Compact and non-compact formulations for the Dominated Coloring Problem

Dilson Lucas Pereira
Departamento de Computação Aplicada, Universidade Federal de Lavras
Lavras, Minas Gerais, Brazil
dilson.pereira@ufla.br

Abilio Lucena
Programa de Engenharia de Sistemas e Computação, Universidade Federal de Minas Gerais
Rio de Janeiro, Rio de Janeiro
Brazil
abiliolucena@pesc.ufrj.br

Alexandre Salles da Cunha
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais
Brazil
acunha@dcc.ufmg.br

ABSTRACT

Assume that a proper coloring (PC) is available for a given undirected graph $G = (V, E)$. Assume as well that all vertices in any of the color classes at hand are simultaneously dominated by a same vertex. Such a PC is then called a dominated coloring (DC) of G and the least number of color classes a DC might have, $\chi_{dom}(G)$, is called the dominated chromatic number of G. In turn, the problem of finding a DC of cardinality exactly $\chi_{dom}(G)$ is called the Dominated Coloring Problem (DCP). In this paper, we investigate two Integer Programming formulations for DCP and accompanying Branch-and-bound algorithms. One formulation relies on the concept of representatives and is strengthened with a set of valid inequalities that substantially improves its Linear Programming Relaxation bounds for sparse graph instances. The other is a set covering (SC) formulation that assigns binary variables to the maximal cliques in the open neighborhoods of every individual vertex of G. Our preliminary numerical results suggest that the clique formulation is, on average, 47% stronger than the formulation by representatives. Additionally, its corresponding Branch-and-bound algorithm also provides substantially better results, despite the fact that, at least for the moment, we explicitly enumerate and keep all necessary cliques, as opposed to resorting to a properly defined restricted master problem, in a column generation scheme.

1 INTRODUCTION

Let $G = (V, E)$ be a undirected graph with $n = |V|$ vertices and $m = |E|$ edges. The (open) neighborhood of $i \in V$, $N(i) = \{j \in V : \{i, j\} \in E\}$, corresponds to the set of vertices that share an edge of E with i. Vertex $i \in V$ dominates a set $S \subset V$ if and only if $S \subseteq N(i)$ applies. A proper coloring of G, or simply a coloring, is a function $c : V \to \{1, 2, \ldots, n\}$ such that no pair of adjacent vertices are colored with the same color. A *color class* $C_i = \{j \in V : c(j) = i\}$ corresponds to all vertices of G which are assigned a same color and consequently defines a stable set of G. A k-coloring of G is a partitioning of V into k color classes. Additionally, a k-coloring of G is *dominated* if and only if $C_i \subseteq N(u)$ holds for some $u \in V$, for every color class C_i in the partitioning. Assume, from now on, that when we say a vertex of V *dominates a color class* it implies that such a vertex dominates, i.e., is a neighbor to, all vertices in that class. The Dominated Coloring Problem (DCP) then asks a dominated k-coloring of G with k as small as possible, i.e., one for which $k = \chi_{dom}(G)$ applies. Note that according to such a definition, color classes may eventually contain a single vertex, provided G has no leaves.

Vertex coloring problems [8] are intensively investigated in the literature. This applies mostly due to their widespread theoretical and practical applicability and also to the fact that they are generally NP-complete. More recently, problems combining coloring and domination demands started to be investigated, DCP among them. In particular, DCP was introduced in [10] where its decision version was proven NP-Complete for arbitrary graphs with $\chi_{dom}(G) \geq 4$. Additionally, a polynomial time algorithm for recognizing graphs with $\chi_{dom}(G) \leq 3$ is also proposed in [10]. Besides its intrinsic theoretical importance, DCP finds applications in social networks [5], in genetic networks [6] for finding minimum groups of proteins satisfying some given types of interactions and in the interconnection of computer networks [7].

A problem that is closely related to DCP is the *Dominator Coloring Problem* (DtorCP) [4, 6]. A coloring is said to be feasible for it if every vertex of G dominates at least one color class, possibly its own color class (i.e., dominates all vertices in that class). Accordingly, among other differences, dominance requirements differ from DCP to DtorCP.

To the best of our knowledge, no Integer Programming (IP) formulations or IP based exact solution approach appears to exist for DCP. In this paper, we introduce two IP formulations, valid inequalities and Branch-and-bound algorithms for the problem. Apart from this introduction, the paper contains four additional sections. In Section 2, we present the formulations and in Section 3 we give some implementation details of our Branch-and-bound algorithms for solving them. Some preliminary computational results are reported in Section 4. We conclude the paper in Section 5, where we highlight our main findings and offer some directions for future research.

2 INTEGER PROGRAMMING FORMULATIONS

The presentation that follows relies on some standard notation in Graph Theory [2], summarized next. The closed neighborhood of i is $N[i] := N(i) \cup \{i\}$. The anti-neighborhoods of i are $\overline{N}(i) = V \setminus (N(i) \cup \{i\})$ and $\overline{N}[i] = V \setminus N(i)$. Pairs of vertices $\{i, j\}$ that are not neighbors in G are identified by the end points to the edge $e = \{i, j\} \in \overline{E}$, where $\overline{E}$ is the complement of E. Accordingly, $\overline{G}$ is the graph $(V, \overline{E})$, that complements G. We assume that G is connected since otherwise DCP would then decompose into various, smaller, DCPs, one for every connected component of the graph. We also assume that G has no leaves, since otherwise any leaf $i \in V$ might be colored with the same color of a vertex j adjacent to the only neighbor of i, p ($p \in N(i) \cap N(j)$), without increasing the chromatic number. Given a set $S \subseteq V$, the subgraph induced by S is $G[S] = (S, E(S))$, where $E(S)$ denotes the set of edges of E with both endpoints in S. Likewise, $\overline{G}[S] = (S, \overline{E}(S))$ denotes a subgraph of $\overline{G}$, where $\overline{E}(S) = \{\{i, j\} \in S : \{i, j\} \in \overline{E}\}$. A clique of G is a set $S \subseteq V$ such that $G[S]$ is complete, i.e., all vertices in S are pairwise neighbors in G. Thus, a stable set of G corresponds to a clique of $\overline{G}$, after we extend the clique definition to subsets of vertices of size 2 and 3, i.e., respectively edges and triangles. As usual, we assume that $\mathbb{B}$ is the set $\{0, 1\}$.

2.1 A formulation by representatives

The formulation uses two sets of decision variables, namely:

- $\mathbf{x} = \{x_{ij} \in \mathbb{B} : i \in V, j \in \overline{N}[i], j \geq i\}$. If $x_{jj} = 1$ applies, vertex j is colored with color $c(j) = j$, and is said to represent C_j, the color class containing it. Conversely, if $x_{jj} = 0$ holds, vertex j must belong to a color class represented by one of its anti-neighbors. In more detail, variable x_{ij} for $i < j$ is used to indicate whether or not vertex j belongs to the color class C_i. In case it does, $x_{ij} = 1$ must hold and both vertices are colored identically with color $c(i) = c(j) = i$ and the two vertices therefore belong to C_i. Otherwise, $x_{ij} = 0$ applies. Note that, in order to break formulation symmetries, variables x_{ij} are not assigned to anti-neighbors i, j such that $i > j$.
- $\mathbf{z} = \{z_u^p \in \mathbb{B} : u \in V, p \in N(u)\}$. Variable z_u^p is used to indicate whether or not $u \in V$ is the vertex chosen to dominate C_p, an eventual color class represented by p. In case it is the chosen vertex, all vertices colored with color $c(p) = p$ must be neighbors of u. Thus, if $z_u^p = 1$, $C_p \subseteq N(u)$ must hold.

DCP can be formulated as the following IP

$$\chi_{dom}(G) = \min \left\{ \sum_{i \in V} x_{ii} : (\mathbf{x}, \mathbf{z}) \in \mathcal{P}_r \cap (\mathbb{B}^{d_x}, \mathbb{B}^{d_z}) \right\}, \quad (1)$$

where the polyhedral set $\mathcal{P}_r$ is defined by constraints (2)-(7) and d_x and d_z respectively denote the dimension of $\mathbf{x}$ and $\mathbf{z}$. Accordingly, $\mathcal{P}_r$ is thus formulated as

$$\sum_{v \in \overline{N}[u], v \leq u} x_{vu} = 1 \qquad u \in V \quad (2)$$

$$x_{pi} + x_{pj} \leq x_{pp} \qquad \{i, j\} \in E, p \in \overline{N}(i) \cap \overline{N}(j), \; p < i, p < j \quad (3)$$

$$\sum_{u \in N(v)} z_u^v = x_{vv} \qquad v \in V \quad (4)$$

$$z_u^v + x_{vt} \leq x_{vv} \qquad u \in V, v \in N(u), \; t \in \overline{N}(u) \cap \overline{N}(v), v < t \quad (5)$$

$$x_{ij} \in [0, 1] \qquad i \in V, j \in \overline{N}[i], j \geq i \quad (6)$$

$$z_u^p \in [0, 1] \qquad u \in V, p \in N(u) \quad (7)$$

Constraints (2) enforce that all vertices of G are assigned to a color class and the number of classes is minimized by the objective function in (1). In turn, constraints (3) imply that a vertex p cannot represent the color of a anti-neighbor, say i, unless p is the representative of its own color class. They also enforce that no pair of neighbors i and j can be represented by a common anti-neighbor p.

The fact that every color class must be dominated by a vertex is ensured by constraints (4) and (5). Notice that constraints (4) imply that if v represents a color class, there must be another vertex u, in the neighborhood of v, such that $z_u^v = 1$ applies. Now, under the assumption that $z_u^v = 1$ holds, constraints (5) forbid vertex v to represent a vertex t outside the neighborhood of u, the vertex chosen to dominate the color class C_v. Note that a color class C_v may well be dominated by more than two neighbors of v. The formulation, however, requires that only one variable, say $z_u^v : u \in N(v)$, assumes value one in such cases.

Formulation $\mathcal{P}_r$ could be reinforced, for instance, by replacing inequalities (3) by its stronger *clique* form:

$$\sum_{i \in Q, i > p} x_{pi} \leq x_{pp}, p \in V, Q \text{ a maximal clique of } G[\overline{N}(p)]. \quad (8)$$

It can also be strengthened with the following set of valid inequalities:

$$x_{kt} \leq \sum_{i \in N(t) \cap N(k)} z_i^k, k \in V, t \in \overline{N}(k), t > k. \quad (9)$$

If $x_{kt} = 0$ applies, inequality (9) is trivially valid. Otherwise, if $x_{kt} = 1$ holds, color k must be dominated by a vertex in the open neighborhood of both k and t. Hence, inequalities (9) are valid for DCP.

For the moment, we do not use the stronger set (8) to enforce proper colorings of G. Thus, denote by $\mathcal{P}_r^+$ the polyhedral region $\mathcal{P}_r$ reinforced with constraints (9) only, i.e., $\mathcal{P}_r^+$ is defined by constraints (2)-(7) and (9).

2.2 A formulation based on cliques of $\overline{G}[N(u)]$

From the seminal work of Mehrotra and Trick [9] onwards, it became a common practice to use maximal cliques to formulate the vertex coloring problem and variants of it.

As a result, column generation based Branch-and-bound algorithms became the standard approach for solving coloring problems. Our additional DCP formulation complies with this standard. It is a set covering one that assigns binary variables to a subset of the cliques of $\overline{G}$. In doing so, the formulation makes sure that every vertex of G must be part of at least one clique. Given that color classes must be dominated by at least one vertex, one must only consider cliques contained in $\{\overline{G}[N(u)] : u \in V\}$.

Our formulation makes a clear distinction between cliques of sizes 3 or greater and cliques of size 2. As we shall discuss later on, among all cliques of size at least 3 for the graphs $\{\overline{G}[N(u)] : u \in V\}$, we can restrict ourselves to maximal ones.

Prior to introducing the formulation, additional notation is required. Firstly, let $\mathcal{Q}_u$ denote the set of all maximal cliques of $\overline{G}[N(u)]$ with at least 3 vertices. Accordingly, denote by $\mathcal{Q} = \bigcup_{u \in V} \mathcal{Q}_u$ the set of all these cliques in $\overline{G}[N(u)]$. Additionally, define the set $\overline{\delta}(u)$ as the subset of edges of the complement graph $\overline{G}$ that are incident to u. Furthermore, define $\overline{\delta}_R(u) = \{\{u, p\} \in \overline{\delta}(u) : u, p \in N(v) \text{ for some } v \in V\}$ as the subset of edges of $\overline{\delta}(u)$ whose endpoints share a common neighbor in E. Finally, define $\overline{E}_R = \bigcup_{u \in V} \overline{\delta}_R(u)$.

We now discuss the decision variables required by the formulation. Recall that DCP allows for color classes composed by singleton vertices. Since we assume that G is connected, any color class C_u composed by just a single vertex u can be dominated by a neighbor of u. In order to model the case where a vertex alone defines a color class, the formulation makes use of variables $\mathbf{w} = \{w_u \in \mathbb{B} : u \in V\}$. If $w_u = 1$, $C_u = \{u\}$, u represents itself and no other vertex in V. Otherwise, if $w_u = 0$ holds, vertex u must be part of a color class containing at least two vertices.

Note that if a color class is composed by just two vertices, say two anti-neighbors u and p, then these two vertices must define an edge of $\overline{\delta}_R(u)$ (and $\overline{\delta}_R(p)$) as they must have a common neighbor v that dominates them. To model these cliques, the formulation uses binary variables $\mathbf{y} = \{y_{up} \in \mathbb{B} : \{u, p\} \in \overline{E}_R\}$. Since two anti-neighbors u and p that do not share a common neighbor cannot define a color class, the formulation needs not to assign variables to edges in $\overline{E} \setminus \overline{E}_R$.

The formulation also uses binary decision variables $\lambda = \{\lambda^Q \in \mathbb{B} : Q \in \mathcal{Q}\}$ associated to the maximal cliques in $\mathcal{Q}$. If $\lambda^Q = 1$, all the vertices in Q define a color class. Otherwise, if $\lambda^Q = 0$ holds, at least one vertex in $\mathcal{Q}$ is colored differently from the others.

Our set covering based formulation is defined as

$$\chi_{dom}(G) = \min \left\{ \sum_{Q \in \mathcal{Q}} \lambda^Q + \sum_{u \in V} w_u + \sum_{\{p,q\} \in \overline{E}_R} y_{pq} : (\mathbf{w}, \mathbf{y}, \lambda) \in \mathcal{P}_c \cap (\mathbb{B}^n, \mathbb{B}^{|\overline{E}_R|}, \mathbb{B}^{|\mathcal{Q}|}) \right\}, \quad (10)$$

where $\mathcal{P}_c$ is the defined by constraints (11)-(14).

$$\sum_{Q \in \mathcal{Q}: u \in Q} \lambda^Q + \sum_{\{u,q\} \in \overline{\delta}_R(u)} y_{uq} + w_u \geq 1 \qquad u \in V \quad (11)$$

$$\lambda^Q \in [0,1] \quad Q \in \mathcal{Q} \quad (12)$$

$$w_u \in [0,1] \quad u \in V \quad (13)$$

$$y_{pq} \in [0,1] \quad \{p,q\} \in \overline{E}_R \quad (14)$$

In order to attest its validity, initially note that the definition of the decision variables enforces that all color classes are dominated by some vertex of V. Therefore the dominance property of our coloring is naturally enforced by set covering constraints (11), which also ensure that every vertex belongs to at least one color class. Since decision variables λ are assigned to maximal cliques of $\{\overline{G}[N(u)] : u \in V\}$ only, and not to general cliques of these graphs, the formulation must impose a covering of the vertices of V and not a partitioning of them. Additionally, in order to certify that assigning a same vertex to two distinct color classes does not represent a problem, let $(\hat{\lambda}, \hat{\mathbf{w}}, \hat{\mathbf{y}})$ be an optimal solution to (10) with $\hat{\chi}_{dom}(G)$ color classes. Suppose as well that u belongs to two or more color classes in such a solution. A dominated coloring of G containing no more than $\hat{\chi}_{dom}(G)$ color classes and such that every vertex of G belongs to a single color class, may be obtained directly from $(\hat{\lambda}, \hat{\mathbf{w}}, \hat{\mathbf{y}})$, as follows:

- If $\hat{w}_u = 1$ and either $\hat{y}_{up} = 1$ or $\hat{\lambda}^Q = 1, u \in Q$, applies, one may safely set $\hat{w}_u = 0$, and obtain a solution with $\hat{\chi}_{dom}(G) - 1$ color classes, contradicting the optimality of $(\hat{\lambda}, \hat{\mathbf{w}}, \hat{\mathbf{y}})$.
- If two or more cliques with at least two vertices contain the same u, we can remove u from all but one of them and an alternative optimal solution to (10), with $\hat{\chi}_{dom}(G)$ color classes is thus obtained. In particular, note that the resulting solution remains a proper coloring of G.

Our numerical results show that optimal solutions to (10) typically involve non-disjoint color classes. Accordingly, a fast post-processing procedure based on the second observation above suffices to obtain an alternative optimal solution, with every vertex of G belonging to exactly one color class.

3 ALGORITHMS FOR SOLVING THE FORMULATIONS

We implemented two Branch-and-bound algorithms for solving DCP formulations $\mathcal{P}_r^+$ and $\mathcal{P}_c$, `BBR` and `BBCLK`, respectively. They are implemented in `Python 3.8.5` and rely on the `XPRESS` Mixed Integer Programming (MIP) suite [12], release 39.01.04, for carrying out Branch-and-bound demands. `XPRESS` thus takes care of solving Linear Programming Relaxations (LPRs) and managing the search tree. The solver separates general purpose cutting planes, implements branching and searches the Branch-and-bound tree according to its default policies. Aside from forbidding multi-threading, we changed no other default `XPRESS` parameters.

As formulation $\mathcal{P}_c$ involves exponentially many decision variables, a natural solution approach for solving it would resort to column generation, i.e., generating maximal cliques on-the-fly. Instead of that, we enumerate all cliques of $\mathcal{Q}$ and use them directly in the formulation. Accordingly, differently from most graph coloring algorithms introduced in the past twenty years, ours is a Branch-and-bound algorithm based on the full blown formulation $\mathcal{P}_c$ and not a Branch-and-price one.

The enumeration of maximal cliques of $\{\overline{G}[N(u)] : u \in V\}$ is carried out by our `C` implementation of the algorithm in [11], a variant of the widely known Bron-Kerbosh algorithm [3].

The most time consuming operation for loading formulation $\mathcal{P}_c$ into the MIP solver was not the enumeration of all required maximal cliques, but checking for duplicates among them. Since the same set Q may define a maximal clique for different graphs $\overline{G}[N(u)]$ and $\overline{G}[N(v)]$, we used a hash table to identify duplicate cliques. This hash table was implemented using standard `Python` data structures. As we shall see next, the number of variables appearing in our formulation tends to be relatively small. Additionally, the total CPU times taken to enumerate maximal cliques we require, to check for duplicate ones among them, and to get the algorithm up and running for solving our initial LPRs were not an issue.

4 PRELIMINARY NUMERICAL EXPERIMENTS

This section presents numerical results obtained with formulations $\mathcal{P}_r$, $\mathcal{P}_r^+$ and $\mathcal{P}_c$. We first compare the quality of their LPR bounds and then compare algorithms `BBR` and `BBCLK`, respectively based on $\mathcal{P}_r^+$ and $\mathcal{P}_c$. From now on, assume that $w(\mathcal{P}_r)$, $w(\mathcal{P}_r^+)$ and $w(\mathcal{P}_c)$ denote the LPR bounds associated with formulations $\mathcal{P}_r$, $\mathcal{P}_r^+$ and $\mathcal{P}_c$, respectively.

Our numerical investigation was conducted with two sets of test instances. One of them comprising 29 graphs frequently used to test exact solution algorithms for the Minimum Connected Dominating Set (MCDS) [1]. These are randomly generated instances with $n \in [30, 120]$ vertices and different graph densities, in the range $[5\%, 70\%]$. Instances are identified as *v_n_den*, where n gives the number of vertices for the corresponding connected input graph and *den* is the instance density. Additional details on how they were generated can be found in [1]. The other set comprises 8 benchmark instances for the Maximal Clique Problem, introduced in the 2nd DIMACS Challenge, being available at the web repository `https://iridia.ulb.ac.be/fmascia/maximum_clique`. Among the instances available there, we collected 8, with $n \leq 120$ vertices, namely: myciel4, myciel5, myciel6, hamming6-2, hamming6-4, johnson8-2-4, johnson8-4-4 and MANN_a9.

Table 1 presents some numerical results. Its first four columns provide the instance name, followed by n, m and graph density (den), $\frac{2m}{n(n-1)}$, in percentage values. The three subsequent columns indicate the dual bounds $w(\mathcal{P}_r)$, $w(\mathcal{P}_r^+)$ and $w(\mathcal{P}_c)$. Additionally, the table provides numerical results for the two algorithms under comparison, `BBR` and `BBCLK`. Each algorithm was allowed to run for 1800 seconds, for every instance involved. The results displayed for algorithm BBR are: the best lower (BLB) and upper (BUB) bounds attained during the search, the CPU time (t, in seconds) taken to solve the instance and the number of nodes investigated in the search. For algorithm `BBCLK`, the table provides an additional information, nv, the total number of decision variables needed to formulate the problem. If the best DCP lower and upper bounds found after hitting the imposed time limit do not match, a label "tl" indicates that the instance remained unsolved after 1800 CPU seconds. Numerical experiments were conducted with a 12 core Intel i7-5820K machine, running at 3.30GHz with 32Gbytes of shared RAM memory. Our clique enumeration algorithm was implemented in `C` and compiled with `gcc`, with optimization flag `-O3` turned on.

We first evaluate the impact of strengthening formulation $\mathcal{P}_r$ with valid inequalities (9). To that aim, some additional results are depicted in Figure 1. For each instance in our test set, we plot the LPR ratio $\frac{w(\mathcal{P}_r^+)}{w(\mathcal{P}_r)}$ in the horizontal axis and the graph density, *den*, in the vertical axis. Our results indicate that the inclusion of inequalities (9) impacts positively for sparse instances. For such cases, bounds $w(\mathcal{P}_r)$ frequently more than doubled, without significantly increasing the CPU time demands for their evaluation. In contrast, these inequalities brought no strengthening benefits for instances with input graph densities in the $[50\%, 70\%]$ range.

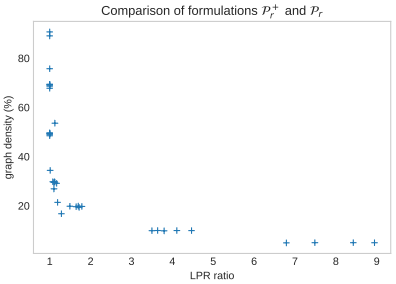

Figure 1: Comparison of formulations $\mathcal{P}_r^+$ and $\mathcal{P}_r$. The horizontal axis gives the ratio $\frac{w(\mathcal{P}_r^+)}{w(\mathcal{P}_r)}$ whereas the vertical axis gives the graph density.

In order to compare formulations $\mathcal{P}_r^+$ and $\mathcal{P}_c$, Figure 2 plots, for each instance, the LPR bound ratio $\frac{w(\mathcal{P}_c)}{w(\mathcal{P}_r^+)}$ (indicated in the horizontal axis) and the ratio between the CPU times taken to compute $w(\mathcal{P}_c)$ and $w(\mathcal{P}_r^+)$ (in the vertical axis). The CPU times we recorded for the computation of $w(\mathcal{P}_c)$ account for the time taken to enumerate cliques, for the checking of duplicate ones, as well as for solving the LPR itself. Measured by the gap $\frac{w(\mathcal{P}_c)-w(\mathcal{P}_r^+)}{w(\mathcal{P}_r^+)}$, formulation $w(\mathcal{P}_c)$ is about 47% stronger, on average, than $w(\mathcal{P}_r^+)$. Compared to $\mathcal{P}_r^+$, formulation $\mathcal{P}_c$ becomes stronger as the graph density increases. Notice that Figure 2 shows that the CPU times taken to compute $w(\mathcal{P}_c)$ are frequently below 50% of the

CPU times needed to compute $w(\mathcal{P}_r^+)$. That happens despite the fact that formulation $\mathcal{P}_c$ involves, at times, more than 150 thousand variables, all of them being explicitly used in the linear programming master program. Our numerical results thus lean in favor of formulation $\mathcal{P}_c$ not only in terms of bound quality but also in terms of the computational effort taken to upload and solve it.

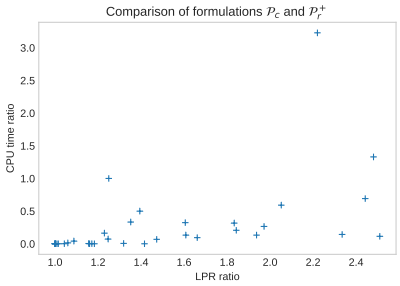

Figure 2: Comparison of formulations $\mathcal{P}_c$ and $\mathcal{P}_r^+$, in terms of LPR relaxation bounds and LPR CPU times. The horizontal axis gives the ratio $\frac{w(\mathcal{P}_c)}{w(\mathcal{P}_r^+)}$ whereas the vertical axis gives the ratio between the CPU time taken to compute bounds $w(\mathcal{P}_c)$ and the time needed to compute $w(\mathcal{P}_r^+)$.

We now discuss numerical results attained by Branch-and-bound algorithms BBCLK and BBR. Out of the 37 instances tested here, BBCLK and BBR respectively solved 30 and 24 instances to proven optimality, within the 1800 seconds time limit. All instances solved by BBR were also solved by BBCLK. While BBCLK takes less than 35 seconds to solve all these 24 instances, BBR takes more than 2744 seconds to accomplish that. BBCLK solved all instances with up to 70 vertices coming from the MCDS literature and failed to solve larger instances with densities in the intermediate range for our test bed. BBCLK solved all 8 maximum clique instances, whereas BBR solved 6 of them.

Considering now the 7 instances both algorithms left unsolved, BBCLK also has the edge for them. Best lower bounds provided by BBCLK are always stronger than BBR's counterparts when the time limit is hit. Similarly, the BUB values attained by BBCLK are strictly smaller than BBR's in 6 out of 7 cases. For just one case, both algorithms attained feasible solutions of the same value. At termination, BBR attains an average duality gap of 38.9% for these 7 instances, while the corresponding figure for BBCLK is just 15.8%.

Another interesting result is that BBR spent the entire CPU time at the root node when solving instance v120_d20. Notice that for this instance, the best lower bound attained by BBR (9.41) is strictly larger than the $w(\mathcal{P}_r^+)$ value (5.94), but smaller than the $w(\mathcal{P}_c)$ counterpart (10.89). BBR root node lower bounds are stronger than the $w(\mathcal{P}_r^+)$ values since BBR (as well as BBCLK) benefits from the general purpose cutting plane algorithm implemented by XPRESS, in the sense that after LPR bounds $w(\mathcal{P}_r^+)$ (and $w(\mathcal{P}_c)$) are computed, XPRESS adds some additional valid inequalities to the formulation at hand. However, for 8 out of the 13 instances not solved by BBR, the best BBR lower bounds, after the addition of these XPRESS cuts throughout the search tree and the enumeration of hundreds of BBR nodes, are weaker than LPR bounds $w(\mathcal{P}_c)$.

5 CONCLUSIONS

We investigated formulations, valid inequalities and Branch-and-bound algorithms for the Dominated Coloring Problem. Two Integer Programming formulations were proposed here. One is based on a model by representatives and the other makes use of exponentially many maximal cliques of the complement graphs induced by the open neighborhood of the vertices of G. Two Branch-and-bound algorithms based on these formulations were also numerically tested here. Our so far limited numerical experience suggests a clear advantage of the formulation based on cliques over the representatives. Linear Programming relaxation bounds for the clique formulation are about 47% stronger than those attained by the formulation by representatives, even after strengthening the latter with a set of valid inequalities introduced here. Better results were also provided by the Branch-and-bound algorithm based on the clique formulation, despite the fact that the algorithm enumerates and explicitly uses all decision variables in the model, without resorting to column generation.

The formulation by representatives could be further strengthened by separating the stronger form (8) of inequalities (3). In doing so, the resulting Branch-and-cut algorithm might become more competitive with the algorithm that relies on maximal cliques. The implementation of a Branch-and-price algorithm that prices cliques instead of explicitly using them in the model should also be an interesting research direction, that might improve the preliminary numerical results provided here.

REFERENCES

[1] Bernard Gendron and Abilio Lucena and Alexandre Salles da Cunha and Luidi Simonetti. 2014. Benders Decomposition, Branch-and-Cut, and Hybrid Algorithms for the Minimum Connected Dominating Set Problem. *INFORMS Journal on Computing* 26(4) (2014), 645–657.
[2] Adrian Bondy and U.S.R. Murty. 2008. *Graph Theory*. Springer.
[3] Coen Bron and Joep Kerbosh. 1973. Finding all cliques of an undirected graph. *Commun. ACM* 16 (1973), 575–577. Issue 9.
[4] Mustapha Chellali and Frédéric Maffray. 2012. Dominator Colorings in Some Classes of Graphs. *Graphs and Combinatorics* 28 (2012), 97–107.
[5] Yen Hung Chen. 2014. The Dominated Coloring Problem and Its Application. In *Computational Science and Its Applications – ICCSA 2014*, Beniamino Murgante et al. (Ed.). Springer International Publishing, 132–145.
[6] Sandi Klavzar and Mostafa Tavakoli. 2021. Dominated and dominator colorings over (edge) corona and hierarchical products. *Appl. Math. Comput.* 390 (2021). https://doi.org/10.1016/j.amc.2020.125647
[7] Minhui Li and Shumin Zhang Chengfu Ye. 2023. Dominated coloring in product graphs. *Journal of Combinatorial Optimization* 46, 4 (2023).
[8] Enrico Malaguti and Paolo Toth. [n.d.]. A survey on vertex coloring problems. *International Transactions in Operational Research* 17, 1 ([n. d.]), 1–34.

Table 1: Linear programming relaxation bounds and numerical results obtained by algorithms BBR and BBCLK.

Instance data				LPR bounds			BBR results				BBCLK results				
Inst	n	m	den	$w(\mathcal{P}\_r)$	$w(\mathcal{P}\_r^+)$	$w(\mathcal{P}\_c)$	BLB	BUB	t	nodes	nv	BLB	BUB	t	nodes
v30_d10	30	44	9.78	3.16	12.00	12.00	12.00	12.00	0.01	1	128	12.00	12.00	0.01	1
v30_d20	30	87	19.33	3.61	6.20	7.20	8.00	8.00	0.08	1	363	8.00	8.00	0.01	1
v30_d30	30	131	29.11	3.50	4.09	5.80	7.00	7.00	1.83	11	589	7.00	7.00	0.15	1
v30_d50	30	218	48.44	5.33	5.33	6.67	7.00	7.00	0.1	1	695	7.00	7.00	0.01	1
v30_d70	30	305	67.78	7.89	7.89	9.20	10.00	10.00	0.01	1	284	10.00	10.00	0.01	1
v50_d10	50	123	9.84	3.00	10.50	11.43	12.00	12.00	1.1	11	545	12.00	12.00	0.01	1
v50_d20	50	245	19.60	3.52	5.79	7.22	8.00	8.00	0.84	1	1443	8.00	8.00	0.02	1
v50_d30	50	368	29.44	4.05	4.46	7.17	8.00	8.00	66.6	1019	2767	8.00	8.00	0.3	1
v50_d50	50	613	49.04	5.02	5.02	9.24	10.00	10.00	6.96	3	3654	10.00	10.00	0.14	1
v50_d5	50	61	4.88	3.17	21.50	21.50	22.00	22.00	0.01	1	160	22.00	22.00	0.01	1
v50_d70	50	858	68.64	9.51	9.51	13.26	14.00	14.00	0.14	1	1094	14.00	14.00	0.02	1
v70_d5	70	121	4.94	2.63	23.50	23.71	24.00	24.00	0.05	1	439	24.00	24.00	0.01	1
v70_d10	70	242	9.88	2.55	11.41	12.09	14.00	14.00	14.35	89	1416	14.00	14.00	0.67	11
v70_d20	70	483	19.71	3.38	6.03	8.88	10.00	10.00	1572.98	19826	4400	10.00	10.00	21.73	1516
v70_d30	70	725	29.59	3.85	4.34	8.56	7.62	12.00	tl	6828	9969	10.00	10.00	21.57	1317
v70_d50	70	1208	49.31	5.32	5.32	10.91	10.55	13.00	tl	6393	13636	12.00	12.00	2.19	1
v70_d70	70	1691	69.02	10.42	10.42	16.73	18.00	18.00	9.16	255	2766	18.00	18.00	0.85	7
v100_d5	100	248	4.96	2.50	21.07	21.98	24.00	24.00	11.12	197	1201	24.00	24.00	0.36	11
v100_d10	100	495	9.90	2.67	10.96	12.67	15.00	15.00	829.04	4006	3717	15.00	15.00	9.45	1793
v100_d20	100	990	19.80	4.00	5.98	9.93	9.35	13.00	tl	261	17324	10.36	12.00	tl	88008
v100_d30	100	1485	29.70	4.00	4.47	10.43	8.05	16.00	tl	95	52674	10.81	12.00	tl	23237
v100_d50	100	2475	49.50	5.78	5.78	14.12	12.42	18.00	tl	543	62598	14.48	16.00	tl	25322
v100_d70	100	3465	69.30	10.91	10.91	21.13	21.29	23.00	tl	24682	7919	22.00	22.00	35.73	6957
v120_d5	120	357	4.96	3.00	22.46	22.80	25.00	25.00	24.43	101	2005	25.00	25.00	0.36	11
v120_d10	120	714	9.92	3.00	10.92	12.79	13.12	16.00	tl	1981	6235	14.22	16.00	tl	279751
v120_d20	120	1428	19.83	3.50	5.94	10.89	9.41	18.00	tl	0	45607	11.09	15.00	tl	13941
v120_d30	120	2142	29.75	4.40	4.73	11.74	8.35	20.00	tl	4	144023	11.83	15.00	tl	1937
v120_d50	120	3570	49.58	6.86	6.86	15.24	12.84	22.00	tl	104	150147	15.35	18.00	tl	2356
v120_d70	120	4998	69.42	9.44	9.44	23.70	23.24	26.00	tl	6090	14401	25.00	25.00	102.98	10632
myciel4	23	71	26.84	2.94	3.24	3.24	5.00	5.00	0.55	25	228	5.00	5.00	0.04	1
myciel5	47	236	21.37	2.98	3.55	3.55	6.00	6.00	203.19	23061	939	6.00	6.00	0.12	11
myciel6	95	755	16.73	2.98	3.83	3.83	4.72	7.00	tl	10582	3900	7.00	7.00	0.32	55
hamming6-2	64	1824	89.06	23.67	23.67	32.00	32.00	32.00	0.01	1	256	32.00	32.00	0.01	1
hamming6-4	64	704	34.38	4.00	4.04	5.33	5.66	7.00	tl	7325	2848	7.00	7.00	0.72	5
johnson8-2-4	28	210	53.57	4.20	4.73	5.60	6.00	6.00	1.16	87	420	6.00	6.00	0.14	1
johnson8-4-4	70	1855	75.71	11.39	11.39	14.00	14.00	14.00	0.36	1	1862	14.00	14.00	0.08	1
MANN_a9	45	918	90.67	15.00	15.00	18.00	18.00	18.00	0.0	1	129	18.00	18.00	0.01	1

[9] Anuj Mehrotra and Michael A Trick. 1996. A Column Generation Approach for Graph Coloring. *INFORMS journal on computing* 8, 4 (1996), 344–354.

[10] Houcine Boumediene Merouane, Mohammed Haddad, Mustapha Chellali, and Hamamache Kheddouci. 2015. Dominated Colorings of Graphs. *Graphs and Combinatorics* 31 (2015), 713–727.

[11] Pablo San Segundo, Jorge Artieda, and Darren Strash. 2018. Efficiently enumerating all maximal cliques with bit-parallelism. *Computers and Operations Research* 92 (2018), 37–46.

[12] FICO XPRESS. 2023. XPRESS mixed integer optimization package, release 8.13, Optimizer 39.01.04.

ACKNOWLEDGMENTS

Alexandre Cunha was supported by CNPq grant 305357/2021-2 and FAPEMIG grants CEX - PPM-00164/17 and RED-00119-21. Abilio Lucena was supported by CNPq grant 310185/2021-1.