A Branch-and-Price Algorithm for the (*k,c*)-Coloring Problem

Enrico Malaguti
DEI, University of Bologna, Viale Risorgimento, 2—40136 Bologna, Italy

Isabel Méndez-Díaz
Department of Computer Science, FCEyN, University of Buenos Aires, Buenos Aires, Argentina

Juan José Miranda-Bront and Paula Zabala
Department of Computer Science, FCEyN, University of Buenos Aires, Buenos Aires, Argentina and Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

In this article, we study the (*k,c*)-coloring problem, a generalization of the vertex coloring problem where we have to assign *k* colors to each vertex of an undirected graph, and two adjacent vertices can share at most *c* colors. We propose a new formulation for the (*k,c*)-coloring problem and develop a Branch-and-Price algorithm. We tested the algorithm on instances having from 20 to 80 vertices and different combinations for *k* and *c*, and compare it with a recent algorithm proposed in the literature. Computational results show that the overall approach is effective and has very good performance on instances where the previous algorithm fails. © 2014 Wiley Periodicals, Inc. NETWORKS, Vol. 000(00), 000–000 2014

Keywords: vertex coloring; multicoloring; branch-and-price; computational experiments; column generation; heuristics; frequency assignment

1. INTRODUCTION

In the vertex coloring problem (VCP), one is required to assign a color to each vertex of an undirected graph in such a way that adjacent vertices receive different colors, and the objective is to minimize the number of colors used. Several problems where a resource (color) has to be shared among conflicting users (vertices connected by an edge in the graph) can be modeled as VCPs. Many applications can be represented as generalizations of the VCP. In the multicoloring problem, for example, users require more than one copy of the resource (see, e.g., [6]). In some applications, the resource cannot be duplicated more than a fixed amount of times, and it is acceptable to have the resource partially shared among conflicting users. This is common in frequency assignment. As an example, Koster and Tieves [10] consider a problem where a group of transmitters has to receive a set of frequencies (colors), in such a way that, when each transmitter chooses at random one frequency among those assigned to its group, the total expected interference among groups (conflicting users) is minimized. The problem is solved in two phases, where the first one is based on a column generation algorithm. Another related problem is studied in Zymolka [22] and Koster and Zymolka [11], and considers the wavelength assignment to lightpaths in optical networks. Wavelengths are modeled as colors in a multicoloring formulation of the problem, where several wavelengths have to be assigned to each lightpath on each link of a given network. A wavelength can be used several times, depending on the link features. Reference [11] describes three formulations for the problem, where the latter has exponential size and is solved through column generation. See Aardal et al. [1] for other examples of frequency assignment problems.

In this article, we consider a generalization of the VCP where each vertex has to receive more than one color, and each pair of conflicting vertices can share a given number of colors.[1]

Formally, the (*k,c*)-coloring problem (KCVCP) is defined as follows. Let $G = (V, E)$ be an undirected graph, with $V = \{1, \ldots, n\}$ the set of vertices, $E \subseteq \{uv : u, v \in V,\ u \neq v\}$ the set of edges, and $R = \{1, \ldots, |R|\}$ the set of available colors. Each vertex $v \in V$ is required to be assigned exactly k

Received August 2013; accepted August 2014
Correspondence to: E. Malaguti; e-mail: enrico.malaguti@unibo.it
Contract grant sponsor: ECI
Contract grant sponsor: MIUR (Italy; to E.M.)
Contract grant sponsors: PICT-2010-0304, PICT-2011-0817, and UBACyT 20020100100666 (to I.M.-D, J.J.M.-B., and P.Z.)
DOI 10.1002/net.21579
Published online in Wiley Online Library (wileyonlinelibrary.com).

[1] A preliminary version of this work appeared in [13].

different colors and each pair of adjacent vertices u, v cannot share more than c colors. The objective is to minimize the total number of colors used. This is called the (k,c)-chromatic number of G, which we denote by $\chi_k^c(G)$. Throughout the article, we assume that $k > c$ and that $|R| \geq \chi_k^c(G)$. The problem is $\mathcal{NP}$-Hard in the general case and reduces to the VCP when $k = 1$ and $c = 0$ (see Garey and Johnson [4] for complexity results on VCP, and Malaguti [12] and Malaguti and Toth [15] for other $\mathcal{NP}$-Hard generalizations of the VCP). The problem remains $\mathcal{NP}$-Hard for the special case when $c = k - 1$ (see Méndez-Díaz and Zabala [20]) while the complexity when $c < k - 1$, $c > 0$, is open.

Coloring problems are very challenging from the computational viewpoint. Considering the classical VCP and restricting our attention to exact algorithms, the most effective methods are based on solving mixed integer linear programming models (MIPs) through advanced techniques.

The so-called natural formulation of the VCP, where variables denote the assignment of colors to vertices, was considered by Méndez-Díaz and Zabala [18, 19]. The formulation was strengthened with additional inequalities, added to eliminate symmetries and to obtain a stronger linear programming (LP) relaxation. In this way, different MIP models for VCP were obtained, which were computationally tested, in terms of their LP relaxation and obtaining exact solutions, both on random instances and on instances from the literature. The most successful alternative model for the VCP is the exponential-size formulation proposed by Mehrotra and Trick [16], where variables are associated with independent sets of vertices (i.e., sets of pairwise non-adjacent vertices). Recently, Malaguti et al. [14], Gualandi and Malucelli [5], and Held et al. [7] achieved very good computational results by solving these formulations through Branch-and-Price (BP) algorithms.

Concerning the KCVCP, Méndez-Díaz and Zabala [21] extended the natural formulation of VCP to the KCVCP, and solved the corresponding model through a Branch-and-Cut (BC) algorithm. The formulation includes symmetry-breaking inequalities and is strengthened by additional inequalities which are generated during the solution process. The authors report computational results on a set of instances with 20 vertices.

As the exponential-size formulation [16] has proved to be very effective in solving not only the classical VCP but also some generalizations (see, e.g., Methrora and Trick [17], Hoshino et al. [8], Furini and Malaguti [3]), in this article, we propose and computationally evaluate an exponential formulation for the KCVCP. The corresponding model is solved through a BP algorithm. To accomplish this task, we develop a primal heuristic and a column generation heuristic as well. Our algorithm is tested on the instances from [21], and compared with the BC algorithm proposed therein.

The rest of the article is organized as follows. In Section 2, we present the MIP formulation proposed in Méndez-Díaz and Zabala [21] and introduce an exponential-size formulation for the KCVCP. In Section 3, we describe the main characteristics of our BP algorithm, including the approach for solving the column generation subproblem, the branching scheme, and a primal heuristic. Computational results are shown in Section 4, and finally, in Section 5, we present some conclusions and future directions.

2. MODELS

We begin by showing the model proposed by Méndez-Díaz and Zabala [21], slightly modified for the particular case of the KCVCP. They consider three different sets of variables. The first one regards binary variables x_{vj}, for $v \in V$ and $j \in R$, taking value 1 if and only if color j is assigned to vertex v. Second, for each arc $uv \in E$ and a color $j \in R$, the binary variable $y_{uvj} = 1$ if and only if color j is assigned to both u and v. Finally, the binary variables w_j, $j \in R$, take value 1 if and only if color j is used by some vertex. The formulation for KCVCP, which we name KCVCP-ASS (colors are assigned to vertices) is the following:

$$\text{(KCVCP-ASS)} \quad z_1 = \min \sum_{j \in R} w_{\mathrm{j}} \tag{1}$$

$$\text{s.t.}: \sum_{j \in R} x_{vj} = k \qquad v \in V \tag{2}$$

$$\sum_{j \in R} y_{uvj} \leq c \qquad uv \in E \tag{3}$$

$$x_{uj} + x_{vj} - y_{uvj} \leq 1 \qquad uv \in E,\ j \in R \tag{4}$$

$$x_{vj} \leq w_j \qquad v \in V,\ j \in R \tag{5}$$

$$x_{vj} \in \{0, 1\} \qquad v \in V,\ j \in R \tag{6}$$

$$y_{uvj} \in \{0, 1\} \qquad uv \in E,\ j \in R \tag{7}$$

$$w_j \in \{0, 1\} \qquad j \in R \tag{8}$$

The objective function (1) minimizes the number of colors used. Constraints (2) establish that exactly k colors are assigned to vertex v and constraints (3) restrict the number of colors that can be shared by two adjacent vertices. Constraints (4) impose $y_{uvj} = 1$ if color j is assigned to both u and v, and constraints (5) set $w_j = 1$ if color j is used by some vertex. Finally, constraints (6–8) establish that all variables must be binary.

Similarly, as with the VCP, this formulation allows symmetric solutions. To eliminate some symmetries, the authors propose to include in the formulation inequalities $w_j \leq w_{j+1}$, for $1 \leq j \leq |R| - 1$.

In this article, we propose a new formulation which does not suffer from symmetry issues. Let $S = 2^V \setminus \{\emptyset\}$ be the power set of V excluding the empty set, and define integer variables x_s, $s \in S$, representing the number of colors assigned to all vertices in s. Actually, in our model any subset of vertices can be feasibly colored by one color when $c > 0$; in the exponential-size formulation for the VCP from [16], instead, only independent sets of vertices are considered. The

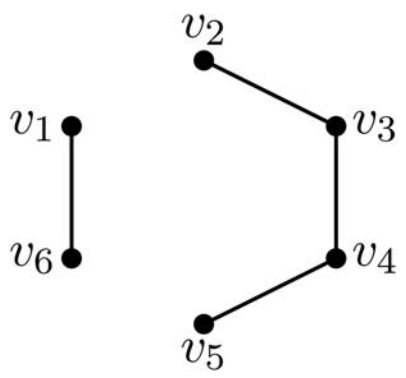

FIG. 1. Graph example.

KCVCP can be formulated as follows:

$$\text{(KCVCP-EXP)} \quad z_2 = \min \sum_{s \in S} x_s \tag{9}$$

$$\text{s.t.} \sum_{s \in S: v \in s} x_s \geq k, \quad v \in V \tag{10}$$

$$\sum_{s \in S: u,v \in s} x_s \leq c, \quad uv \in E \tag{11}$$

$$x_s \in \mathbb{Z}^+, \quad s \in S \tag{12}$$

The objective function (9) minimizes the number of colors used. Constraints (10) establish that a vertex must receive at least k colors. To speed up the column generation algorithm, we consider maximal sets $s \in S$ and modify slightly the definition of KCVCP, where each vertex is required to be assigned at least k colors. Formally, a set $s \in S$ is maximal if for all $v \in V, v \notin s$, there exists a vertex $u \in V, u \in s$ such that $uv \in E$. Consider the following example: let $G = (V, E)$ be an undirected graph, with $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E = \{v_1v_6, v_2v_3, v_3v_4, v_4v_5\}$ (see Fig. 1). Sets $\{v_3, v_5, v_6\}$ and $\{v_1, v_2, v_3, v_4\}$ are maximal, but set $\{v_3, v_4, v_5\}$ is not because either v_1 or v_6 can be included as well without having the set inducing a new edge.

Note that any solution obtained by this formulation can be easily transformed into one having exactly k colors per vertex. In addition, any solution obtained having exactly k colors can be transformed into an equivalent one considering only maximal sets and having the same objective value. The procedure is rather simple: given a nonmaximal sets solution, for each $s \in S$ iteratively add vertices $v \in V, v \notin s$ until s is maximal. In case of obtaining a maximal set s' as the outcome for more than one $s \in S$, set the value of the corresponding variable as the sum of the original ones. Finally, constraints (11) restrict the number of colors assigned to pairs of adjacent vertices in G and constraints (12) require variables x_s, $s \in S$, to be nonnegative integers. Unless otherwise stated, through the rest of the article we assume $S \subseteq 2^V$ including maximal sets only.

Similarly to the VCP, we can establish a dominance relation for the LP relaxation of both formulations. Let z_1^{LP} and z_2^{LP} be the value of the LP relaxation of formulations KCVCP-ASS and KCVCP-EXP, respectively. We also denote by K_n the complete graph on n vertices. The following two propositions provide results on the quality of the lower bounds for each of the considered formulations.

Proposition 1. *Let $G = (V, E)$, with $E \neq \emptyset$. Then $z_2^{LP} \geq 2k - c$.*

Proof. Let x^* be an optimal solution of the LP relaxation of KCVCP-EXP. As E is nonempty, consider edge $uv \in E$ and define $S(u, v) = \{s \in S : u \in s \text{ or } v \in s\}$. Using this definition, we have

$$z_2^{\text{LP}} = \sum_{s \in S} x_s^* \geq \sum_{s \in S(u,v)} x_s^* = \sum_{\substack{s \in S,\\ u \in s}} x_s^* + \sum_{\substack{s \in S,\\ v \in s}} x_s^* - \sum_{\substack{s \in S,\\ u,v \in s}} x_s^*$$
$$\geq k + k - c = 2k - c,$$

where the last inequality is implied by constraints (10) for vertices u and v and constraint (11) for edge uv. ■

Proposition 2. *Let $G = (V, E)$, with $E \neq \emptyset$. Then $z_1^{LP} = k$.*

Proof. Let $(x^*, y^*, \text{and } w^*)$ be an optimal solution of the LP relaxation of KCVCP-ASS. We first show that k is a lower bound on z_1^{LP}, and then provide a feasible (fractional) solution with exactly this objective function. Let $v \in V$ be any vertex; by adding up constraints (5) for $j \in R$ and considering Equation (2) for v, we have

$$k = \sum_{j \in R} x_{vj}^* \leq \sum_{j \in R} w_j^* = z_1^{\text{LP}}.$$

To prove the second condition, we consider the following solution $(\bar{x}, \bar{y}, \text{and } \bar{w})$. Define variables $\bar{x}_{vj} = \frac{k}{|R|}$ for $v \in V, j \in R$, $\bar{w}_j = \frac{k}{|R|}$ for $j \in R$, and $\bar{y}_{uvj} = \frac{c}{|R|}$ for $uv \in E, j \in R$. It is easy to check that this solution satisfies constraints (2), (3), and (5). Regarding constraints (4), for $uv \in E$ and $j \in R$ we have

$$\bar{x}_{uj} + \bar{x}_{vj} - \bar{y}_{uvj} = \frac{k}{|R|} + \frac{k}{|R|} - \frac{c}{|R|} = \frac{2k - c}{|R|}.$$

as $2k - c \leq \chi_k^c(G) \leq |R|$, the ratio is less than or equal to one and thus (4) is satisfied. Finally, $\sum_{j \in R} \bar{w}_j = k$, which completes the proof. ■

Based on these two results, the next corollary holds. The proof follows directly from Propositions 1 and 2 given that $k > c$.

Corollary 1. *Given $G = (V, E)$, with $E \neq \emptyset$, $z_1^{LP} < z_2^{LP}$.*

In the context of the VCP, the lower bound provided by the LP relaxation of the exponential-size formulation is at least the value of the optimal coloring of a maximum clique in the graph. However, in the context of the KCVCP this is not true in the general case and depending on the relation between k and c, the result may hold or not. We next show two cases, one for each situation.

In Méndez-Díaz and Zabala [20] it is shown that, for the special case $c = k-1$, $\chi_k^{k-1}(K_n)$ is the minimum $j \in \mathbb{N}$ such that $\binom{j}{k} \geq n$. Furthermore, the following remark provides a basic relation regarding χ_k^c for arbitrary graphs.

Remark 1. *Let $G = (V,E)$. Then $\chi_k^c(G) \leq \chi_k^{c'}(G)$, for $c' < c$, and $\chi_k^c(G) \leq \chi_{k'}^c(G)$, for $k < k'$.*

When restricting to $k \leq 2c$, we can establish also an upper bound on the value of the LP relaxation of KCVCP-EXP, z_2^{LP}.

Lemma 1. *Let $G = (V,E)$, and consider $k \leq 2c$. Then $z_2^{LP} < 2k$.*

Proof. Consider $\tilde{S} = 2^V \setminus \{\emptyset\}$, and define the solution $\bar{x}_s = \frac{k}{2^{|V|-1}}$ for each $s \in \tilde{S}$. Recall that we can construct a solution having only maximal sets and with the same objective value, as mentioned before in this section. This solution satisfies constraints (10) and (11), and its objective value is smaller than $2k$. ∎

Under the assumption that $k \leq 2c$, the upper bound for the value of z_2^{LP} depends only on the value of k, independent of the size of the graph. In particular, if $c = k-1$, then we can construct a clique of arbitrary size n such that $\chi_k^{k-1}(K_n) > 2k$. Furthermore, by Remark 1 this same clique satisfies $\chi_k^c(K_n) > 2k$, for $c \geq k/2$.

On the contrary, for the case where $k = c(n-1) + r$, with $r \geq 0$, we can show that z_2^{LP} is at least the value of the optimal coloring of a maximum clique in the graph. The following lemma establishes an upper bound on $\chi_k^c(K_n)$ for this particular setting of k and c.

Lemma 2. *Let $k = c(n-1)+r$, with $r \geq 0$. Then, $\chi_k^c(K_n) \leq \frac{n(n-1)c}{2} + rn$*

Proof. First, we consider $\frac{n(n-1)c}{2}$ different colors that can be distributed by assigning c to each edge $uv \in E$ in such a way that for any two of them, the intersection among the colors assigned is empty. Then, we assign to each $v \in V$ the colors of the incident edges. As a result, each vertex has assigned $c(n-1)$ colors, sharing exactly c with all the remaining vertices. To complete the coloring, for each vertex $v \in V$ we assign r new colors, obtaining a feasible (k, c)-coloring with $\frac{n(n-1)c}{2} + rn$ colors in total. ∎

Similarly to Proposition 1, we can derive a lower bound on z_2^{LP} in the particular case where the graph is complete.

Proposition 3. *If $G = K_n$, then $z_2^{LP} \geq kn - \frac{n(n-1)c}{2}$.*

Proof. First, we remark that as the graph is complete, every nonempty set of vertices is maximal, that is, $S = 2^V \setminus \{\emptyset\}$. By adding constraints (10) for $v \in V$ we get the following inequality

$$\sum_{v \in V} \sum_{\substack{s \in S, \\ v \in s}} x_s \geq kn. \tag{13}$$

The left-hand side can be rewritten and bounded as shown below, using the fact that $i - 1 \leq \binom{i}{2}$ for $i \geq 2$.

$$\begin{aligned} \sum_{v \in V} \sum_{s \in S, v \in s} x_s &= \sum_{s \in S} x_s + \sum_{i=2}^{n} (i-1) \sum_{\substack{s \in S, \\ |s|=i}} x_s \\ &\leq \sum_{s \in S} x_s + \sum_{i=2}^{n} \binom{i}{2} \sum_{\substack{s \in S, \\ |s|=i}} x_s \end{aligned} \tag{14}$$

Given a set $s \in S$, as the graph is complete we note that $|\{uv \in E : u, v \in s\}| = \binom{|s|}{2}$. By adding constraints (11) for each $uv \in E$, we get the following inequality

$$\begin{aligned} \frac{c(n-1)n}{2} &\geq \sum_{uv \in E} \sum_{\substack{s \in S, \\ u,v \in s}} x_s = \sum_{s \in S} \sum_{\substack{uv \in E, \\ u,v \in s}} x_s \\ &= \sum_{i=2}^{n} \sum_{\substack{s \in S, \\ |s|=i}} \sum_{\substack{uv \in E, \\ u,v \in s}} x_s = \sum_{i=2}^{n} \binom{i}{2} \sum_{\substack{s \in S, \\ |s|=i}} x_s. \end{aligned}$$

Finally, by inequalities (13) and (14), we obtain the following lower bound on the objective function

$$\sum_{s \in S} x_s \geq kn - \frac{c(n-1)n}{2} \tag{15}$$

which proves the result. ∎

From the results in Lemma 2 and Proposition 3 we can state the following corollary. The proof is omitted as it follows directly by replacing $k = c(n-1) + r$.

Corollary 2. *If $G = K_n$ and $k = c(n-1)+r$, then $\chi_k^c(K_n) = z_2^{LP} = \frac{c(n-1)n}{2} + rn$.*

Given a graph $G = (V,E)$, it is straightforward to check that for every subgraph H of G, $z_2^{\mathrm{LP}}(G) \geq z_2^{\mathrm{LP}}(H)$. When $k = c(n-1) + r$, this property combined with Corollary 2 allows us to state that, similarly to the VCP, the value of the LP relaxation of KCVCP-EXP is at least the optimal (k,c)-coloring of the maximum clique in the graph.

3. BP ALGORITHM

In this section, we present the details of the exact algorithm proposed for the KCVCP, which is built on formulation KCVCP-EXP. This formulation has an exponential number of variables (columns), and therefore, it cannot be formulated explicitly, even for medium size instances. The algorithm starts by executing the greedy heuristic proposed in Méndez-Díaz and Zabala [21]. The solution returned by the heuristic is used as the initial set of columns for formulation KCVCP-EXP, and its value as an initial upper bound for the problem. Then, we solve the LP relaxation of model KCVCP-EXP by means of a column generation procedure. In the following, we denote the LP relaxation of model KCVCP-EXP as the master problem. The main idea behind this technique is to start with a restricted set of columns, obtaining as a result a restricted master problem, and iteratively add columns with negative reduced cost until the master problem is solved to optimality. This procedure is later included within a BP scheme, which has been proven to be quite effective in solving this kind of model. At each node of the enumeration tree, we execute a primal heuristic aiming to improve the upper bound by exploiting the information provided by the solution of the LP relaxation.

3.1. *Initialization*

The initial heuristic proposed in Méndez-Díaz and Zabala [21] follows a greedy criterion to construct a feasible coloring. The procedure begins with an ordered list $L = \{1, \dots, k\}$ and in every step an uncolored vertex is chosen and it is assigned the first k colors from L that are compatible with its neighbors. In case the colors in L are not enough, new colors are added. The colors used in the assignment are moved to the end of L. The procedure runs twice, considering different ordering for the selection of the next vertex.

The output produced by the heuristic is used in two different ways. First, the value of the solution is used as an initial upper bound, which may reduce the size of the enumeration tree within the BP scheme. Second, the solution is used as a part of the initial set of variables for the model KCVCP-EXP. For each color, we construct a set containing all vertices colored with it to generate the corresponding column for model KCVCP-EXP. In addition, we consider as initial columns sets $s = \{v\}$, for $v \in V$, and $s = V$ (i.e., the set containing all vertices).

With this set of variables, we generate an initial restricted master problem and start the column generation procedure.

3.2. *Column Generation*

Let $\bar{S} \subset S$ be this restricted set of columns (e.g., before starting the algorithm, the set of columns described in Section 3.1). The restricted master problem is defined by

$$\min \sum_{s \in \bar{S}} x_s \tag{16}$$

$$\text{s.t.} \sum_{s \in \bar{S}: v \in s} x_s \geq k, \quad v \in V \tag{17}$$

$$\sum_{s \in \bar{S}: u,v \in s} x_s \leq c, \quad uv \in E \tag{18}$$

$$x_s \geq 0 \qquad s \in \bar{S} \tag{19}$$

which is solved to optimality.

Let (π, ρ) be the optimal values of the dual variables associated with constraints (17) and (18), respectively, with $\pi \geq 0$ and $\rho \leq 0$. Aiming to identify sets $s \in S$ for which variable x_s has negative reduced cost, we formulate the following column generation subproblem. Let z_v, $v \in V$, be a binary variable that takes value one if and only if vertex v belongs to s and, for each $uv \in E$, let y_{uv} be a binary variable that takes value 1 if vertices u and v are both in s. The formulation for the subproblem is

$$\max \sum_{v \in V} \pi_v z_v + \sum_{uv \in E} \rho_{uv} y_{uv} \tag{20}$$

$$\text{s.t. } z_u + z_v - y_{uv} \leq 1, \quad uv \in E \tag{21}$$

$$z_v \in \{0,1\}, \qquad v \in V \tag{22}$$

$$y_{uv} \in \{0,1\}, \qquad uv \in E \tag{23}$$

The objective function (20) maximizes the reduced cost of the set characterized by variables z_v. Inequality (21) forces variable y_{uv} to take value 1 when both u and v belong to the set. Finally, constraints (22) and (23) establish integrality conditions.

As the subproblem is a maximization problem and ρ is nonpositive, the only case where y_{uv} takes value 1 but u or v are not in the set s requires $\rho_{uv} = 0$, and thus does not affect the value of the reduced cost. Instead, when a dual variable $\pi_v = 0$, it can be that the optimal solution of model (20–23) corresponds to a set s which is not maximal. This can happen when the vertex v associated with π_v has no neighboring vertices in s, that is, $u \in s, uv \in E$. To forbid this situation, we add the following constraints to the subproblem:

$$z_v + \sum_{u \in V, uv \in E} z_u \geq 1, \quad v \in V. \tag{24}$$

If the objective value of the optimal solution of the subproblem is less than or equal to 1, then the master problem has been solved to optimality. Otherwise, the optimal solution represents a maximal set s^* with negative reduced cost, to be added to the restricted master problem. Then, we set $\bar{S} = \bar{S} \cup \{s^*\}$ and repeat the procedure.

The problem defined by formulation (20–24) stands for a generalization of the maximum weighted stable set problem, where there is a penalization for each pair of conflicting vertices simultaneously included in the set. Therefore, the subproblem is $\mathcal{NP}$-Hard and we have to resort to heuristic procedures to find as quickly as possible columns with negative reduced costs. However, in case the heuristic does not find an entering column, we still have to check if the LP relaxation has been solved to optimality. For this purpose, we consider solving the subproblem by means of an MIP solver.

3.2.1. Heuristic Approach.

Given a set s, the associated profit in the subproblem (20–24) is computed as $\pi(s) = \sum_{v \in s} \pi_v - \sum_{\substack{uv \in E,\\ u,v \in s}} \rho_{uv}$. For a vertex $v \notin s$, we define the residual profit $r(v) = \pi_v - \sum_{\substack{uv \in E, \\ u \in s}} \rho_{uv}$, similarly, for a vertex $v \in s$ we define $r(v) = -\pi_v + \sum_{\substack{uv \in E, \\ u \in s}} \rho_{uv}$. Here, $r(v)$ denotes the change in $\pi(s)$ when a vertex $v \notin s$ (resp., $v \in s$) enters (resp., leaves) set s. The procedure starts by randomly constructing a set s of maximal profit, that is, a subset of V such that no vertex with positive residual profit exists. For a specified number of iterations, the procedure starts from the (current) maximal profit set s and randomly adds a vertex $v \notin s$ to s. Residual profits are updated, and then, the procedure randomly selects a vertex v with $r(v) > 0$ and either (i) adds it to s when $v \notin s$, or (ii) removes it from s when $v \in s$. Residual profits are again updated and the procedure is repeated until set s is of maximal profit, which concludes one iteration of the procedure. Before updating the best incumbent solution, set s is made maximal in a greedy way. To avoid cycling, an anticycling rule is implemented, that is, the vertex initially selected to enter the initial set s of maximal profit, cannot be selected again for a specified number (tenure) of iterations. The procedure is stopped before the specified iterations are completed if a solution of profit larger than a given threshold is produced. The heuristic procedure is described in Algorithm 1.

Algorithm 1 Column Generation Heuristic

```
1.  Greedily compute a maximal profit set $s$, let $\pi(s)$ the
    corresponding profit. Set $s^* = s$
2.  for $i = 0, \ldots, iterations$ do:
3.      Randomly select a (non-tabu) vertex $v \notin s$, set $s = s \cup \{v\}$
4.      Mark $v$ as tabu for the next tenure iterations
5.      Update $\pi(s)$ and residual profits $r(v)$, $v \in V$
6.      while exists $v \in V$ such that $r(v) > 0$
7.          Randomly select $v \in V$ such that $r(v) > 0$
8.          If $v \in s$, set $s = s \setminus \{v\}$
9.          If $v \notin s$, set $s = s \cup \{v\}$
10.         Update $\pi(s)$ and residual profits $r(v)$, $v \in V$
11.     end while
12.     Make $s$ maximal in a greedy way
13.     If $\pi(s) > \pi(s^*)$, set $s^* = s$
14.     If $\pi(s^*) > threshold$, break
15. end for
16. Return set $s^*$ and value $\pi(s^*)$.
```

3.2.2. Exact Approach. When the heuristic procedure cannot find a column with negative reduced cost, we still need to check if the LP relaxation has been solved to optimality. For this purpose, we consider solving the subproblem integer formulation (20–24) using a general purpose MIP solver.

In some cases, solving the subproblem to optimality can be very time consuming. To overcome this issue, we follow a common approach (see, e.g., [5, 7, 14]) and under certain circumstances we stop the optimization of the subproblem before reaching optimality. If during the solution process the solver finds a column with negative reduced cost, the solver is stopped and the column is added to the master problem. Constraints (24) guarantee that the column corresponds to a maximal set. The aim behind this idea is to save the potential time required by the MIP solver to (eventually) prove optimality of the subproblem solution.

In addition, during the solution of the subproblem, we exploit dual information to eventually stop the column generation procedure or to fathom the current node before convergence. By following the same argument as in Farley [2], we have the next result.

Remark 2. *Let Z^* be the optimal solution value to the current restricted master problem and let UB_{SP} be an upper bound on the optimal solution value of the subproblem. Then, a valid lower bound LB on the optimal solution value of the master problem is*

$$LB = \frac{Z^*}{UB_{SP}} \tag{25}$$

Actually, Z^* is the value of a primal-feasible dual-infeasible solution, and UB_{SP} is an upper bound on the left-hand side of any dual constraint. If $\text{UB}_{SP} > 1$, by dividing the value of each dual variable of the dual-infeasible solution by UB_{SP}, we obtain a dual-feasible solution of value $Z^*/\text{UB}_{\text{SP}}$. By weak duality, this value is a lower bound on the optimal solution value of the master problem.

We use the value of LB in two different ways:

- If $\lceil \text{LB} \rceil$ is greater than or equal to the value of the best incumbent solution, then the incumbent solution cannot be improved and the node can be fathomed.
- If $\lceil \text{LB} \rceil = \lceil Z^* \rceil$, then the column generation can be stopped as the LP relaxation cannot be improved at the current node.

Note that the computation of such a bound is computationally costless (the value UB_{SP} is returned by the MIP solver through a callback) and it has the only effect of reducing the LP solution time (i.e., the later behavior of the BP algorithm is not affected).

3.3. *Branching Scheme*

After solving the LP relaxation of model KCVCP-EXP, if the solution is not integer or the node is not fathomed as mentioned in the previous section, we need to continue with the BP algorithm and partition the actual problem into smaller problems by means of a branching scheme. In our case, the BP algorithm branches on the original (primal) variables x_s, $s \in S$, as we describe in this section.

Let x^* be the (noninteger) optimal solution of the LP relaxation, and let x_s, $s \in S$, be a fractional variable. As the variables are defined as nonnegative integers, we partition the solution space and generate two new subproblems defined by the following inequalities

$$x_s \leq \lfloor x_s^* \rfloor \text{ or } x_s \geq \lceil x_s^* \rceil . \tag{26}$$

In each case, the feasible range of values for variable x_s is imposed by adjusting the corresponding bound in the master problem for the descendant nodes. Although the scheme showed to be effective in practice, it is not robust in the sense that adding any of the above inequalities to the formulation modifies the structure of the column generation subproblem solved at each node of the Branch-and-Bound (BB) tree. We then need to account for this situation when solving the subproblem.

When a variable x_s is selected for branching, its bounds in the master problem are adjusted to reflect the new range of feasible values.

Each time we branch on a $x_s \leq \lfloor x_s^* \rfloor$ condition, we have to properly translate the information on the new bound to the subproblem. Otherwise, the upper-bounded variable x_s, that would have negative reduced cost in the master problem, would be regenerated in the subproblem. A way to handle this situation is to remove set s from the set of feasible solutions of the subproblem in the descendant nodes. When solving the subproblem by means of the MIP formulation, this can be achieved by adding to the model (20–24) the following constraint:

$$\sum_{u \in s} z_u \leq |s| - 1 + \sum_{u \notin s} z_u. \tag{27}$$

Constraint (27) imposes that any feasible solution of (20–24) cannot be exactly the same set defined by s.

Each time we branch on a $x_s \geq \lceil x_s^* \rceil$ condition, we do not have to worry about regenerating variable x_s in the subproblem (actually x_s is not upper bounded in the master problem), but instead we look for the constraints (18) which become tight, that is, we look for those $uv \in E$ for which the branching conditions determine $\sum_{s \in S: u,v \in s} x_s = c$. In this case, any column x_s, such that $s \notin \bar{S}, u, v \in s$, cannot enter the solution with a positive value. Again, this information has to be reflected in the subproblem, by explicitly imposing

$$z_u + z_v \leq 1.$$

This can be easily handled considering inequality (21) for edge uv and setting $y_{uv} = 0$.

Regarding the heuristic solution of the subproblem, we also need to account for sets which must not be generated during the process. For this purpose, we keep a pool of forbidden sets which is checked by the heuristic algorithm before updating the incumbent solution. In particular, when we branch on a variable x_s, the set s is stored in a hash table which is later checked by the heuristic in the descendant nodes.

It is important to remark that, as variables x are defined as nonnegative integers, the same variable can be chosen more than once as the branching variable—but considering a different branching value. In this case, the addition of constraint (27) to the subproblem and the inclusion of the corresponding set into the hash table have to be done only the first time the variable is selected for branching, considered in all the descendant nodes of the BB tree and removed when moving to the parent node. Thus, our algorithm includes additional information to keep track of the number of times each variable has been selected as the branching one to maintain the correspondence between the subproblem and the actual state of the node in the BB tree.

The branching variable is selected as the one having the largest fractional part in the (noninteger) solution of the current LP relaxation. We never branch on the initial columns corresponding to sets $s = \{v\}, v \in V$, to guarantee the feasibility of the master problem. Actually, after branching on a fractional valued variable $x_s = x_s^*$, the restricted master problem for the child node may become infeasible even when the LP relaxation is feasible but only by including additional variables not considered so far. The consequence is that dual information for the problem is not available, making difficult the task of solving the LP relaxation of the new node in the BB tree.

The following remark states that it is always possible to branch on a variable other than the initial ones.

Remark 3. *Let x^* be an optimal (noninteger) solution of the LP relaxation of a restricted master problem of KCVCP-EXP. For each $v \in V$, $x^*_{\{v\}}$ cannot be the only fractional variable covering vertex v.*

This is because the right-hand side of constraints (10) is integer and KCVCP-EXP is a minimization problem.

We illustrate the case of a problem that becomes infeasible after branching with the following example. Consider the graph from Section 2, where $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E = \{v_1v_6, v_2v_3, v_3v_4, v_4v_5\}$, $k = 2$, $c = 1$. Suppose $x_{\{v_3\}} \leq 0$ has been imposed at a previous level when branching. Let also $s_1 = \{v_3, v_5, v_6\}$, $s_2 = \{v_1, v_2, v_4, v_5\}$, $s_3 = \{v_2, v_3, v_4, v_6\}$, $s_4 = \{v_1, v_3, v_4, v_6\}$, and $s_5 = \{v_1, v_2, v_3, v_5, v_6\}$, and define $\bar{x}_{s_2} = 1$ and $\bar{x}_{s_i} = 0.5$ for $i = 1, 3, 4$, and 5. Note that all sets s_i correspond to maximal sets, and that $\bar{x}$ is an optimal solution of the LP relaxation of the initial master problem as it is feasible and its objective value is 3 (which is a lower bound by Proposition 1). The constraint matrix for these sets is shown below.

$$\begin{array}{c|ccccc} & x_{s_1} & x_{s_2} & x_{s_3} & x_{s_4} & x_{s_5} \\ v_1 & 0 & 1 & 0 & 1 & 1 \\ v_2 & 0 & 1 & 1 & 0 & 1 \\ v_3 & 1 & 0 & 1 & 1 & 1 \\ v_4 & 0 & 1 & 1 & 1 & 0 \\ v_5 & 1 & 1 & 0 & 0 & 1 \\ v_6 & 1 & 0 & 1 & 1 & 1 \\ v_3v_4 & 0 & 0 & 1 & 1 & 0 \\ v_2v_3 & 0 & 0 & 1 & 0 & 1 \\ v_1v_6 & 0 & 0 & 0 & 1 & 1 \\ v_4v_5 & 0 & 1 & 0 & 0 & 0 \end{array}$$

If variable x_{s_1} is chosen, when branching on $x_{s_1} \leq 0$ the corresponding LP becomes infeasible as the constraint (10) for vertex v_3 cannot be satisfied. However, this new restricted master problem is feasible, for example, setting variables $x_{\{v_i\}}$ to an appropriate value to satisfy the corresponding constraints. This shows the benefit of not branching on the initial variables $x_{\{v\}}, v \in V$.

We mention that the BB tree is traversed in a Depth First Search fashion.

Finally, concerning the possibility of developing a robust branching rule for the KCVCP, we note that the well-known idea proposed by Mehrotra and Trick [16] cannot be easily adapted. In particular, the resulting subproblems are not instances of the KCVCP as the maximum number of colors that two adjacent vertices can share is different from the input value c.

3.4. *Primal Heuristic*

After solving the LP relaxation at a node (or stopping the column generation if no improvement can be achieved), we execute a heuristic procedure which tries to use the information in the (fractional) solution aiming to improve the incumbent solution.

The heuristic considers two phases. In the first stage, the algorithm tries to assign vertices to sets using the variables in the solution of the LP relaxation which have strictly positive values. If, after the first phase, all vertices have been assigned to at least k sets, then the solution is feasible. Otherwise, we execute the second phase to complete the assignment and obtain a feasible solution. If the value of the solution improves the value of the incumbent solution, then the latter is updated and the algorithm continues.

Let x^* be the current solution of the LP relaxation. For the first phase, we consider variables x_s such that $x_s^* > 0$ and sort them according to nonincreasing value. We try to round-up the value of x_s^* and, if the assignment is feasible, assign the vertices in the set s, update the information and continue with the next set, as explained in the following. It is important to note that, after an assignment is done, the resulting problem is not a KCVCP but it has a similar structure. In particular, each vertex $v \in V$ is required to be colored with k_v colors and each pair of adjacent vertices u and v (edge $uv \in E$), has associated a capacity c_{uv} that represents the number of colors that can be shared between u and v. Thus, for a currently considered set $s \subseteq V$, we define a parameter α_s as the maximum number of times set s can be used in a feasible solution without covering some vertex more than needed. Formally,

$$\alpha_s = \min \left\{ \min_{\substack{uv \in E, \\ u,v \in s}} c_{uv}, \max_{v \in s} k_v \right\}$$

Each set s is assigned a number of times which is the minimum between the rounded-up value of the associated variable x_s^* and the value of parameter α_s. The procedure for this phase is described in Algorithm 2.

After the first phase, we still may need to color some vertices to obtain a feasible solution. To complete the coloring and obtain a feasible solution, we consider the following heuristic strategy. We iteratively construct sets $s \subseteq V$ and decide how many times it is used until all vertices $v \in V$ belong to at least k_v sets.

At each iteration, we start by removing from G vertices $v \in V$ such that $k_v \leq 0$. On the resulting induced subgraph, we compute a maximal independent set s and its associated value α_s as previously defined. Then, we iteratively insert vertices into the set in a greedy fashion. For this purpose, we define the following measures for a given set $s \subseteq V$:

Algorithm 2 Primal Heuristic (Phase 1)

```
1. Sort variables $x_s$ such that $x_s^* > 0$ according to non-decreasing value of $x_s^*$. Let $s_0, \ldots, s_p$ be the resulting ordering. Set $k_v = k$, for $v \in V$ and $c_{uv} = c$ for $uv \in E$. Initialize $z_1 = 0$.
2. for $i = 0, \ldots, p$ do:
3.     Let $\alpha = \min\{\lceil x_{s_i}^* \rceil, \alpha_{s_i}\}$. Set $z_1 = z_1 + \alpha$.
4.     Update $k_v = k_v - \alpha$, for $v \in s_i$, and $c_{uv} = c_{uv} - \alpha$, for $uv \in E, u, v \in s_i$.
5. end for
6. Return value $z_1$.
```

1. $f_1^{(s,\beta)}$, which stands for the number of times that, after using β times the set s, there are pairs of vertices $u, v \in V, uv \in E$, with $k_u = 0$ or $k_v = 0$ and $c_{uv} > 0$ in the resulting problem. The insight of this measure is to try to detect situations where the number of colors that two adjacent vertices can share is not fully used.
2. $f_2^{(s,\beta)}$, which counts the number of vertices $v \in V$ for which $k_v = 0$ in the resulting problem, after using β times the set s. This is the number of vertices that are completely colored by set s.
3. $f_3^{(s,\beta)}$, which counts the number of edges $uv \in E$ for which $c_{uv} = 0$ but $k_u, k_v > 0$, after using β times the set s. This is the number of edges whose capacity is consumed when using set s.

Let s be the current set and $\beta = \alpha_s$, and consider a vertex $w \notin s$. We say that w is a candidate vertex to be added to s if for $s' = s \cup \{w\}$ and $\beta' = \alpha_{s'}$, we have $f_1^{(s',\beta')} < f_1^{(s,\beta)}$, or in case of a tie, if $f_2^{(s',\beta')} > f_2^{(s,\beta)}$. In case of a further tie, we finally say that w is a candidate vertex if $f_3^{(s',\beta')} < f_3^{(s,\beta)}$. When there is at least one candidate vertex, we add the best one according to this criterion to s and iterate again. Otherwise, we color the vertices in s and look for another set repeating this procedure. The pseudocode for the second phase is shown in Algorithm 3.

When the second phase finishes, if $z_1 + z_2$ is smaller than the objective value of the incumbent solution, then the best known upper bound has been improved. The incumbent solution is updated and the optimization continues.

4. COMPUTATIONAL RESULTS

We conducted computational experiments to evaluate the behavior of the approach proposed in this article and compare the results with another exact algorithm from the related literature. The experiments were run on a workstation with an Intel(R) Core(TM) i7 CPU (3.40GHz) and 16 GB of RAM. The algorithms are coded in C++ using CPLEX 12.1 Callable Library as the LP and MIP solver.

Algorithm 3 Primal Heuristic (Phase 2)

```
1.  Initialize $z_2 = 0$.
2.  while exists $v \in V$ such that $k_v > 0$ do:
3.      Preprocess the graph by removing $v \in V$ such that
        $k_v \leq 0$ and its incident edges.
4.      Compute a maximal independent set $s$ on $G$ and mark
        vertices $v \in s$.
5.      Set $\beta = \alpha_s$. Compute $f_1^{(s,\beta)}, f_2^{(s,\beta)}$ and $f_3^{(s,\beta)}$.
6.      For each unmarked vertex $w$, set $s' = s \cup \{w\}$, $\beta' = \alpha_{s'}$
        and compute $f_1^{(s',\beta')}, f_2^{(s',\beta')}$ and $f_3^{(s',\beta')}$. Depending on
        the values $f_1^{(s',\beta')}, f_2^{(s',\beta')}$ and $f_3^{(s',\beta')}$, eventually denote $w$
        as candidate.
7.      If at least one candidate vertex exists then:
8.          Let $u$ be the best candidate. Set $s = s \cup \{u\}$ and
            mark vertex $u$. Return to step 5.
9.      end if
10.     Update $k_v = k_v - \beta$, for $v \in s$, and $c_{uv} = c_{uv} - \beta$,
        for $uv \in E$, $u, v \in s$. Set $z_2 = z_2 + \beta$.
11. end while
12. Return value $z_2$.
```

We consider a first set of instances from [21], which consist of randomly generated graphs having 20 vertices and varying the density in terms of the number of edges, considering values of $10, 20, \ldots, 90$ percent of the edges. For each of these values we consider 10 instances. We also consider different combinations for k and c, with $2 \leq k \leq 5$ and $1 \leq c \leq k-1$, giving a total of 900 instances.

We evaluated the following exact algorithms:

1. BC: the BC algorithm from Méndez-Díaz and Zabala [21],
2. BP: the BP algorithm presented in Section 3, and
3. BP-Clique: the BP algorithm, including an initial lower bound obtained by computing a clique and computing the minimum number of colors required to color it. As opposed to the VCP, for the KCVCP the coloring of a clique is not solved unless $c = k - 1$. Thus, we constructed a priori a table for different combinations of k, c and the size of a clique, and used a backtracking algorithm to obtain the value of the optimal clique coloring (a time limit of 1 s for each run is imposed). Then, before starting BP, we compute the size of a maximum clique on the input graph by means of the algorithm proposed by Konc and Janezic [9] and use the precalculated table, in combination with the size of the clique, to obtain a feasible initial lower bound which is incorporated into BP.

We group the instances into three classes of density: low (10% – 30%), medium (40% – 60%), and high (70% – 90%). For medium and high density instances, we report aggregated results in Figures 2 and 3, respectively. This aggregated information is not reported for low density instances, as all algorithms have similar behavior for these problems. Detailed results and computation times are reported in Table 5 in the Appendix.

For each group and each value $k - c$, the first figure reports the optimality gap for:

- the root node of the BC, that is, the LP relaxation of the KCVCP-ASS formulation, strengthened by valid inequalities described in [21] and including an initial lower bound obtained by computing the optimal coloring of a clique,
- the root node of the BP, that is, the LP relaxation of the KCVCP-EXP formulation,
- the root node of the BP-Clique, that is, the LP relaxation of the KCVCP-EXP formulation, including an initial lower bound obtained by computing the optimal coloring of a clique.

Gaps are computed as 100*(BESTUB - ROOT)/ROOT, where BESTUB is the best solution found by any of the algorithms and ROOT is the value of the root node of the corresponding algorithm. The LP relaxation of the KCVCP-ASS formulation with no additional inequalities is very weak (between 39%, for low density graphs and $k - c = 1$, and 269%, for high density graphs and $k - c = 4$, on average) and is not reported.

We observe that the root node of the BP always has a very small optimality gap, zero or almost zero for low and

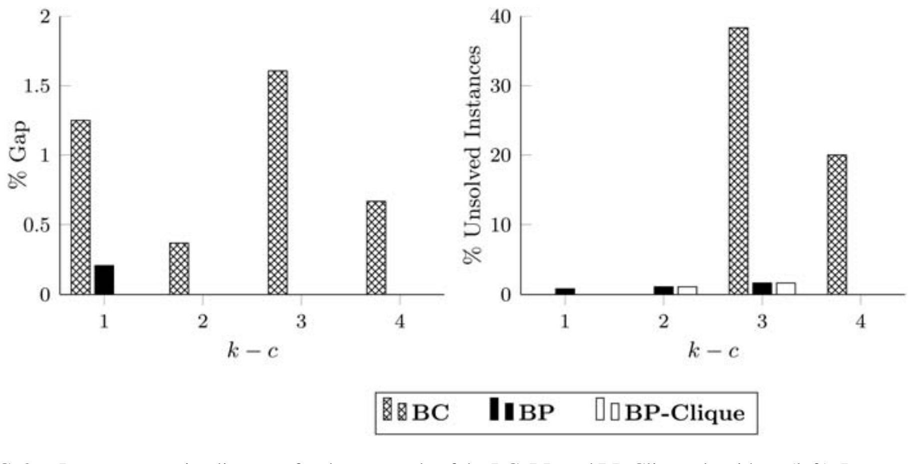

FIG. 2. Percentage optimality gaps for the root node of the BC, BP, and BP-Clique algorithms (left). Percentage of unsolved instances after 3600 s of computation time for the BC, BP, and BP-Clique algorithms (right). Medium density graphs.

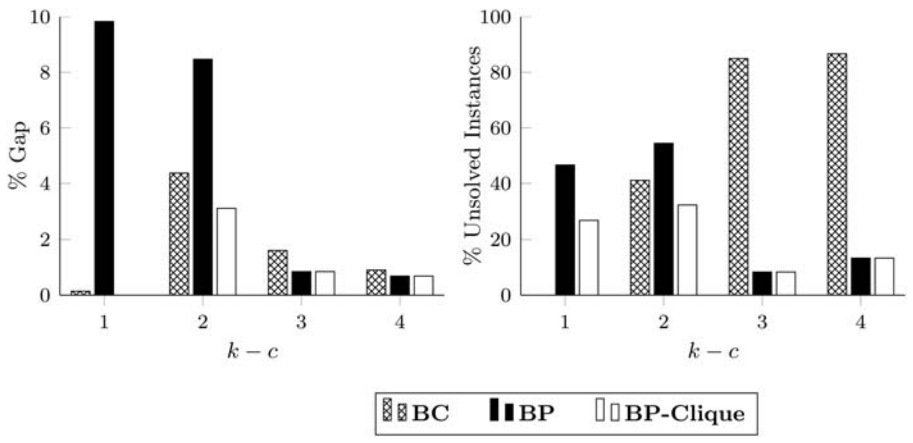

FIG. 3. Percentage optimality gaps for the root node of the BC, BP, and BP-Clique algorithms (left). Percentage of unsolved instances after 3600 s of computation time for the BC, BP, and BP-Clique algorithms (right). High density graphs.

TABLE 1. Computational results for larger instances

	Dens. = 10% *n* = 60		Dens. = 20% *n* = 45		Dens. = 30% *n* = 30	
(*k*, *c*)	Opt.	Time	Opt.	Time	Opt.	Time
(5,1)	7	497.49	6	143.04	10	3.75
(5,2)	9	165.14	5	195.41	9	180.88
(5,3)	9	75.10	10	34.70	10	1.96
(5,4)	10	137.87	10	36.81	10	2.00

medium density instances, showing the benefit of considering an exponential formulation for the problem. For high density instances, the optimality gap is always smaller than 10%, and at most 3.1% for the BP-Clique. The root node of the BC has an optimality gap never exceeding 4.73%, but it is nonzero for small and medium density instances. This is reflected also in the computation times and the number of tree nodes explored, where BP shows on average a reduction compared to BC (see also the Appendix).

For medium and high density instances, we also report, in a second figure, the percentage of unsolved problems after 3600 s of computation time by each algorithm. For low density instances, the BP and BP-Clique solve all instances within the time limit while the BC reached the time limit for one instance with $k - c = 4$. On the whole set, the BC is able to solve 756 instances while BP solves 783. By including the clique computation in the BP algorithm, BP-Clique increases the number of solved instances to 828. In terms of the characteristics of the instances, BC is able to solve all instances where $c = k - 1$, independent of the density of the graph. However, it finds difficulties on instances where c is not close to k. Conversely, BP can solve almost all instances when $k - c > 2$, and produces good results in the remaining cases. In particular, it can solve all instances but three for low and medium density graphs (60% or less), independent of the values of k and c.

For high density graphs, the number of instances solved by BP tends to decrease when c becomes close to k. In general, for both algorithms, instances seem to be harder to solve as the density of the graph increases, with a tendency to solve a smaller number of instances and to increase the computation times.

4.1. Larger Instances

With a second set of instances, we investigate the limit of the BP-Clique in solving the KCVCP for larger graphs; we concentrate on $k = 5$. We generate groups of 10 homogeneous instances with the previously described structure, that is, densities of $10, 20, \ldots, 90$ percent of the edges, $k = 5$ and $1 \le c \le k - 1$. For each density, we are interested in finding the largest n for which we can solve at least five instances for all values of c; thus, we considered increasing values of $n = 25, 30, 35, \ldots,$ *etc*. In Table 1, we report the largest n for which we can solve at least five instances for all values of c, the number of solved instances (Opt.) and the average time in seconds (calculated over solved instances only). For graphs with densities $\ge$ 40% and $n = 30$, we could not solve at least five instances for all values of c.

Table 1 shows that the easiest combination for the BP-Clique is $c = k - 1$ ($k = 5,\ c = 4$), thus, in Table 2 we

TABLE 2. Computational results for larger instances, $k = 5,\ c = 4$

Dens.	*n*	Opt.	Time
10%	80	6	2203.74
20%	60	10	2271.25
30%	50	7	1602.24

TABLE 3. Average computation times for solving the LP relaxation of KCVCP-EXP with and without the column generation heuristic

	Low		Medium		High	
$k - c$	Heur. + MIP	MIP	Heur. + MIP	MIP	Heur. + MIP	MIP
1	0.03	0.34	0.32	4.18	1.90	11.07
2	0.05	1.02	0.41	11.83	2.62	89.03
3	0.03	0.63	0.39	27.24	1.72	229.74
4	0.01	0.20	0.30	13.63	1.27	73.93

report the largest n for which at least five instances are solved in this case, the number of solved instances and the average time in seconds (for solved instances).

These experiments show that the difficulty of the KCVCP, when solved with the BP-Clique, largely depends on the density of the corresponding instances, which cannot be consistently solved for $n \geq 30$ and density $\geq 40\%$ while for density = 10% it is possible to solve instances with $n = 80$.

4.2. *Evaluation of the BP Algorithm Components*

We discuss here some ingredients which are important for the behavior of BP-Clique.

The heuristic procedure developed to solve the column generation subproblem (Section 3.2.1) proved to be very effective, finding columns with negative reduced cost to include in the master problem avoiding unnecessary calls to the MIP solver. The procedure is invoked with the following parameters: iterations = 1000, tenure = 10, threshold = 1.1. Over all the tested instances, columns with negative reduced cost are found by the heuristic procedure in 94% of the cases. To evaluate the reduction in computing time obtained by introducing the column generation heuristic, we solved, for all the instances of the first set, the LP relaxation of KCVCP-EXP (i.e., the master problem) with and without the column generation heuristic. Table 3 reports the average computation times in seconds grouped by values of $k - c$, when only the MIP solver is used (column MIP), and when the column generation heuristic is used as well (column Heuristic + MIP). Note that the same value of threshold = 1.1 is used to stop the MIP solver when a column with negative reduced cost is found. The table shows the importance of the column generation heuristic within the overall BP algorithm. Using the column generation heuristic, the reduction in computation times for solving the LP relaxation of KCVCP-EXP is between one and two orders of magnitude.

TABLE 4. Average computation times for large graphs from Table 1, solved with and without the primal heuristic

(k, c)	Without primal heuristic	With primal heuristic
(5,1)	367.96	190.35
(5,2)	178.11	114.1
(5,3)	55.18	35.95
(5,4)	85.34	58.89

This, combined with the lower bound computed during the solution process of the MIP formulation, allows BP-Clique to solve the LP relaxation effectively and to enumerate, if necessary, a large number of nodes in the BB tree.

Regarding the primal heuristic (described in Section 3.4), we conducted a further experiment. Considering all instances solved within the time limit by BP, we repeated the experiment using the same framework but without including the primal heuristic. For instances of the first set, the total computation times increased 21%. For the second set (corresponding to larger graphs), we considered all the instances from Table 1, that is, $n = 30$ and density of 30%, $n = 45$ and density of 20%, $n = 60$ and density of 10%. For $k = 5$ and $1 \leq c \leq k-1$, we report in Table 4 the average computation times for solving the corresponding instances without including the primal heuristic and with the primal heuristic. The results show that the use of the primal heuristic can produce a good reduction in computation times especially for those instances whose features are at the limit of the performance of the BP-Clique algorithm.

5. CONCLUSIONS

This article presents an exact algorithm for the KCVCP, a generalization of the VCP. We propose an exponential-size formulation for the problem. Based on this formulation, we develop a BP algorithm which proved to be very effective. Compared with the BC approach from the related literature proposed by Méndez-Díaz and Zabala [21], the proposed algorithm is capable of solving 72 more instances out of a total of 900. In particular, the formulation proposed in this article produces in general tight LP relaxations and the overall approach produces very good computational results.

As future research, it would be interesting to deeper investigate the behavior of the BP algorithm for the combinations of k and c for which the algorithm has reduced performance. One of the possible aspects to consider is the node selection strategy, and study whether a different approach would improve the results. Furthermore, based on the results obtained in this research, it would be worth considering a combined approach between our approach and the one from Méndez-Díaz and Zabala [21], and by means of a decomposition of the formulation develop a Branch-Cut and Price algorithm. The inclusion of specific-purpose cuts may help to improve the quality of the LP relaxations on difficult instances.

APPENDIX

In Table 5, we present more detailed results of our experiments, aggregated by groups of 10 homogeneous instances. In the first two columns of the table, we indicate the features of the instances and then, for each algorithm, the average computation times (in seconds) and the number of explored BB tree nodes. For BC, we report the optimality gaps of the LP relaxation (%lpG) of the corresponding KCVCP-ASS formulation, computed as 100*(BESTUB - LB)/LB, where BESTUB is the best solution found by any of the algorithms and LB is the value of the LP relaxation. Similarly, we also show the optimality gaps at the root node (%lpR), after applying problem-specific cuts and including an initial lower bound

TABLE 5. Computation times, number of tree nodes explored, and gap of the linear relaxation—root node for all algorithms (averages)

		BC				BP			BP-Clique		
(*k*, *c*)	Density (%)	Time	Nodes	%lpG	%lpR	Time	Nodes	%lpG	Time	Nodes	%lpG
(2,1)	10	0.00	0.00	50.00	0.00	0.00	1.20	0.00	0.00	0.30	0.00
	20	0.04	2.80	80.00	3.33	0.04	1.30	0.00	0.02	0.60	0.00
	30	0.04	0.00	100.00	0.00	0.06	1.00	0.00	0.00	0.00	0.00
	40	0.20	1.30	100.00	0.00	0.39	2.10	0.00	0.25	1.70	0.00
	50	0.65	9.00	105.00	0.00	(9) 1.51	15.33	2.50	1.33	13.80	0.00
	60	8.04	149.20	100.00	0.00	62.09	331.70	0.00	61.33	331.70	0.00
	70	0.50	0.80	150.00	0.00	* **	7631.8	25.00	1.53	13.60	0.00
	80	3.21	50.70	150.00	0.00	***	6453	25.00	(4) 681.77	1171.50	0.00
	90	1.03	0.50	200.00	0.00	***	5424.9	50.00	(5) 0.00	0.00	0.00
(3,1)	10	0.01	0.00	86.67	0.00	0.01	1.40	0.00	0.00	0.50	0.00
	20	0.09	1.30	100.00	0.00	0.04	2.40	0.00	0.03	1.80	0.00
	30	0.88	25.30	100.00	0.00	0.27	3.50	0.00	0.26	3.50	0.00
	40	42.96	827.10	126.67	3.33	0.33	2.50	0.00	0.24	2.20	0.00
	50	4.17	12.60	133.33	0.00	0.89	14.10	0.00	0.66	13.70	0.00
	60	19.41	84.60	133.33	0.00	1.20	4.10	0.00	1.17	4.10	0.00
	70	(4) 255.61	1539.50	160.00	5.54	(8) 98.89	355.00	1.25	(8) 97.34	355.00	1.25
	80	(8) 121.89	103.00	200.00	2.50	***	4653.1	12.50	(5) 9.84	36.00	2.50
	90	(3) 1.067.67	1946.67	240.00	13.33	***	4189.2	27.50	***	4191.6	13.33
(3,2)	10	0.01	0.00	33.33	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	20	0.01	0.00	33.33	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	30	0.15	2.70	36.67	0.00	0.12	1.60	0.00	0.11	1.50	0.00
	40	0.15	0.60	60.00	0.00	0.15	1.70	0.00	0.06	1.10	0.00
	50	0.17	0.20	66.67	0.00	0.22	1.20	0.00	0.03	0.30	0.00
	60	0.62	1.40	66.67	0.00	0.70	6.90	0.00	0.62	6.70	0.00
	70	6.27	9.70	66.67	0.00	2.42	15.50	0.00	2.43	15.50	0.00
	80	21.84	51.60	66.67	0.00	(5) 8.34	119.00	0.00	(5) 8.46	119.00	0.00
	90	1.99	1.50	96.67	0.00	(1) 14.02	160.00	18.00	(6) 3.26	27.67	0.00
(4,1)	10	0.03	0.00	120.00	0.00	0.00	2.10	0.00	0.00	1.30	0.00
	20	0.15	0.20	140.00	0.00	0.05	6.10	0.00	0.04	5.80	0.00
	30	1.63	18.00	147.50	0.00	0.19	7.80	0.00	0.19	7.70	0.00
	40	21.41	151.10	150.00	0.00	0.40	6.70	0.00	0.41	6.70	0.00
	50	(4) 861.92	8497.25	172.50	4.91	0.83	5.40	0.00	0.84	5.40	0.00
	60	(3) 34.71	152.67	190.00	3.73	(9)66.20	240.11	0.00	(9)66.62	240.11	0.00
	70	(4) 427.57	585.75	200.00	1.82	36.01	99.30	0.91	36.22	99.30	0.91
	80	(1) 455.76	297.00	205.00	1.67	6.83	6.20	0.00	6.92	6.20	0.00
	90	(2) 2.873.65	3743.00	225.00	0.00	6.44	6.60	0.00	6.51	6.60	0.00
(4,2)	10	0.02	0.00	50.00	0.00	0.00	1.00	0.00	0.00	0.10	0.00
	20	0.17	4.40	65.00	0.00	0.05	4.00	0.00	0.04	3.90	0.00
	30	1.13	21.80	75.00	3.33	0.13	4.70	0.00	0.13	4.70	0.00
	40	3.54	31.90	75.00	0.00	0.35	5.20	0.00	0.35	5.20	0.00
	50	19.13	45.60	75.00	0.00	0.78	3.90	0.00	0.78	3.90	0.00
	60	279.05	555.90	75.00	0.00	1.38	3.90	0.00	1.39	3.90	0.00
	70	(4) 189.39	146.50	95.00	7.14	1.97	5.40	0.00	1.99	5.40	0.00
	80	239.12	97.00	100.00	0.00	5.54	5.60	0.00	5.60	5.60	0.00
	90	(4) 1.104.17	6864.50	102.50	1.25	(9) 6.07	5.11	1.25	(9) 6.14	5.11	1.25

TABLE 5. (Continued)

		BC				BP			BP-Clique		
(*k, c*)	Density (%)	Time	Nodes	%lpG	%lpR	Time	Nodes	%lpG	Time	Nodes	%lpG
(4,3)	10	0.01	0.00	25.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	20	0.04	0.00	25.00	0.00	0.01	1.10	0.00	0.01	0.40	0.00
	30	0.33	10.20	25.00	0.00	0.11	2.50	0.00	0.11	2.40	0.00
	40	4.54	174.80	30.00	2.00	0.24	2.20	0.00	0.22	2.10	0.00
	50	15.19	271.20	42.50	8.00	0.42	1.70	0.00	0.37	1.50	0.00
	60	0.35	0.10	50.00	0.00	0.58	3.30	0.00	0.43	2.90	0.00
	70	0.67	0.50	50.00	0.00	1.37	5.60	0.00	1.14	5.40	0.00
	80	1.64	1.40	50.00	0.00	3.13	18.40	0.00	3.16	18.40	0.00
	90	22.54	39.60	50.00	0.00	(3) 6.93	33.00	0.00	(3) 6.98	33.00	0.00
(5,1)	10	0.13	4.10	118.00	1.11	0.01	2.40	0.00	0.01	1.60	0.00
	20	0.13	1.70	144.00	0.00	0.01	3.20	0.00	0.01	2.60	0.00
	30	(9) 0.76	3.44	180.00	0.83	0.07	6.10	0.00	0.06	5.60	0.00
	40	51.58	301.60	184.00	0.00	0.28	11.00	0.00	0.28	11.00	0.00
	50	(8) 848.21	4969.63	202.00	0.67	0.61	10.80	0.00	0.61	10.80	0.00
	60	(6) 1.228.79	5146.33	220.00	1.33	1.26	9.10	0.00	1.26	9.10	0.00
	70	(4) 2.088.30	5089.00	244.00	0.67	67.49	213.50	0.00	67.50	213.50	0.00
	80	***	***	268.00	0.56	(9) 75.28	107.33	0.56	(9) 75.15	107.33	0.56
	90	***	***	296.00	1.50	(7) 420.65	413.29	1.50	(7) 421.41	413.29	1.50
(5,2)	10	0.06	2.00	74.00	0.00	0.01	1.90	0.00	0.00	1.10	0.00
	20	0.55	17.30	80.00	0.00	0.07	6.70	0.00	0.07	6.60	0.00
	30	3.81	26.80	98.00	0.00	0.18	6.60	0.00	0.18	6.60	0.00
	40	29.52	136.50	100.00	0.00	1.17	25.00	0.00	1.19	25.00	0.00
	50	(8) 150.44	622.40	100.00	0.00	2.41	23.60	0.00	2.44	23.60	0.00
	60	(2) 444.42	1430.00	108.00	1.00	6.34	42.60	0.00	6.41	42.60	0.00
	70	(2) 359.08	353.00	120.00	2.00	7.04	14.50	0.00	7.10	14.50	0.00
	80	***	***	120.00	0.00	5.32	6.80	0.00	5.40	6.80	0.00
	90	***	***	148.00	4.17	(5) 326.50	232.20	4.17	(5) 327.87	232.20	4.17
(5,3)	10	0.03	0.30	40.00	0.00	0.01	1.90	0.00	0.01	1.10	0.00
	20	0.18	22.80	46.00	0.00	0.06	2.40	0.00	0.04	2.00	0.00
	30	14.78	1197.40	60.00	2.86	0.14	3.70	0.00	0.14	3.70	0.00
	40	1.49	16.00	60.00	0.00	0.23	5.60	0.00	0.23	5.60	0.00
	50	15.80	91.70	60.00	0.00	0.51	6.80	0.00	0.51	6.80	0.00
	60	109.76	174.70	60.00	0.00	(9) 37.59	438.67	0.00	(9) 38.18	438.67	0.00
	70	(5) 570.41	573.80	70.00	5.00	(3) 50.91	201.67	6.25	(4) 39.07	152.75	5.00
	80	(9) 123.85	76.78	78.00	1.25	(1) 1.34	8.00	11.25	(9) 2.25	7.89	1.25
	90	(6) 628.75	7115.33	86.00	3.33	***	5311.5	16.25	(6) 8.75	15.67	3.33
(5,4)	10	0.01	0.00	20.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
	20	0.05	0.20	20.00	0.00	0.03	2.30	0.00	0.03	1.70	0.00
	30	0.20	3.90	20.00	0.00	0.08	2.40	0.00	0.08	2.20	0.00
	40	0.46	7.50	20.00	0.00	0.18	4.10	0.00	0.18	4.10	0.00
	50	39.21	283.60	22.00	1.67	0.38	3.20	0.00	0.38	3.20	0.00
	60	68.07	345.50	28.00	3.33	0.73	2.10	0.00	0.67	2.00	0.00
	70	46.52	182.40	40.00	1.67	1.01	4.40	0.00	0.75	4.10	0.00
	80	1.32	0.70	40.00	0.00	(9) 1.91	7.67	0.00	(9) 1.81	7.56	0.00
	90	5.39	1.50	40.00	0.00	(6) 3.74	10.33	0.00	(6) 3.51	10.17	0.00

provided by a clique. For BP and BP-Clique the optimality gaps at the root node (%lpR) of the corresponding algorithm are reported. For each instance, we impose a time limit of 3600 s for the overall time required by each algorithm. The average computation times and number of nodes are calculated only over instances for which the corresponding algorithm finishes before reaching the time limit. A cell filled with *** means that the algorithm was not able to solve any of the instances (in this case only, the average number of nodes is for unsolved intances), and a number between parenthesis stands for the number of instances effectively solved within the time limit.

ACKNOWLEDGMENTS

This work was started when Enrico Malaguti was visiting the University of Buenos Aires on the occasion of the ECI winter school. The authors are grateful to two anonymous referees for their careful reading and useful comments, that helped very much in improving the article.

REFERENCES

[1] K. Aardal, S. van Hoesel, A. Koster, C. Mannino, and A. Sassano, Models and solution techniques for the frequency assignment problem, 4OR 1 (2003), 261–317.

[2] A.A. Farley, A note on bounding a class of linear programming problems, including cutting stock problems, Oper Res 38 (1990), 922–923.

[3] F. Furini and E. Malaguti, Exact weighted vertex coloring via branch-and-price, Discrete Optim 9 (2012), 130–136.

[4] M. Garey and D. Johnson, Computers and intractability: A guide to the theory of NP-completeness, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Freeman, New York, 1979.

[5] S. Gualandi and F. Malucelli, Exact solution of graph coloring problems via constraint programming and column generation, INFORMS J Comput 24 (2012), 81–100.

[6] M. Halldórsson and G. Kortsarz, Multicoloring: Problems and techniques, mathematical foundations of computer science 2004, Lect Notes Comput Sci 3153 (2004), 25–41.

[7] S. Held, W. Cook, and E. Sewell, Maximum-weight stable sets and safe lower bounds for graph coloring, Math Program Comput 4 (2012), 363–381.

[8] E. Hoshino, Y. Frota, and C.C. de Souza, A branch-and-price approach for the partition coloring problem, Oper Res Lett 39 (2011), 132–137.

[9] J. Konc and D. Janezic, An improved branch and bound algorithm for the maximum clique problem, MATCH Commun Math Comput Chem 58 (2007), 569–590.

[10] A. Koster and M. Tieves, Column generation for frequency assignment in slow frequency hopping networks, EURASIP J Wireless Commun Networking 253 (2012), 1–14.

[11] A. Koster and A. Zymolka, Tight LP-based lower bounds for wavelength conversion in optical networks, Stat Neerl 61 (2007), 115–136.

[12] E. Malaguti, The vertex coloring problem and its generalizations, 4OR Q J Oper Res 7 (2009), 101–104.

[13] E. Malaguti, I. Méndez-Díaz, J. Miranda-Bront, and P. Zabala, $(k,\ c)$-Coloring via column generation, Electron Notes Discrete Math 41 (2013), 447–454.

[14] E. Malaguti, M. Monaci, and P. Toth, An exact approach for the vertex coloring problem, Discrete Optim 8 (2011), 174–190.

[15] E. Malaguti and P. Toth, A survey on vertex coloring problems, Int Trans Oper Res 17 (2010), 1–34.

[16] A. Mehrotra and M. Trick, A column generation approach for graph coloring, INFORMS J Comput 8 (1996), 344–354.

[17] A. Mehrotra and M. Trick, “A branch-and-price approach for graph multicoloring,” Extending the horizons: Advances in computing, optimization, and decision technologies, Operations Research/Computer Science Interfaces Series, E. Baker, A. Joseph, A. Mehrotra, and M. Trick (Editors), Springer, 2007, Vol. 37, pp. 15–29.

[18] I. Méndez-Díaz and P. Zabala, A branch-and-cut algorithm for graph coloring, Discrete Appl Math 154 (2006), 826–847.

[19] I. Méndez-Díaz and P. Zabala, A cuttting plane algorithm for graph coloring, Discrete Appl Math 156 (2008), 159–179.

[20] I. Méndez-Díaz and P. Zabala, The $(k, k-1)$ coloring problem, Proc Int Symp Comb Optim, Warwick, UK, 2008.

[21] I. Méndez-Díaz and P. Zabala, Solving a multicoloring problem with overlaps using integer programming, Discrete Appl Math 158 (2010), 349–354.

[22] A. Zymolka, Design of survivable optical networks by mathematical optimization, Ph.D. Thesis, TU Berlin, 2006.