives an upper bound, $\hat{\chi}$, and allows us to eliminate model variables.

4.3. Separation algorithms

Given a fractional solution of $\mathscr{CP}$, we look for a set of constraints to cut it off. After adding these valid inequalities, we resolve the LP-relaxation. The separation phase is the central part of a cutting plane algorithm and efficient separation algorithms are crucial for the success of this approach.

Next, we describe the identification procedure of violated valid inequalities. In what follows, let (x^*, w^*) denote the fractional optimal solution to the current formulation.

4.3.1. Clique and multicolor clique inequalities

The *Clique* inequalities are used as cutting planes in many problems [1,12], and the separation problem is known to be NP-Hard. In the literature we can mainly find two strategies to detect them.

The first approach is to construct in a preprocessing phase a list of maximal cliques and keep it in a pool. At each separation round, this list is scanned in an attempt to find violated cuts. This simple procedure is very fast but, according to our experience, it cannot detect enough cuts. It has the advantage of looking for cliques once, but the clique construction does not exploit the information of the current fractional solution. We think this is the reason for the poor performance observed in our computational experience.

The second approach is to look for cliques by considering the current point and by applying a greedy heuristic. For each color j_0, the greedy criterion is to go for violated *Clique* inequalities, and it makes sense to do so by constructing an ordered list of the elements in $\{x^*_{ij_0} : i \in V \text{ and } x^*_{ij_0} < 1\}$ in decreasing value, where x^* denotes the current fractional

solution. If x_{ij_0} is a fractional variable, we initialize a clique with vertex i. Then, it is grown into a bigger clique trying to add other adjacent vertices following the ordered list. We perform several trials limited by an input parameter. In trial k, we choose the fractional variable $x_{i'j_0}$ such that vertex i' is the kth adjacent vertex to i in the ordered list. We add this vertex to the clique and then look in order in the rest of the list.

To avoid any additional computational effort, the clique found is also used to try a violated *Multicolor Clique* inequality. Since $\sum_{i\in V} x_{ij_0} > w_{j_0}$, then $\sum_{j=j_0}^{n}\sum_{i\in V} x_{ij} > w_{j_0}$ and thus the clique has good chances to violate the inequality. It remains to determine the set of $p-1$ colors, where p is the clique size. A greedy strategy is to look for colors where the associated clique inequality is also violated or it has the smallest slack value. Then, for each color j, with $1 \leqslant j \leqslant j_0 - 1$, we compute S_j where $S_j = \sum_{i\in\text{clique}} x_{ij} - w_j$. The first $p-1$ colors in order of decreasing S_j values are considered to attempt to find a violated inequality.

4.3.2. Block color inequality

These inequalities are handled by brute-force. In order to have chances to find a violated *Block Color* inequality, it is necessary that w_{j_0} be fractional. Then, for all j_0 such that $0 < w_{j_0} < 1$, we enumerate all inequalities and find those violated by the fractional current solution.

4.3.3. Multicolor path inequality

For each fractional variable w_k, the weight $c_{uv} = \max_{j=1,\ldots,k-1}\{x^*_{uj} + x^*_{vj} - w^*_j\} + \sum_{j=k}^{n}(x^*_{uj} + x^*_{vj})$ is associated with each edge $(u, v) \in E$. We compute for each vertex $v \in V$, the heaviest path in G by using a greedy procedure. For any $v \in V$, we initialize the candidate path P as v and make t_v trials. On trial j we extend the path by adding a vertex w characterized by being the jth maximum c_{vw} among all the adjacent vertices to v. Then, iteratively, vertices are added to the path by choosing the one adjacent to the last vertex added to the path, not previously included in P and with maximum weight.

The computational experience shows it is not convenient to allow a vertex belongs to many paths since the found associated *MultiColor Path* inequalities have similar support. To avoid this situation and according to our computational tests, it is forbidden to consider a vertex belonging to $n/10$ violated paths in future paths.

Moreover, long size paths have few chances to give violated inequality then, the above procedure is stopped when the size of the path is equal to an input parameter. Our computational experiments show that paths with length greater than 6 are not good candidates to be violated.

A path with weight greater than w^*_k corresponds to a violated *Multicolor Path* inequality.

4.3.4. Hole inequality

The separation procedure is based on the GLS algorithm proposed in [5] for the *Independent Set* problem. Given a graph $G' = (V', E')$, an auxiliary bipartite graph is constructed as follows. Let $B = (V_1 \cup V_2, E_B)$ be a bipartite graph where for each $v \in V'$ we include two vertices $v_1 \in V_1$ and $v_2 \in V_2$. If $(u, v) \in E'$, then (u_1, v_2) and $(v_1, u_2) \in E_B$. It is easy to see that a path P_{v_1} in B beginning in v_1 and ending in v_2, considered as a set of nodes, is a cycle C_v in G'.

In order to find violated hole inequalities, we consider j_0 such that $w^*_{j_0} > 0$ and $V' \subset V$ where $v \in V'$ if $x^*_{vj_0}$ is fractional. We build the associated bipartite graph B and we consider a weight $c_{u_1,v_2} = c_{v_1,u_2} = \max(0, w^*_{j_0} - x^*_{uj_0} - x^*_{vj_0})$ for each edge (u, v) of B. The weight of P_{v_1} is $\sum_{(u'z'')\in P_{v_1}} \max(0, w^*_{j_0} - x^*_{uj_0} - x^*_{zj_0})$.

Thus, if it is found the shortest path between v_1 and v_2, the cycle C_v will be a good candidate to link to a violated hole inequality. We compute the shortest path by using the well-known Dijkstra's algorithm.

4.4. Cut management scheme

A cutting plane algorithm constructs and solves a sequence of LP-relaxation. A set of violated constraints is added at each iteration and it is important to take care that the linear program does not become very big, and takes a long time to solve.

This problem can be avoided by removing previously introduced cuts that become not relevant to the current solution. As it is usual in cutting planes algorithms, we maintain a cut pool which contains the cuts generated so far in the algorithm that are either not included in the relaxation or subsequently dropped because they no longer appear to be active.

This cut pool is very useful to memory management and it can also be considered as an auxiliary mechanism for performing separation since the pool cuts can be checked quickly for violation. Since we use a heuristic procedure to find violated constraints, it can be possible that we are not able to detect some inequalities that were detected before. Whenever the LP has been reoptimized, we first check all cuts in the cut pool, and reoptimize the LP, if more than 200 violated cuts have been found. Otherwise we call our separation routines looking for new violated inequalities.

5. Computational experience

In this section we present our computational experience with the cutting plane algorithm. We have performed the experiments on a Sun ULTRA workstation and the times are reported in seconds. The code is implemented in C++ using the ABACUS framework [6] and CPLEX 8.1 LP solver [4]. Our goal is to evaluate the lower bound improvement of the LP-relaxation when we strengthen it by adding valid inequalities that have already been characterized for the polytope. We have introduced several classes of valid inequalities and have given conditions under which they are facet-defining. However, facetness does not necessarily guarantee good performance if the inequality is considered as a cutting plane. Even though the separation procedure is successful, the cuts may not help to increase the lower bound.

5.1. Cutting planes

An indirect way of evaluating the quality of a cutting plane is to observe the increase produced in the lower bound when it is added to the LP-relaxation. Larger increases mean better constraints because they define deeper cuts in the relaxation polytope. However, a right balance between different aspects has to be considered.

If the added cuts are dense, they increase memory requirements and may slow down the solution of the LP's. Besides that, if the separation routine for a class of inequalities is computationally expensive in relation to the lower bound increase when they are added to the LP's, it is not worthwhile including them in the algorithm.

We conduct experiments to determine a good cut combination scheme by considering several combinations of cut families.

To compare the combinations, a cutting plane algorithm is applied for 50 rounds on random instances. $G(n, p)$ is a random graph of n vertices and an edge between each pair of vertices with independent probability p. We use random graphs of 125 vertices with low (less than 0.3), medium (between 0.4 and 0.6) and high (more than 0.7) density.

We observe the evolution of the lower bound, *LB*, and take into account the CPU time needed to achieve it. The experiments show that any combination achieves the same lower bound at the end of 50 rounds, except combinations that exclude *Clique* cuts. The cutting plane performance is mainly due to the addition of these cuts.

The different combinations considered are tested on eight instances for each density. See references for each cut combination in Table 1. Table 2 shows the average initial gap percentage $(100(\hat{\chi} - n\_cli)/\hat{\chi})$, final gap percentage $(100(\hat{\chi} - LB)/\hat{\chi})$, CPU time and the round where the best lower bound is achieved.

For low density graphs, there is no significant increase of the initial lower bound and the final lower bound is achieved on the first rounds. In many cases, the algorithm stops before 50 rounds because it is not possible to find violated inequalities. The CPU time increases when *Hole* cuts are included in the cut scheme.

For medium density graphs, the initial lower bound increases during the first iterations, then the objective function does not change from one round to the next until the middle, and it does not change again until the end. There are no instances where the algorithm stops because it cannot find cuts. The addition of *Hole* and *Multicolor Clique* cuts results in CPU time increase and a reduction of rounds to achieve the final lower bound.

Table 1
Family cuts combination

C_1	Clique
C_2	Clique + Block Color + MultPath
C_3	Clique + Block Color + MultPath + MultCli
C_4	Clique + MultCli + Hole
C_5	Clique + Block Color + MultPath + MultCli + Hole

Table 2
Family cuts combinations on random graphs

	Low density		Medium density		High density	
	Initial gap	Final gap	Initial gap	Final gap	Initial gap	Final gap
	47	35	52	38	43	27
	Time	Round	Time	Round	Time	Round
C_1	24	11	215	29	369	35
C_2	28	11	214	28	386	35
C_3	26	11	249	25	484	37
C_4	39	11	250	26	426	32
C_5	37	11	265	26	524	36

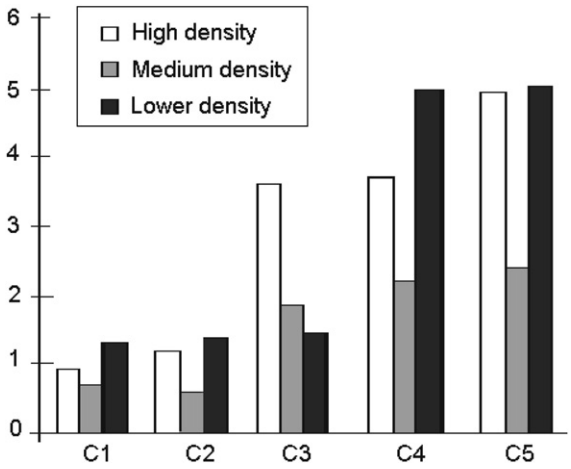

Fig. 1. Average GAP.

For high density graphs, the lower bound increases significantly during the algorithm, even in the last rounds. The CPU time increases significantly when *Hole* or *Multicolor Clique* cuts are included.

The experiments show that there is no combination with the best performance for all instances. To get a more direct comparison, Fig. 1 gives a summary of the above results by using the following measure of efficiency. For each cut combination C_j, we obtain TC_j the average over the eight instances of the ratio of the difference between the CPU time for this cut combination and the CPU time for the best cut combination for that instance over the best cut combination. If $Time_i^{C_j}$ is the CPU time required to achieve the lower bound with cut combination C_j on instance i and $BestTime_i = \min\{Time_i^{C_j},\ j = 1, \ldots, 5\}$, then

$$P_j = \sum_{i=1}^{8} (Time_i^{C_j} - BestTime_i)/BestTime_i.$$

The lower the value of PC_j, the better the combination C_j. There is no clear computational winner among the combinations considered. However, since combinations without *Multicolor Clique* and *Hole* inequalities are generally superior in CPU time, we think it is worth including these inequalities when the algorithm cannot find any other family cuts. The scheme using *Clique*, *Block Color* and *Multicolor Path* inequalities is the best for medium density graphs, and its behavior is good enough for the other densities.

Table 3
Separation time vs. total time

	Low density			Medium density			High density		
	Total	Separation	%	Total	Separation	%	Total	Separation	%
C_1	47.2	6.6	14	254.8	9.2	4	472.6	11	2
C_2	49.6	6.8	14	255	10.8	4	481	12.4	3
C_3	51.8	8	15	298.8	11.2	4	564.2	17.2	3
C_4	66.2	15.8	24	312	23.6	8	661.8	35.8	5
C_5	69.2	18	26	328.4	24.8	8	629.4	38	6

Table 4
Separation efficiency

Dens. (%)	Clique	MultCli	MultPath	Block color	Hole
0.9	31.4	31.4	80.0	0.8	29.5
	29.6	3.9	*	*	26.4
	27.6	0.0	*	*	21.3
	25.9	0.0	*	*	13.6
	21.5	11.6	7.9	*	11.5
0.7	17.6	17.6	76.0	2.2	83.2
	20.1	0.3	12.6	*	74.6
	22.0	0.0	*	*	73.3
	19.9	0.6	2.0	*	62.1
	20.2	0.0	*	*	53.3
0.6	16.8	16.8	70.0	3.9	90.4
	20.9	0.2	10.6	*	88.5
	21.5	0.0	21.6	*	83.0
	20.2	0.3	8.0	*	73.4
	20.1	0.0	*	*	69.8
0.5	20.2	14.6	70.0	1.3	94.9
	24.2	0.1	24.0	*	91.8
	20.7	3.5	40.6	0.0	91.1
	22.4	0.0	*	*	87.0
	20.1	1.5	0.2	*	82.0
0.4	28.2	3.5	80.0	0.0	95.4
	26.5	0.0	*	*	94.1
	24.7	0.0	*	*	89.3
	19.9	8.1	53.3	0.0	91.2
	23.4	0.0	*	*	85.3
0.3	33.6	2.1	80.0	*	89.3
	27.9	1.9	80.0	*	87.4
	26.5	0.5	100.0	*	86.4
	25.5	0.0	*	*	85.4
	23.6	0.0	*	*	80.8

5.2. Separation time

A factor that does not influence our choice of family cuts is separation procedure time. Table 3 shows for each combination, the total average time of the eight instances in the 50 rounds of the cutting plane, the average separation process time, and the corresponding percentage over the total average time.

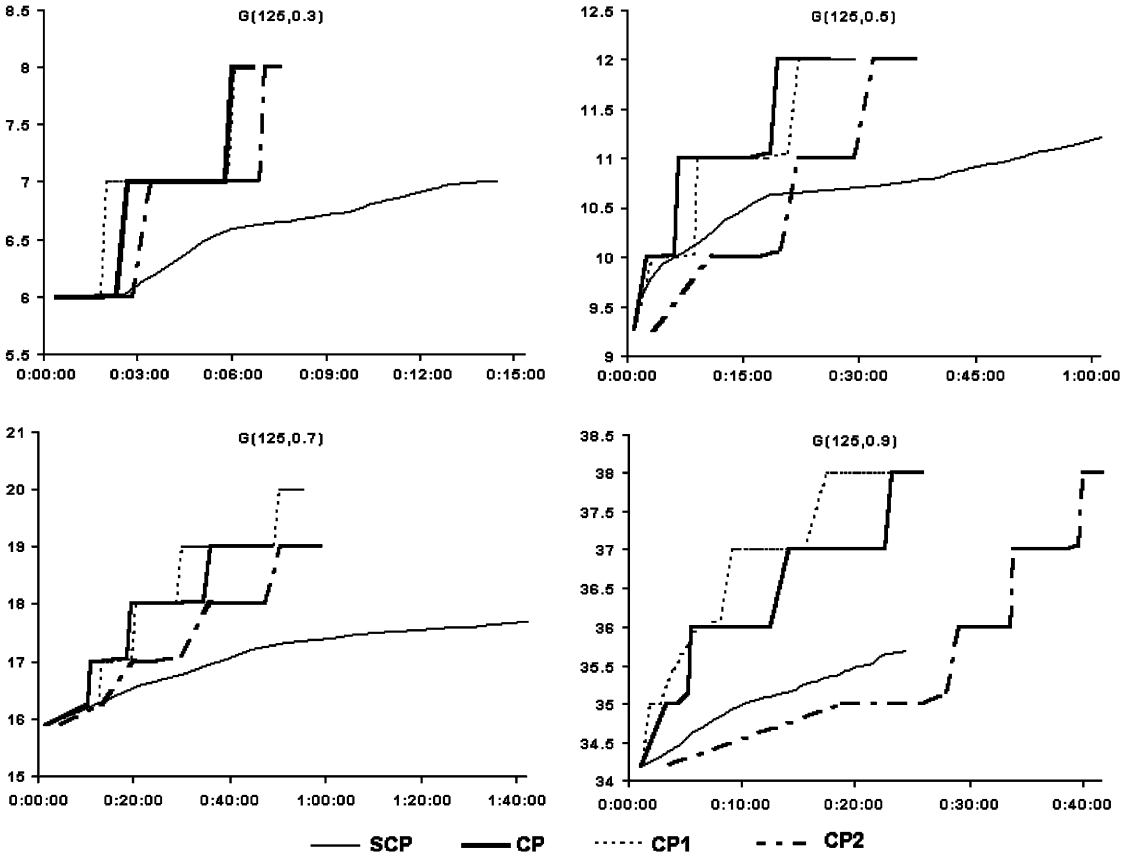

Fig. 2. Lower bound evolution on different relaxations.

It can be observed the ratio of overall separation time is not significant, although it increases as graph's density decreases.

The process of separating *Hole* inequalities is a determining factor for the increase of total separation time proportion. The separation of *Multicolor Cliques* is also an important factor, although it does not have such a strong influence. *Block Color* and *Multicolor Path* separation procedures do not cause a significant increase in the time. We consider separation algorithms' time is not a decisive factor to choose one combination over the other.

5.3. Separation algorithm efficiency

To evaluate the efficiency of separation algorithms to find violated inequalities, we take as measure the percentage of violated cuts over the number of cuts analyzed by each procedure. We run five rounds of the cutting plane algorithm on random graphs of 125 vertices and different densities. Table 4 presents the results.

Block Color and *Multicolor Path* inequalities present the possibility of being violated only if variables w_k, which are in the corresponding restriction, present a fractional value. Otherwise, the separation algorithm is not called. An asterisk ($*$) shows this condition in the table.

The *Multicolor Path* inequality separation process has a good efficiency rate, regardless the graph density. *Block Color* inequalities are explored by brute-force and the percentage of violated ones is very low. Nevertheless, since the procedure is very fast and its inclusion shows an improvement of algorithm's performance, such inclusion is justified.

The most inefficient separation process is the *Multicolor Clique* inequalities. It is a heuristic process and the percentages obtained may lead to the conclusion that the strategy used might not be a good one. However, experiments with

Table 5
Instances solved by the initial heuristics

Problem	n	m	χ
DSJR500_1	500	12458	12
inithx.i.1	864	18707	54
inithx.i.2	645	13979	31
inithx.i.3	621	13969	31
le450_25a	450	8260	25
le450_25b	450	8263	25
le450_5c	450	9803	5
mulsol.i.1	197	3925	49
mulsol.i.2	188	3885	31
mulsol.i.3	184	3916	31
mulsol.i.4	185	3946	31
mulsol.i.5	185	3973	31
school1	385	19095	14
school1_nsh	352	14612	14
zeroin.i.1	211	4100	49
zeroin.i.2	211	3541	30
zeroin.i.3	206	3540	30
anna	138	493	11
david	87	406	11
homer	561	1629	13
huck	74	301	11
jean	80	254	10
games120	120	638	9
miles250	128	387	8
miles500	128	1170	20
miles750	128	2113	31
queen8_12	96	1368	12
qg_order30	900	26100	30

graphs with fewer vertices, and a more thorough search have not shown better results. Everything seems to indicate they are not inequalities frequently violated by the optimal solution of the LP relaxations.

Hole separation efficiency increases as graph density decreases, whereas *Clique* separation efficiency maintains an even rate regardless of density.

We use several criteria to evaluate valid inequalities: the lower bound evolution, separation algorithm times and efficiency rate. All are related and lead us to conclude that they are not in conflict. The presence of *Clique* inequalities is essential, which along with *Block Color* and *Multicolor Path* combined, provide a good cutting plane scheme. *Multicolor Clique* and *Hole inequalities* do not present any feature that justifies their inclusion except when no other cuts can be found.

5.4. Comparing relaxations

In Section 2 we define four polyhedra: $\mathscr{SCP}$, $\mathscr{CP}$, $\mathscr{CP}1$ and $\mathscr{CP}2$. Polyhedron $\mathscr{SCP}$ is associated with the classical formulation of the coloring problem. Polyhedra $\mathscr{CP}$, $\mathscr{CP}1$ and $\mathscr{CP}2$ correspond to the set of solutions provided by the three models we presented with different criteria for eliminating symmetry.

Some facet-defining inequalities derived for $\mathscr{SCP}$ in [3] are shared in $\mathscr{CP}$, for instance, *Clique* inequalities.

From the polyhedral study's perspective, we have already pointed out the difficulty polyhedra $\mathscr{CP}1$ and $\mathscr{CP}2$ present. However, since both are included in $\mathscr{CP}$, the valid inequalities arising from our study are also valid for these polyhedra. In particular, in a number of instances we find out that *Clique* inequalities are also facet-defining inequalities for both polyhedra.

Moreover, based on our experience, the addition of *Clique* inequalities in the cutting plane algorithm is essential to improve the lower bound which provides the optimum linear relaxation value. Therefore, we consider it is reasonable to use these inequalities in a cutting plane algorithm to compare the different relaxations.

Table 6
Hard DIMACS instances

Problem	n	m	n_cli	$\hat{\chi}$	χ
DSJC500_1	500	12 458	5	15	?
DSJC1000_1	1000	49 629	6	26	?
DSJC1000_5	1000	249 826	14	116	?
latin_square_10	900	307 350	90	129	?
le450_15a	450	8168	15	17	15
le450_15b	450	8169	15	17	15
le450_15c	450	16 680	15	24	15
le450_15d	450	16 750	15	23	15
le450_25c	450	17 343	25	28	25
le450_25d	450	17 425	25	28	25
le450_5a	450	5714	5	9	5
le450_5b	450	5734	5	9	5
le450_5d	450	9757	5	10	5
queen10_10	100	2940	10	12	?
queen11_11	121	3960	11	14	11
queen12_12	144	5192	12	15	?
queen13_13	169	6656	13	16	13
queen14_14	196	8372	14	17	?
queen15_15	225	10 360	15	18	?
queen16_16	256	12 640	16	20	?
queen8_8	64	728	8	10	9
queen9_9	81	1056	9	11	10
mug88_1	88	146	3	4	4
mug88_25	88	146	3	4	4
mug100_1	100	166	3	4	4
mug100_25	100	166	3	4	4
abb313GPIA	1557	46 546	8	10	?
will199GPIA	701	6772	6	7	7
2-FullIns_5	852	12 201	4	7	?
4-FullIns_5	4146	77 305	6	9	?
5-FullIns_3	154	792	7	8	8
wap01	2368	110 871	41	46	?
wap02	2464	111 742	40	45	?
wap03	4730	286 722	40	56	?
wap04	5231	294 902	40	50	?
wap05	905	43 081	50	51	?
wap06	947	43 571	40	44	?
wap07	1809	103 368	40	46	?
wap08	1870	104 176	40	47	?
qg_order40	1600	62 400	40	42	40
qg_order60	3600	212 400	60	63	60

We run the algorithm on random graph of 125 vertices and 0.3, 0.5, 0.7 and 0.9 densities. In the four formulations we fix the color of the maximal clique found by our heuristic. The restrictions eliminating symmetric solutions are taken into account for the remaining vertices and for colors having a label greater than the clique size. We perform 50 iterations of a cutting plane algorithm with *Clique* inequalities. Fig. 2 shows the evolution of the objective function for the four relaxations.

$\mathscr{SCP}$ linear relaxation is the one presenting the worst performance and it does not achieve the same lower bound values. The large number of symmetrical solutions found in the polyhedron is the reason for the slow progress in the objective function increase, and also the delay time for each cutting plane algorithm iteration.

The other three relaxations show a similar performance regarding lower bound's behavior. Nevertheless, the relaxation resolution time in each iteration is longer for $\mathscr{CP}2$. This is logical since $\mathscr{CP}2$ has $n\_cli(\hat{\chi} - n\_cli)$ restrictions more than $\mathscr{CP}$. This CPU time difference increases with graph density.

Table 7
Lower bounds DIMACS instances

Problem	n	m	n_cli	$\hat{\chi}$	χ	$\mathscr{CP}$		$\mathscr{CP}1$	
						Bound	Time	Bound	Time
DSJC125_1	125	736	4	5	5	**5**	1	**5**	1
DSJC125_5	125	3891	9	20	17	12	77	12	73
DSJC125_9	125	6961	32	47	42	42	354	41	376
DSJC250_1	250	3218	4	9	8	5	11	5	11
DSJC250_5	250	15 668	11	36	?	14	3339	14	3523
DSJC250_9	250	27 897	37	88	?	48	3605	47	880
DSJC500_5	500	62 624	12	63	?	13	538	13	501
DSJC500_9	500	112 437	47	161	?	59	5870	59	5344
DSJR500_1C	500	121 275	72	87	?	80	4470	79	2093
DSJR500_5	500	58 862	117	131	?	119	1211	119	1262
DSJC1000_9	1000	449 449	55	301	?	66	4546	65	1497
fpsol2_i_1	496	11 654	55	65	65	**65**	8	**65**	8
fpsol2_i_2	451	8691	29	30	30	**30**	1	**30**	1
fpsol2_i_3	425	8688	29	30	30	**30**	1	**30**	1
miles1000	128	3216	41	42	42	**42**	0	**42**	0
miles1500	128	5198	71	73	73	**73**	0	**73**	0
ash331GPIA	662	4185	3	4	4	**4**	48	**4**	48
ash608GPIA	1216	7844	3	4	4	**4**	692	**4**	692
ash958GPIA	1916	12 506	3	5	4	4	4236	4	4236
1-Insertions_4	67	232	2	5	5	3	2	3	2
1-Insertions_5	202	1227	2	6	5	3	0	3	0
1-Insertions_6	607	6337	2	7	7	3	3	3	3
2-Insertions_3	37	72	2	4	4	3	0	3	0
2-Insertions_4	149	541	2	5	4	3	0	3	0
2-Insertions_5	597	3936	2	6	5	3	3	3	3
3-Insertions_3	56	110	2	4	4	3	0	3	0
3-Insertions_4	281	1046	2	5	4	3	0	3	0
3-Insertions_5	1406	9695	2	6	6	3	61	3	61
4-Insertions_3	79	156	2	4	4	3	0	3	0
4-Insertions_4	475	1795	2	5	4	3	2	3	2
1-FullIns_3	30	100	3	4	4	**4**	0	**4**	0
1-FullIns_4	93	593	3	5	5	4	0	4	0
1-FullIns_5	282	3247	3	6	6	4	0	4	0
2-FullIns_3	52	201	4	5	5	**5**	0	**5**	0
2-FullIns_4	212	1621	4	6	6	**6**	4	**6**	4
3-FullIns_3	80	346	5	6	6	**6**	0	**6**	0
3-FullIns_4	405	3524	5	7	7	6	4	6	3
3-FullIns_5	2030	33 751	5	8	8	6	292	6	292
4-FullIns_3	114	541	6	7	7	**7**	0	**7**	0
4-FullIns_4	690	6650	6	8	8	7	16	7	16
5-FullIns_4	1085	11 395	7	9	8	8	55	8	55

6. Final results and conclusions

Finally, we report our results on DIMACS instances. We have performed the experiments on a Sun ULTRA workstation and the times are reported in seconds (Machine Benchmarks User time: r100.5 = 0 s, r200.5 = 0 s, r300.5 = 1 s, r400.5 = 6 s, r500.5 = 24 s).

There are instances where the lower and the upper bound obtained with the initial heuristics are equal, then the cutting plane algorithm is not used for these graphs. These instances are showed in Table 5.

Instances where there is no lower bound increase because the algorithm terminates, either because the CPU time limit (2 h) is reached or no further cutting planes are found, are reported in Table 6. The information provided in the table is the number of vertices and edges, the size of the clique and the initial upper bound found by our heuristic, and

the chromatic number ("?" means unknown). The initial lower bound is equal to the chromatic number for most of these graphs, then it is not possible for the algorithm to improve the lower bound.

Table 7 reports instances where the lower bound is improved. The information provided in this table is the number of vertices and edges, the chromatic number, the size of the clique found by our heuristic and the initial upper bound. The last columns report the lower bound found for our algorithm and the times to achieve it by using LP-relaxations of $\mathscr{CP}$ and $\mathscr{CP}1$, respectively. The results show that the algorithm obtains a significant improvement of the initial lower bounds with both relaxations, specially on graphs where the chromatic number is much larger than the maximum clique size. Moreover, there are instances where our algorithm is able to solve to optimality since the initial gap is closed.

There is no relaxation with the best performance for all instances. The CPU time difference to achieve the same lower bound between the relaxations is not as significant to prefer one relaxation over the other. There are some instances where the lower bound obtained with $\mathscr{CP}$ relaxation is improved although more CPU time is required.

This behavior confirms that our approach is a good strategy to obtain lower bounds when compared to the classical maximum clique size, except in instances deliberately constructed to be hard to color.

This conclusion points out to interesting research avenues. First, further work could be directed to characterize new valid inequalities and implement separation algorithms for them, in order to improve the cutting plane algorithm. Finally, the cutting plane procedure can be included in a branch-and-cut algorithm to obtain a competitive exact algorithm for graph coloring.

References

[1] R. Borndörfer, Aspects of set packing, partitioning and covering, Ph.D. Dissertation, Technischen Universitat Berlin, 1999.
[2] D. Brélaz, New methods to color the vertices of a graph, Comm. ACM 22 (1979) 251–256.
[3] P. Coll, J. Marenco, I. Méndez-Díaz, P. Zabala, An integer programming model for the graph coloring problem, in: Annals of X CLAIO, Mexico, 2000.
[4] CPLEX Linear Optimization 8.1 with Mixed Integer & Barrier Solvers, ILOG, 1997–1998.
[5] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, Berlin, 1988.
[6] M. Junger, S. Thienel, The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization, Software Practice and Experience 30 (11) (2000) 1325–1352.
[7] R. Karp, Reducibility among combinatorial problems, in: R. Miller, J. Thatcher (Eds.), Complexity of Computer Computations, 1972, pp. 85–104.
[8] M. Kubale, B. Jackowski, A generalized implicit enumeration algorithm for graph coloring, Comm. ACM 28 (4) (1985) 412–418.
[9] A. Mehrotra, M. Trick, A column generation approach for graph coloring, INFORMS J. Comput. 8 (4) (1996) 344–353.
[10] I. Méndez-Díaz, P. Zabala, A Polyhedral Approach for Graph Coloring, Electronic Notes in Discrete Mathematics, vol. 7, 2001.
[11] G.L. Nemhauser, L. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, 1988.
[12] M.W. Padberg, On the facial structure of set packing polyhedral, Math. Programming 5 (1973) 199–215.
[13] T.J. Sager, S. Lin, A pruning procedure for exact graph coloring, ORSA J. Comput. 3 (3) (1991) 226–230.
[14] E.C. Sewell, An improved algorithm for exact graph coloring, in: D. Johnson, M. Trick (Eds.), DIMACS, 1996.
[15] L. Wolsey, Integer a Programming, Wiley, New York, 1998.