RAIRO-Oper. Res. 55 (2021) S1949–S1967
https://doi.org/10.1051/ro/2020065

RAIRO Operations Research
www.rairo-ro.org

INTEGER AND CONSTRAINT PROGRAMMING APPROACHES FOR PROVIDING OPTIMALITY TO THE BANDWIDTH MULTICOLORING PROBLEM

Bruno Dias[1,*], Rosiane de Freitas[1], Nelson Maculan[2] and Philippe Michelon[3]

Abstract. In this paper, constraint and integer programming techniques are applied to solving bandwidth coloring problems, in the sense of proving optimality or finding better feasible solutions for benchmark instances from the literature. The Bandwidth Coloring Problem (BCP) is a generalization of the classic vertex coloring problem (VCP), where, given a graph, its vertices must be colored such that not only adjacent ones do not share the same color, but also their colors must be separated by a minimum given value. BCP is further generalized to the Bandwidth Multicoloring Problem (BMCP), where each vertex can receive more than one different color, also subject to separation constraints. BMCP is used to model the Minimum Span Channel Assignment Problem (MS-CAP), which arises in the planning of telecommunication networks. Research on algorithmic strategies to solve these problems focus mainly on heuristic approaches and the performance of such methods is tested on artificial and real scenarios benchmarks, such as GEOM, Philadelphia and Helsinki sets. We achieve optimal solutions or provide better upper bounds for these well-known instances, We also compare the effects of multicoloring demands on the performance of each exact solution approach, based on empirical analysis.

Mathematics Subject Classification. 90C10, 68R10, 90C27.

Received December 31, 2018. Accepted June 17, 2020.

1. Introduction

Let $G = (V, E)$ be an undirected graph. A *k-coloring* of G is an assignment of colors $\{1, 2, \ldots, k\}$ to the vertices of G so no two adjacent vertices share the same color. The *chromatic number* χ_G of a graph is the minimum value of k for which G is *k-colorable*. The classic vertex coloring problem (VCP), which consists in finding the chromatic number of a graph, is a well-known combinatorial optimization problem which belongs to NP-hard complexity class [10, 12, 14].

One of the main applications of such problems consists of assigning channels to transmitters in a mobile wireless network. Each transmitter is responsible for the calls made in the area it covers and the communication among devices is made through a channel consisting of a discrete slice of the electromagnetic spectrum. However,

Keywords. Bandwidth coloring, channel assignment, integer and constraint programming, graph theory.

[1] Instituto de Computação (IComp/UFAM), Universidade Federal do Amazonas, Manaus, Brazil.
[2] PESC/COPPE, Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro – RJ, Brazil.
[3] Centre d'Enseignement et de Recherche en Informatique, Université d'Avignon et des Pays de Vaucluse, Avignon, France.
*Corresponding author: bruno.dias@icomp.ufam.edu.br

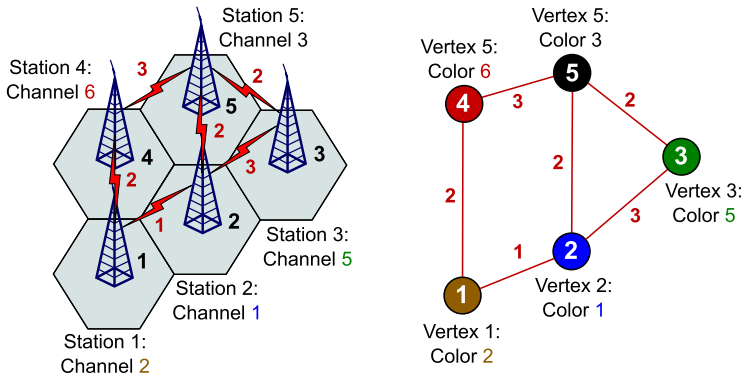


FIGURE 1. Example of channel assignment and its modeling as an instance of Bandwidth Coloring Problem.

the channels cannot be assigned to calls in an arbitrary way, since there is the problem of interference among devices located near each other using approximate channels. Given this scenario, channels must be assigned to calls so interference is avoided and the spectrum usage is minimized [17, 27, 28].

Thus, the channel assignment scenario can be modeled as a graph coloring problem by considering each transmitter as a vertex in a simple undirected graph and the channels to be assigned as the colors the vertices will receive. Some more general graph coloring problems were proposed in the literature in order to take the separation among channels into account, such as the T-coloring problem, also known as the Generalized Coloring Problem (GCP) [8, 13], which was one of the first combinatorial optimization approaches to channel assignment, where each edge is assigned a given forbidden set such that the absolute difference between colors given to each incident edge is in the set. The scenario where stations may need more than one channel, that is, vertices in the corresponding graph may have a demand of more than one color, is modeled as the set T-coloring problem [8], in which there is also a forbidden set for each vertex so that the absolute difference between colors assigned to the same vertex must not be in its forbidden set.

A special case of T-coloring consists of forbidden sets containing only consecutive integer numbers starting from zero (that is, sets of form $\{0, 1, 2, \ldots, d\}$, or, equivalently, intervals $[0, \ldots, d] \subset \mathbb{Z}$), which means the absolute difference between colors assigned to each vertex must be greater or equal a certain value. This case is known in the literature as the Bandwidth Coloring Problem (BCP) [18, 20, 21, 24], since this requirement occurs with respect to frequency bands in a wireless network. An example of channel assignment instance and its corresponding model as a BCP instance is shown in Figure 1. Note that the edges are only assigned the upper interval bound $d_{i,j}$.

The separation constraints in the BCP can be seen as a type of distance constraint, so we can see the channel assignment as a type of distance geometry (DG) problem, since we have to place the channels in the transmitters respecting some distances imposed in the edges, where a coloring $x : V \to \mathbb{N}$, such that $\forall\{i, j\} \in E,\ |x(i) - x(j)| \geq d_{i,j}$, must be found [6, 7]. This theoretical model can be used to derive integer and constraint programming models based on characteristics from the problem as well as previous works with similar problems.

The main contribution of this paper consists of the use of integer and constraint programming models to provide exact solutions to BCP, applying them to existing instances, including ones based on real channel assignment scenarios. There are many algorithms for BCP in the literature, including some based on classic metaheuristics, including simulated annealing [3], local search [9], evolutionary algorithms [20] and tabu search [8, 18]. However, there is no optimality guarantee in these methods. Using integer and constraint programming approaches, we were able to prove the optimality of some solutions found by heuristic methods, such as

the multistart iterated tabu search proposed in [18], and obtain better upper bounds for some problems, including optimal solutions for open instances. In this process, we also found some inconsistencies in the literature with respect to the quality of some approximate algorithms, where the heuristic presented solutions better than the optimal ones found by an exact method.

The remainder of this paper is organized as follows. Section 2 formally defines the Bandwidth Coloring Problem and discusses its characteristics. Section 3 gives a mathematical formulation in constraint programming based on theoretical distance geometry models, and also gives an integer programming formulation for comparison. Section 4 shows results of some experiments done with implementations of the mathematical models. Finally, in Section 5, final remarks are made and next steps of ongoing research are stated.

2. Bandwidth Coloring definitions and models

The Bandwidth Coloring Problem (BCP) can be stated as follows. Given an undirected graph $G = (V, E)$ where, for each edge $(i, j) \in E$, there is a positive integer $d_{i,j}$, each vertex i must receive a color $x(i)$ and, for each edge $(i, j) \in E$, the condition $|x(i) - x(j)| \geq d_{i,j}$ must hold.

This problem is a special case of T-coloring [13], since we can build a T-coloring instance from any BCP instance by setting the forbidden set of an edge $(i, j) \in E$ to $T_{i,j} = \{0, 1, \ldots, d_{i,j}\}$, The constraint $|x(i) - x(j)| \notin T_{i,j}$ is, then, the same as the one from BCP, that is, the former corresponds to a T-coloring instance with forbidden sets consisting of consecutive values.

The constraints imposed on BCP are a kind of distance constraint, so it can be modeled using concepts from distance geometry (DG) [4–6]. The fundamental Distance Geometry Problem (DGP) has, as input, a graph $G = (V, E)$ such that for each edge $(i, j) \in E$, a distance $d_{i,j} \in \mathbb{R}_+$ is attributed to it and an embedding $x : V \to \mathbb{R}^n$ must be found such that $||x(i) - x(j)|| = d_{i,j}$ for all $(i, j) \in E$. When $n = 1$, we have 1-DGP, which is equivalent to 1-Embeddability [25]. If the condition $\forall (i, j) \in E,\ d_{i,j} \in \mathbb{N}$ also holds, then we have the Discretizable Distance Geometry Problem (DDGP) in one dimension, or 1-DDGP. Based on this model, we can formalize the following coloring problem.

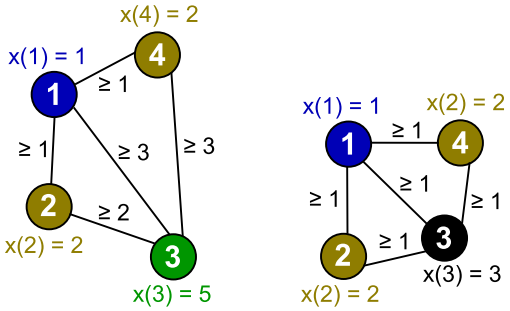
Definition 2.1 (Minimum Equal Coloring Distance Geometry Problem (MinEQ-CDGP))**.** Given a simple weighted undirected graph $G = (V, E)$, where, for each $(i, j) \in E$, there is a weight $d_{i,j} \in \mathbb{N}$, find an embedding $x : V \to \mathbb{N}$ such that $|x(i) - x(j)| = d_{i,j}$ for each $(i, j) \in E$ whose span S, defined as $S = \max_{i \in V} x(i)$, that is, the maximum used color, is the minimum possible.

By substituting the equality constraint in MinEQ-CDGP into an inequality (greater than or equal) and considering only one dimension for the embedding and only its positive integer points, a DG model for BCP can be derived, as defined below.

Definition 2.2 (Minimum Greater than or Equal Coloring Distance Geometry Problem – MinGEQ-CDGP [4, 5])**.** Given a simple weighted undirected graph $G = (V, E)$, where, for each $(i, j) \in E$, there is a weight $d_{i,j} \in \mathbb{N}$, find an embedding $x : V \to \mathbb{N}$ such that $|x(i) - x(j)| \geq d_{i,j}$ for each $(i, j) \in E$ whose span $(\max_{i \in V} x(i))$ is the minimum possible.

The MinGEQ-CDGP is equivalent to BCP, since, for each vertex $i \in V$, the point $x(i)$ assigned to it in MinGEQ-CDGP corresponds to its color in BCP. A special case when, for all $(i, j) \in E$, we have $d_{i,j} = \beta$, where $\beta \in \mathbb{N}$. When $\beta = 1$, we have a VCP instance, where colors between adjacent vertices must only be different among each other. The input graph can be stated only by its vertices, edges and the β value. The corresponding DG model is stated below and exemplified in Figure 2.

Definition 2.3 (Minimum Greater than or Equal Coloring Distance Geometry Problem with Constant Distances – MinGEQ-CDGP-Unif)**.** [4,5]: Given a simple weighted undirected graph $G = (V, E)$, and a nonnegative integer β, find an embedding $x : V \to \mathbb{N}$ such that $|x(i) - x(j)| \geq \beta$ for each $(i, j) \in E$ whose span $(\max_{i \in V} x(i))$ is the minimum possible.



(a) MinGEQ-CDGP. (b) MinGEQ-CDGP-Unif.

FIGURE 2. Examples of instances for MinGEQ-CDGP (BCP) and MinGEQ-CDGP-Unif problems. Note that the MinGEQ-CDGP-Unif instance used is also a VCP one.

A variation of BCP is the Bandwidth Multicoloring Problem (BMCP), which is also a special case of set T-coloring, where the vertex i has an associated color demand q_i and a weight $d_{i,i}$, such that it receives q_i colors (instead of just one). Let $x(i,k)$ be the k-th color assigned to vertex i (with $1 \leq k \leq q_i$). Then, for each pair of colors $x(i,k)$ and $x(i,m)$ associated to i, the constraint $|x(i,k) - x(i,m)| \geq d_{i,i}$ must be respected in BMCP. An equivalent problem to BMCP is the Minimum Span Channel Assignment (MS-CAP) [1,17], also known as Minimum Span Frequency Assignment (MS-FAP) where channels correspond to colors and devices to vertices. However, the input consists of positions for these devices and reuse distances, where, based on the distance between two devices, the separation between colors is calculated. If this input is converted into a graph where edges are weighted according to this separation, then it becomes a BMCP instance.

For BMCP, we can extend MinGEQ-CDGP as shown below.

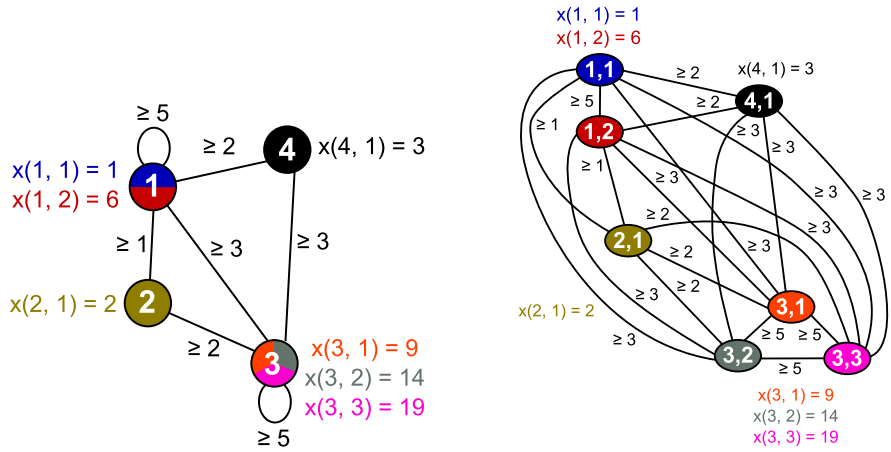
Definition 2.4 (Minimum Greater than or Equal Multicoloring Distance Geometry Problem – MinGEQ-Multi-CDGP)**.** Given a simple weighted undirected graph $G = (V, E)$, where, for each $(i,j) \in E$, there is a weight $d_{i,j} \in \mathbb{N}$, and, for each vertex $i \in V$, there are weights $q_i,\ d_{i,i} \in \mathbb{N}$ find an embedding $x : V \to 2^{\mathbb{N}}$ such that $|x(i)| = q_i$; letting $x(i,k)$ be the k-th color assigned to i then for each $(i,j) \in E$, $1 \leq k \leq q_i$ and $1 \leq m \leq q_j$, $|x(i,k) - x(j,m)| \geq d_{i,j}$; and, for each $i \in V$, $1 \leq k,l \leq q_i$, $|x(i,k) - x(i,l)| \geq d_{i,i}$; whose span $(\max_{i \in V,\ 1 \leq k \leq q_i} x(i))$ is the minimum possible.

As is the case with all multicoloring problems, there is an equivalence between MinGEQ-CDGP and MinGEQ-Multi-CDGP, that is, an instance of MinGEQ-Multi-CDGP can be converted into MinGEQ-CDGP by replicating each vertex i into a clique of q_i subvertices. Each edge in the clique has a distance imposed on it equal to $d_{i,i}$ from the original MinGEQ-Multi-CDGP instance, and each subvertex is adjacent to all other vertices that the original vertex also was adjacent to. Figure 3 shows an example of this conversion. By employing this transformation, any algorithm for MinGEQ-CDGP can be used for MinGEQ-Multi-CDGP. However, if the algorithm does not explore specific characteristics of multicoloring, its runtime will be much higher [18,22].

In Figure 4, we show the hierarchy of coloring problems shown in this section and which ones are the focus of this paper. We refer the reader to [5] for more information about other distance coloring models.

3. Constraint and integer programming approaches

The distance geometry approach for graph coloring problems with distances is directly mapped to a constraint programming (CP) approach, since it addresses the graph coloring problem with distances as an embedding of the graph in one dimension, that is, a labeling of the vertices with natural numbers indicating its projection on the line, such that the distances of the segments defined by the edge weights of the graph are met, and such



(a) Original MinGEQ-Multi-CDGP instance.

(b) Transformation to MinGEQ-CDGP.

FIGURE 3. Example of MinGEQ-Multi-CDGP instance and its transformation into a MinGEQ-CDGP instance.

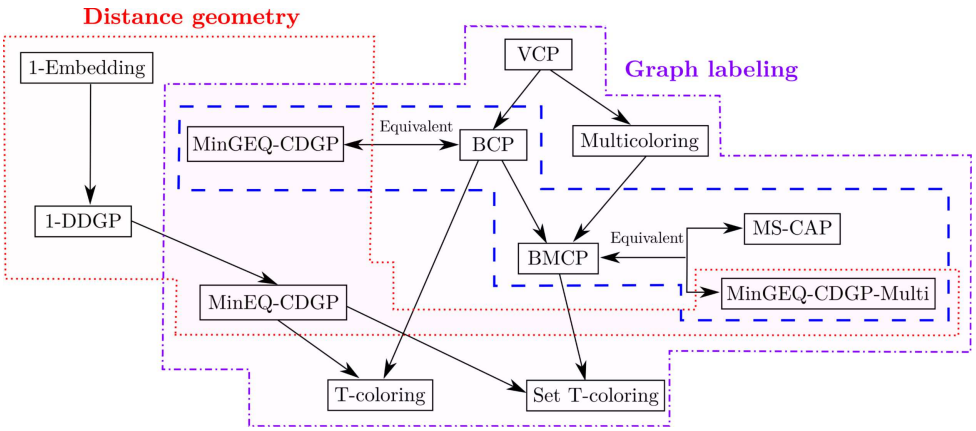


FIGURE 4. Hierarchy of graph coloring problems with distance constraints. The problems explored in this paper are highlighted by the dashed polygon.

that the total range is minimized (the span of colors is minimized). In this way, a mathematical technique that handles these segments or distance constraints is very useful, which is the case for the constraint programming model. We also present a traditional integer programming formulation, to compare both approaches.

Thus, the first formulation presented is based on constraint programming, which will be denoted by MinGEQ-CDGP-CP and is proposed in this work by directly using the problem definition.

Let x_i be an integer variable consisting of the color assigned to vertex i. Then MinGEQ-CDGP-CP is defined as:

$$\text{Minimize} \quad \max_{i \in V} x(i) \tag{3.1}$$

$$\text{Subject to} \quad |x(i) - x(j)| \geq d_{i,j} \qquad (\forall (i,j) \in E) \tag{3.2}$$

$$x(i) \in \mathbb{Z}^* \qquad (\forall i \in V) \tag{3.3}$$

The objective (3.1) is to minimize the maximum used color among all vertices (the coloring span). Constraint set (3.2) involves weighted edges with inequality constraints. For each variable x_i, the initial domain (before constraint propagation) $D(x_i)$ consists of all integers between 1 and a given upper bound U, that is, $D(x_i) = [1, \ldots, U]$. Note that all initial domains are the same. This model has $O(|V|)$ variables (one for each vertex) and $O(|E|)$ constraints (one for each edge), since it captures the essential definition of the problem.

A special CP model can be stated for the case when all distances are the same (MinGEQ-CDGP-Unif) and the input graph is complete, using a specific global constraint. We propose such model, denoted by MinGEQ-CDGP-Unif-CP, which is defined as:

$$\text{Minimize} \quad \max_{i \in V} x(i) \tag{3.4}$$

$$\text{Subject to} \quad \text{allMinDistance}(\{x(i) : i \in V\},\ \beta) \tag{3.5}$$

$$x(i) \in \mathbb{Z}^* \qquad (\forall i \in V) \tag{3.6}$$

The *allMinDistance* global constraint used in (3.5) takes as its arguments a set of variables and a number w, which is the minimum distance to be respected, and ensures that, for all pairs of variables y and z in the set, the condition $|y - z| \geq w$ is valid, which is the case for each vertex and its neighbors. This special case has $O(|V|)$ variables and only one global constraint. This formulation has fewer constraints and is able to use specialized propagators. We note that *allMinDistance* is a generalization of the classical *allDifferent* global constraint, such that when the constant w (that is, the distance to be respected) is set to 1, the result of both constraints would be the same. However, there is no guarantee that the algorithms for separating *allDifferent* will be used in this scenario.

For MinGEQ-Multi-CDGP, a CP formulation can be constructed by using characteristics from both previously shown models. As discussed in Section 2, a multicoloring problem can be converted into a coloring with single demands by transforming a vertex i into a clique of q_i vertices, each adjacent to all other vertices that were adjacent to i. By using this, we have that, essentially, each vertex consists of a small MinGEQ-CDGP-Unif subinstance, where the larger graph (that is, considering the constraints imposed on the original edges between different vertices), if its multicoloring demands are ignored, is a MinGEQ-CDGP instance. Based on this combination, we propose the following CP formulation, denoted by MinGEQ-Multi-CDGP-CP:

$$\text{Minimize} \quad \max_{\substack{i \in V \\ 1 \leq k \leq q_i}} x(i,k) \tag{3.7}$$

$$\text{Subject to} \quad |x(i,k) - x(j,m)| \geq d_{i,j} \qquad (\forall (i,j) \in E,\ 1 \leq k \leq q_i,\ 1 \leq m \leq q_j) \tag{3.8}$$

$$\text{allMinDistance}(\{x(i,k) : 1 \leq k \leq q_i\},\ d_{i,i}) \qquad (\forall i \in V) \tag{3.9}$$

$$x(i,k) \in \mathbb{Z}^* \qquad (\forall i \in V,\ 1 \leq k \leq q_i) \tag{3.10}$$

In MinGEQ-Multi-CDGP-CP, constraints (3.8) ensure that colors assigned to different vertices respect the distance imposed on edges. Constraints (3.9) require that different colors assigned to the same vertex respect the minimum distance $d_{i,i}$ between each other (using the *allMinDistance* global constraint, since they form a small MinGEQ-CDGP-Unif subinstance). This formulation has $O(|V|q_{\max})$ variables (where $q_{\max} = \max_{i \in V} q_i$, that is, the largest color demand in the graph), since, for each color needed to each vertex, there is a variable, and $O(|E|q_{\max})$ constraints.

The second mathematical formulation approach is an integer programming model, which will be denoted by MinGEQ-CDGP-IP. It is based on models defined in [17], where we modified the formulation in order to make it more compact. A similar formulation was proposed independently in [19]. Two sets of variables are used:

- $x_{ic} = \begin{cases} 1 & \text{if color } c \text{ is assigned to vertex } i; \\ 0 & \text{otherwise.} \end{cases}$
- $z_{\max}$ = value of maximum color used in the solution (the coloring span).

Using these variables, MinGEQ-CDGP-IP is defined as follows:

$$\text{Minimize} \quad z_{\max} \tag{3.11}$$

$$\text{Subject to} \quad \sum_{c=1}^{U} x_{ic} = 1 \qquad (\forall i \in V) \tag{3.12}$$

$$x_{ic} + x_{je} \leq 1 \qquad (\forall (i,j) \in E;\ 1 \leq c, e \leq U \mid |c-e| < d_{i,j}) \tag{3.13}$$

$$z_{\max} \geq c x_{ic} \qquad (\forall i \in V;\ 1 \leq c \leq U) \tag{3.14}$$

$$x_{ic} \in \{0,1\} \qquad (\forall i \in V;\ 1 \leq c \leq U) \tag{3.15}$$

$$z_{\max} \in \mathbb{R} \tag{3.16}$$

In MinGEQ-CDGP-IP, the objective (3.11) is to minimize the value of variable $z_{\max}$, which will be the coloring span. Constraint set (3.12) ensures that all vertices must be colored. Constraint set (3.13) ensures that the absolute difference between the colors assigned to adjacent vertices is less than the distance given by the weight of the respective edge, then only one of the vertices can receive that color. Constraints (3.14) require that variable $z_{\max}$ be greater than or equal to any color used, so it will be the maximum color used. Constraints (3.15) require that variables in the set x_{ic} use only values 0 and 1, while constraint (3.16) states that $z_{\max}$ is a free variable, although its value will always be an integer, since, at the optimal solution, $z_{\max} = c x_{ic}$ for some $i \in V$ and $c \in [\![1, U]\!]$. The value U denotes the upper bound for colors to be used, since variables are indexed by vertex and color, so the color set must be limited. This IP model has $O(U|V|)$ variables (one for each pair of color and vertex) and $O(U|V| + U^2|E|)$ constraints, that is, it has pseudopolynomial size.

For MinGEQ-Multi-CDGP, the integer programming model can also be changed to accomodate multicoloring demands. Such modified formulation, which will be denoted as MinGEQ-Multi-CDGP-IP, is obtained by making two modifications to the previous model. The first one is changing the RHS of constraints (3.12) from 1 to q_i, which ensures that, instead of receiving only one color, each vertex i will receive q_i colors. The second one is expanding the set of constraints (3.13) in order to add new ones for ensuring that the same vertex i avoids using colors that violate the distance $d_{i,i}$, that is, there will be one constraint for each $(i,j) \in E;\ 1 \leq c, e \leq U$ such that $|c-e| < d_{i,j}$ and also for each $i \in V;\ 1 \leq c, e \leq U$ such that $|c-e| < d_{i,i}$. Note that this is equivalent to having an edge (i,i) for each vertex i, which would make the new constraints be automatically included in the original constraint set. The full MinGEQ-Multi-CDGP-IP formulation we propose is given below. These modifications do not impact the asymptotic size of the formulation.

$$\text{Minimize} \quad z_{\max} \tag{3.17}$$

$$\text{Subject to} \quad \sum_{c=1}^{U} x_{ci} = q_i \qquad (\forall i \in V) \tag{3.18}$$

$$x_{ic} + x_{je} \leq 1 \qquad (\forall (i,j) \in E;\ 1 \leq c, e \leq U \mid |c-e| < d_{i,j}) \tag{3.19}$$

$$x_{ic} + x_{ie} \leq 1 \qquad (\forall i \in V;\ 1 \leq c, e \leq U \mid |c-e| < d_{i,i}) \tag{3.20}$$

$$z_{\max} \geq c x_{ic} \qquad (\forall i \in V;\ 1 \leq c \leq U) \tag{3.21}$$

$$x_{ci} \in \{0,1\} \qquad (\forall i \in V;\ 1 \leq c \leq U) \tag{3.22}$$

$$z_{\max} \in \mathbb{R} \tag{3.23}$$

Another formulation that can be used to solve MinGEQ-Multi-CDGP (and MinGEQ-CDGP) is based on the one by [11], which was originally proposed for channel assignment problems. Instead of using a single integer variable to track the coloring span ($z_{\max}$ in the above IP formulations), a set of binary variables are used instead, as defined below:

- $u_c = \begin{cases} 1 & \text{if color } c \text{ is the maximum one used in the solution;} \\ 0 & \text{otherwise.} \end{cases}$

TABLE 1. Summary of constraint and integer programming formulations.

Formulation	Type	Problem	# Vars	# Constr.
MinGEQ-CDGP-CP	Constraint programming	BCP	$O(\|V\|)$	$O(\|E\|)$
MinGEQ-CDGP-Unif-CP		BCP w/ Unif. Dist. (Complete graphs)	$O(\|V\|)$	$O(\|V\|)$
MinGEQ-Multi-CDGP-CP		BMCP	$O(\|V\|q_{\max})$	$O(\|E\|q_{\max})$
MinGEQ-CDGP-IP	Integer programming	BCP	$O(U\|V\|)$	$O(U\|V\| + U^2\|E\|)$
MinGEQ-Multi-CDGP-IP		BMCP	$O(U\|V\|)$	$O(U\|V\| + U^2\|E\|)$
MinGEQ-Multi-CDGP-IP-FullBinary		BMCP	$O((U\|V\|) + U)$	$O(U^2\|V\| + U\|V\| + U^2\|E\|)$

The resulting model, which we will denote as MinGEQ-Multi-CDGP-IP-FullBinary, is then given by:

$$\text{Minimize} \quad \sum_{c=1}^{U} c u_c \tag{3.24}$$

$$\text{Subject to} \quad \sum_{c=1}^{U} x_{ci} = q_i \qquad (\forall i \in V) \tag{3.25}$$

$$x_{ic} + x_{je} \leq 1 \qquad (\forall (i,j) \in E;\ 1 \leq c, e \leq U \mid |c-e| < d_{i,j}) \tag{3.26}$$

$$x_{ic} + x_{ie} \leq 1 \qquad (\forall i \in V;\ 1 \leq c, e \leq U \mid |c-e| < d_{i,i}) \tag{3.27}$$

$$\sum_{c=1}^{U} u_c = 1 \tag{3.28}$$

$$x_{ic} + u_e \leq 1 \qquad (\forall i \in V;\ 1 \leq c, e \leq U \mid c > e) \tag{3.29}$$

$$x_{ci} \in \{0,1\} \qquad (\forall i \in V;\ 1 \leq c \leq U) \tag{3.30}$$

$$u_c \in \{0,1\} \qquad (\forall 1 \leq c \leq U) \tag{3.31}$$

In MinGEQ-Multi-CDGP-IP-FullBinary, constraint sets (3.25), (3.26) and (3.27) are the same as sets (3.18), (3.19) and (3.20) from MinGEQ-Multi-CDGP-IP. Constraints (3.28) ensure that only one color can be the maximum one used. The set of constraints (3.29) require that, if a certain color is the maximum one used, then no vertex can receive a color higher than the maximum. This formulation has $O((U|V|) + U)$ variables and $O(U^2|V| + U|V| + U^2|E|)$ constraints, making it larger than MinGEQ-Multi-CDGP-IP.

A summary of the constraint and integer programming formulations described in this section is given in Table 1.

3.1. Upper bounds for color sets

Both CP and IP models need a finite color set, which, as shown previously, consist of an interval of integers $[1, \ldots, U]$, where U is an upper bound for the coloring span. A trivial value for U can be calculated by summing the distances imposed on all edges plus 1, that is, $U = \sum_{(i,j)\in E} d_{i,j} + 1$. However, this upper bound is very weak, since, by summing all distances, we are ignoring the fact that colors can be reused by vertices not adjacent to each other, which makes the coloring span become a large value far from optimum. This also makes the color set have a large cardinality, which has a huge impact on the computing performance of these models, especially

the IP model, since the number of variables and constraints are proportional to the upper bound, which can lead to out-of-memory scenarios.

A better value for U is given by preprocessing the input graph, where a heuristic which does not need an explicit upper bound is applied to it. The span of the resulting solution is used as the value U when assembling CP and IP models for the input graph. A greedy algorithm for coloring the input graph can be used for this, where the vertices are processed following an order based on their color demands – vertices with higher demands are colored first. A color for a vertice i is determined by first setting its color candidate as $numCol[i] \times d_{i,i} + 1$, where $numCol[i]$ is the number of colors already assigned to i, and checking if it violates any separation constraint with any of its neighbor vertices. If a violation occurs, the color candidate is incremented by 1 and this checking is made again until a feasible color is found. The color is then assigned to i and, if its demands are not fully satisfied, an additional color is calculated and assigned to it. This is repeated until the graph is fully colored. Algorithm 1 gives pseudocode for this greedy heuristic. The upper bound is, then, the span of the solution returned by this method.

Algorithm 1. Greedy heuristic for generating starting solutions for BCP and BMCP.

Require: graph G (with set V of vertices and set E of edges), distances $d : E \cup \{(i,i) : i \in V\} \to \mathbb{Z}_{\geq 0}$, color demands $q : V \to \mathbb{N}$.

```
1: function GreedyStartingSolution(G = (V, E), d, w)
2:     V' ← V
3:     for each i ∈ V do
4:         numCol[i] ← 0
5:         colorAssign[i] ← ∅
6:     while V' ≠ ∅ do
7:         i ← arg max_{v∈V'} q_v
8:         while numCol[i] < q_i do
9:             candColor ← (numCol[i] × d_{i,i}) + 1
10:            violated ← true
11:            while violated = true do
12:                violated ← false
13:                for each j ∈ V − (V' ∪ {i}) do
14:                    for each k ∈ colorAssign[j] do
15:                        if |k − candColor| < d_{i,j} then
16:                            violated ← true
17:                            candColor ← candColor + 1
18:                            break
19:                    if violated = true then
20:                        break
21:            colorAssign[i] ← colorAssign[i] ∪ {candColor}
22:            numCol[i] ← numCol[i] + 1
23:            V' ← V' − {i}
24:    return colorAssign
```

4. Computational experiments

The constraint and integer programming formulations were implemented in C++ using IBM/ILOG CPLEX solver, version 12.5.1, and its CP Optimizer component. The resulting programs were executed in a Microsoft Azure A9 Virtual Machine, consisting of Intel Xeon E5-2670 processors (16 cores @ 2.6 GHz), 112 GB of RAM and Ubuntu Linux 14.04.1 LTS operating system. Both formulations used the standard parameters of the solver, but using only one thread, and each instance was limited to 48 hours of CPU time (172 800 s).

We used two benchmarks from the literature in our experiments. The first set of literature instances is known as GEOM, generated by Michael Trick [26] for BCP and its multicoloring variant, BMCP, and consists of 33

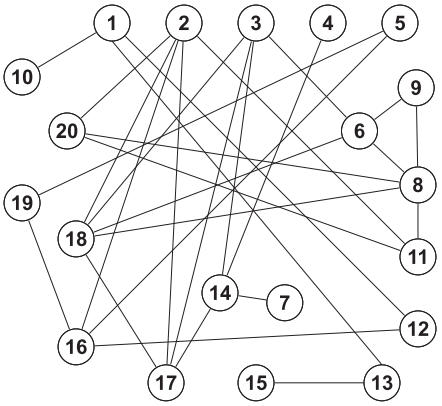
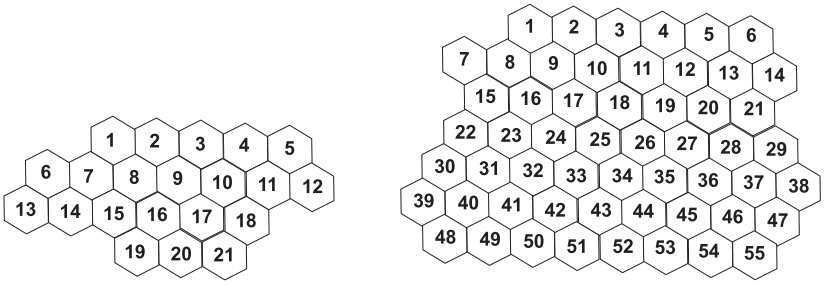


FIGURE 5. Graph layout of instance GEOM20b.



(a) Philadelphia. (b) Artificial.

FIGURE 6. Cellular networks used in Philadelphia and Artificial (55-cell) instances.

instances of three types: GEOMn are sparse graphs, while GEOMna and GEOMnb are denser graphs (where n is the number of vertices in the instance). An example of graph from this set of instances is given in Figure 5, where the graph from instance GEOM20b is shown.

The second set of instances consists of the classic Philadelphia (21 stations) and Helsinki (25 vertices) problems for MS-CAP, based on cellular networks from the cities of the same names, and an artificial problem (55 vertices) that extends a Philadelphia instance, as seen in [2,6,15]. The cellular models (hexagonal cells) for Philadelphia and artificial instances are given in Figure 6.

We remark that, for MS-CAP instances, the graph induced by the cellular network is not directly used as BCP/BMCP instances. Rather, another one is determined from the network layout, called interference graph, which takes into account the minimum distance between each pair of cells. The instances defined on such networks are based on a two-band buffering system, that is, each cell interferes only with others situated at most 2 units of distance apart [1]. To exemplify this concept, the interference graph for the 21-cell Philadelphia network is given in Figure 7. Since these instances are defined for BMCP only, we applied the suggestion by the authors of the GEOM instances in these as well to generate BCP instances.

When constructing the models for each instance, we executed the preprocessing discussed in Section 3.1 in order to obtain a feasible solution and an upper bound. To further help the solvers, we fed the entire starting solution to them, namely, we passed the solution as a starting point to CP Optimizer and as a MIP start to CPLEX, instead of only using the span as an upper bound. This is especially important for CPLEX, since it guarantees that an optimality gap is calculated as soon as the enumeration starts.

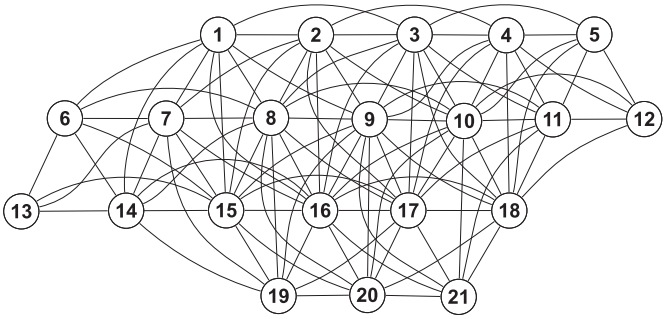


FIGURE 7. Interference graph for the 21-cell Philadelphia network. The distances imposed on the edges, as well as multicoloring demands, which would define loop edges, were omitted in the graph.

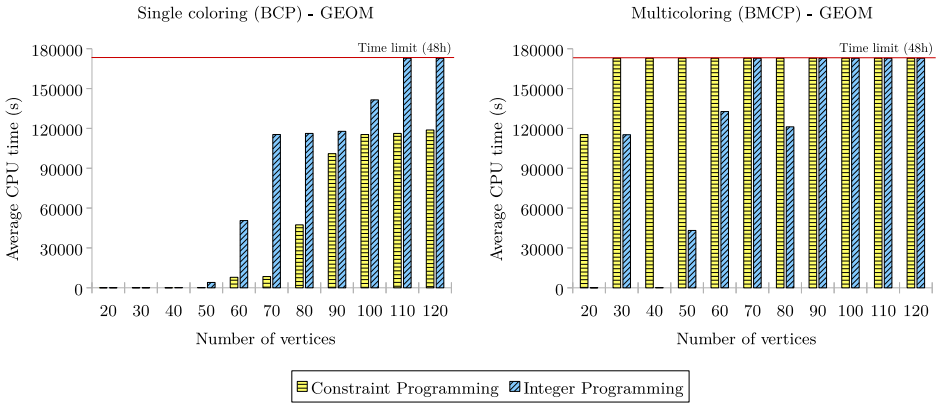


FIGURE 8. Number of vertices × CPU time needed to prove optimality (if found) for each method on GEOM instances.

The first results presented are for BCP. Table 2 shows the results for the GEOM BCP instances, where underlined values for the span (in Best Found and Best Reported columns) indicate that it is proved to be optimal using at least one of the exact approaches (CP or IP). Since the BCP variants are also used in the literature, we compared our results with the Discropt heuristic framework in [24] and the multistart iterated tabu search heuristic in [18] to verify the correctness of the solutions by the CP and IP formulations. For all sparse instances (the ones without "a" or "b" in the name), the constraint programming implementation was able to prove optimality. However, we emphasize that, for some instances, no method achieved the best solution presented in [24]. As noted in [18], no other work has obtained the same results, while our exact approaches reached the same best solutions for these instances obtained by other authors, which leads us to believe there is a mistake in [24], as marked in Table 2.

Table 3 shows the results for MS-CAP (Philadelphia, Helsinki and Artificial) instances without considering multicoloring demands. We note that, for each Philadelphia constraint matrix C^i_{21} where i is odd, by dropping the multicoloring demands, the instances become equal, since the only difference among them is the separation between colors of the same vertex. The same occurs for each even i. Given that, we grouped together results for each odd i (1, 3, 5 and 7) and for each even i (2, 4, 6, 8). Again, the CP formulation reaches optimality for each instance much faster, although runtimes are small due to the size of these relaxed instances.

TABLE 2. Results for the constraint and integer programming formulations applied on literature BCP instances (GEOM set). Underlined results in "Best Found" columns are optimal values.

Instance	$\|V\|$	$\|E\|$	Phan and Skiena [24]	Lai and Lu [18]	Constr. Progr. (CP Optimizer)		Integer Progr. (CPLEX)			
			Best reported	Best reported	Best found	Time (s)	Best found	Best LB	Gap (%)	Time (s)
GEOM20		40	20*	21	21	0.00	21	21.000	0.00	0.33
GEOM20a	20	57	20	20	20	0.02	20	20.000	0.00	0.95
GEOM20b		52	13	13	13	0.01	13	13.000	0.00	0.09
GEOM30		80	27*	28	28	0.05	28	28.000	0.00	0.88
GEOM30a	30	111	27	27	27	0.05	27	27.000	0.00	8.06
GEOM30b		111	26	26	26	0.03	26	26.000	0.00	2.27
GEOM40		118	27*	28	28	0.05	28	28.000	0.00	1.97
GEOM40a	40	186	38	37	37	1.39	37	37.000	0.00	278.66
GEOM40b		197	36	33	33	2.06	33	33.000	0.00	252.39
GEOM50		177	29	28	28	0.26	28	28.000	0.00	21.44
GEOM50a	50	288	54	50	50	374.42	50	50.000	0.00	3457.25
GEOM50b		299	40	35	35	144.69	35	35.000	0.00	8494.52
GEOM60		245	34	33	33	1.12	33	33.000	0.00	45.73
GEOM60a	60	339	54	50	50	684.59	50	50.000	0.00	16 755.65
GEOM60b		426	47	41	41	22 915.94	41	41.000	0.00	134 996.77
GEOM70		337	40	38	38	2.39	38	38.000	0.00	533.53
GEOM70a	70	529	64	61	61	24 798.03	≤62	38.000	38.71	172 815.55
GEOM70b		558	54	47	47	534.65	≤49	44.0000	10.20	172 834.40
GEOM80		429	44	41	41	8.18	41	41.000	0.00	3019.18
GEOM80a	80	692	69	63	63	87 770.77	≤65	39.0000	40.00	172 803.55
GEOM80b		743	70	60	60	54 320.89	≤66	21.0000	68.18	172 800.25
GEOM90		531	48	46	46	55.18	46	46.000	0.00	7768.62
GEOM90a	90	879	74	63	63	130 050.12	≤72	7.000	90.28	173 100.57
GEOM90b		950	83	69	≤69	172 800.00	≤85	2.1127	97.51	172 802.83
GEOM100		647	55	50	50	545.79	50	50.0000	0.00	78 836.94
GEOM100a	100	1092	84	67	≤70	172 800.01	≤85	3.0863	96.37	172 824.54
GEOM100b		1150	87	72	≤71	172 800.02	≤101	2.2271	97.75	172 840.38
GEOM110		748	59	50	50	2982.24	50	50.0000	0.00	170 043.88
GEOM110a	110	1317	88	72	≤73	172 800.01	≤97	2.1963	97.70	172 811.66
GEOM110b		1366	87	78	≤79	172 800.01	≤99	2.2208	97.76	172 821.35
GEOM120		893	67	59	59	10 778.18	≤60	24.0000	60.00	173 296.11
GEOM120a	120	1554	101	82	≤84	172 800.01	≤110†	2.1039	98.09	173 181.91
GEOM120b		1611	103	84	≤85	172 800.01	≤113†	2.1245	98.12	173 187.16

*Results lower than the obtained optimum – possibly wrong in the corresponding work. †No solution found by B&C, but CPLEX returns the initial greedy heuristic solution.

The next experiments are for BMCP. For these instances, a trivial lower bound can be calculated as $L = \max[(d_{i,i}(q_i - 1)) + 1]$, as shown by [2]. This value is also inserted in the models by adding a single constraint requiring that the objective value be greater than or equal to L, which helps the enumeration end faster when the optimal solution has span equivalent to this lower bound, especially when using CP. It also enables IP to report an optimality gap even when the root node takes too much time to find even the linear relaxation solution.

Table 4 shows results obtained for the GEOM instances. We compared our results for these instances with the same multistart iterated tabu search heuristic from [18], where the algorithm for BCP is applied to BMCP instances by expanding the vertices into cliques as shown in Section 2. Furthermore, Table 5 shows results

TABLE 3. Results for the constraint and integer programming formulations applied on literature MS-CAP instances (Philadelphia, Helsinki and Artificial) without multicoloring demands. Underlined results in "Best Found" columns are optimal values. Since IP reaches all optimal solutions, the best LB has been omitted in the table.

Const. matrix	$\|V\|$	$\|E\|$	Constr. Progr. (CP optimizer)		Integer Progr. (CPLEX – B&C)	
			Best found	Time (s)	Best found	Time (s)
C_{21}^1, C_{21}^3, C_{21}^5 and C_{21}^7	21	102	7	0.40	7	0.87
C_{21}^2, C_{21}^4, C_{21}^6 and C_{21}^8			9	0.06	9	2.66
C_{25}^1	25	134	8	4.71	8	1.90
C_{55}^1	55	362	7	0.79	7	30.63

TABLE 4. Results for the constraint and integer programming formulations applied on literature BMCP instances (GEOM set). Underlined results in "Best Found" columns are optimal values.

				Lai and Lu [18]	Constr. Progr. (CP Optimizer)		Integer Progr. (CPLEX – B&C)			
Instance	$\|V\|$	$\|E\|$	Trivial LB	Best reported	Best found	Time (s)	Best found	Best LB	Gap (%)	Time (s)
GEOM20		40	91	149	≤149	172 800.01	149	149.0000	0.00	15.17
GEOM20a	20	57	91	169	≤169	172 800.01	169	169.0000	0.00	18.49
GEOM20b		52	21	44	44	476.92	44	44.0000	0.00	1.58
GEOM30		80	91	160	≤160	172 800.01	≤160	159.0000	0.62	172 830.72
GEOM30a	30	111	91	209	≤215	172 800.01	≤211	189.0000	10.43	172 813.47
GEOM30b		111	21	77	≤77	172 800.00	77	77.0000	0.00	41.87
GEOM40		118	91	167	≤168	172 800.01	167	167.0000	0.00	1192.28
GEOM40a	40	186	91	213	≤225	172 800.01	213	213.0000	0.00	111 262.08
GEOM40b		197	21	74	≤74	172 800.00	74	74.0000	0.00	17 027.77
GEOM50		177	91	224	≤226	172 800.02	224	224.0000	0.00	52 450.85
GEOM50a	50	288	91	314	≤332	172 800.03	≤361	95.5218	73.54	172 820.13
GEOM50b		299	21	83	≤85	172 800.00	≤87	52.0000	40.23	172 819.47
GEOM60		245	91	258	≤259	172 800.02	258	258.0000	0.00	156 987.80
GEOM60a	60	339	91	356	≤380	172 800.03	≤448	93.5801	78.93	172 813.01
GEOM60b		426	21	113	≤117	172 800.01	≤125	34.0000	72.80	172 897.07
GEOM70		337	91	270	≤284	172 800.03	≤305	94.2092	69.11	172 804.56
GEOM70a	70	529	91	467	≤483	172 800.05	≤578	91.0000	84.26	172 839.51
GEOM70b		558	21	116	≤123	172 800.01	≤134	22.7359	83.03	172 805.88
GEOM80		429	91	381	≤395	172 800.04	≤511	95.2644	80.19	172 809.70
GEOM80a	80	692	91	361	≤382	172 800.05	≤479	91.0000	81.00	172 885.02
GEOM80b		743	21	139	≤145	172 800.01	≤170	23.0547	86.44	172 820.56
GEOM90		531	91	330	≤342	172 800.05	≤423	93.2736	77.73	172 811.82
GEOM90a	90	879	91	375	≤392	172 800.07	≤452	91.0000	79.87	172 830.60
GEOM90b		950	21	144	≤156	172 800.01	≤212	22.2574	89.50	172 844.07
GEOM100		647	91	404	≤426	172 800.07	≤493	94.2797	80.88	173 190.69
GEOM100a	100	1092	91	442	≤465	172 800.08	≤596	91.0000	84.73	172 871.84
GEOM100b		1150	21	156	≤169	172 800.02	≤220	21.0000	90.45	172 810.33
GEOM110		748	91	381	≤399	172 800.07	≤500	91.0000	81.80	172 840.98
GEOM110a	110	1317	91	488	≤527	172 800.10	≤610	91.0000	85.08	173 095.42
GEOM110b		1366	21	204	≤207	172 800.01	≤250	22.0001	91.20	172 835.82
GEOM120		893	91	396	≤427	172 800.06	≤505	93.9762	81.39	172 923.18
GEOM120a	120	1554	91	554	≤585	172 800.16	≤641	91.0000	85.80	173 312.14
GEOM120b		1611	21	189	≤202	172 800.01	≤247	21.8723	91.14	172 852.82

TABLE 5. Results for the constraint and integer programming formulations applied on literature MS-CAP instances (Philadelphia, Helsinki and Artificial). Underlined results in "Best Found" columns are optimal values. Since IP reaches all optimal solutions, the best LB has been omitted in the table.

Const. matr.	Demd. vect.	$\|V\|$	$\|E\|$	Lower bound	Chakraborty [2]	Kendall and Mohamad [15]	Kim *et al.* [16]	Constr. Progr. (CP Optimizer)		Integer Progr. (CPLEX – B&C)	
					Best reported	Best reported	Best reported	Best found	Time (s)	Best found	Time (s)
C_{21}^1	D_{21}^1	21	102	533	533	533	533	533	4.20	533	0.50
C_{21}^1	D_{21}^2			309	309	309	309	309	1.34	309	1.22
C_{21}^2	D_{21}^1			533	533	533	533	533	10.53	533	308.04
C_{21}^2	D_{21}^2			309	309	309	309	309	625.93	309	165.54
C_{21}^3	D_{21}^1			457	457	457	–	457	3.96	457	0.39
C_{21}^3	D_{21}^2			265	265	265	–	265	3.54	265	1.52
C_{21}^4	D_{21}^1			457	457	457	–	457	41.24	457	202.01
C_{21}^4	D_{21}^2			265	265	265	–	≤ 266	172 800.06	265	214.01
C_{21}^5	D_{21}^1			381	381	381	381	381	3.23	381	0.29
C_{21}^5	D_{21}^2			221	221	221	221	221	100.81	221	5.09
C_{21}^6	D_{21}^1			381	463	435	427	≤ 449	172 800.08	427	6827.49
C_{21}^6	D_{21}^2			221	273	268	253	≤ 266	172 800.04	253	2026.67
C_{21}^7	D_{21}^1			305	305	305	–	305	12.85	305	1.10
C_{21}^7	D_{21}^2			177	197	185	–	≤ 180	172 800.06	180	24.54
C_{21}^8	D_{21}^1			305	465	444	–	≤ 435	172 800.07	427	1185.27
C_{21}^8	D_{21}^2			177	278	271	–	≤ 267	172 800.06	253	1116.45
C_{25}^1	D_{25}^3	25	134	21	73	73	–	≤ 73	172 800.00	73	1.10
C_{25}^1	D_{25}^4			89	121*	200	–	≤ 200	172 800.07	200	2.18
C_{55}^1	D_{55}^5	55	362	309	309	309	–	309	11 078.95	309	460.12
C_{55}^1	D_{55}^6			71	79	72	–	71	6.33	71	28.56

*Results lower than the obtained optimum – possibly wrong in the corresponding work.

for the MS-CAP (Philadelphia, Helsinki and Artificial) instances in their original forms (that is, including multicoloring), which were compared with the constructive heuristic from [2], the local search algorithm from [15] and the memetic algorithm from [16].

For BMCP, the efficiency between CP and IP is practically reversed: most instances are solved to optimality faster with IP than CP. In fact, for the MS-CAP instances, we were able to obtain all optimal values using IP. This is explained by taking into account, as discussed in Section 3, that the IP model is able to consider multicoloring demands without expanding the number of vertices, only having to change a set of constraints and add another set for the different colors for each vertex. However, the CP model has to grow more, since, essentially, a BMCP instance is treated as a special BCP one, since the number of variables increases and a new set of constraints is introduced. Figure 8 shows this difference in efficiency between methods. However, CP still has an advantage in BMCP: when it is unable to solve a problem to optimality in the given time limit, the solution that it returns has a better quality than the one found by IP (that is, the coloring span of the solution found by CP is lower than the one found by IP). This happens because CP has explicit information about which colors each vertex can assume, instead of calculating such colors.

TABLE 6. Number of branches and fails (for CP) and of global cuts of each type (for IP) applied to GEOM instances with unit demands.

Instance	Num. Vert.	Const. Prog. (CP Optimizer)		Integer Progr. (CPLEX)				
		# Branches	# Fails	# Clique cuts	# Implied bound cuts	# MIP rounding cuts	# Zero-half cuts	# Gomory fractional cuts
GEOM20		972	462	9	15	0	8	10
GEOM20a	20	4667	2298	3	27	0	2	9
GEOM20b		2224	1096	14	7	0	5	0
GEOM30		10102	4782	13	62	0	4	31
GEOM30a	30	9024	4353	25	473	0	20	60
GEOM30b		6651	3210	9	194	0	11	49
GEOM40		9965	4699	13	130	0	3	56
GEOM40a	40	206960	98635	43	4	117	3	1
GEOM40b		275519	131721	14	543	0	30	15
GEOM50		57869	27159	27	0	0	9	0
GEOM50a	50	40966958	19591854	227	13	138	12	0
GEOM50b		14438962	6973218	564	3	6	18	0
GEOM60		178478	82471	63	0	0	0	2
GEOM60a	60	59350249	28292260	241	19	182	28	24
GEOM60b		1700740733	807817043	1691	25	176	23	0
GEOM70		320560	148877	128	0	12	3	1
GEOM70a	70	1662200599	781294815	881	6	192	20	0
GEOM70b		301138496	143985463	1171	21	226	46	5
GEOM80		2173324	1008986	372	5	131	19	0
GEOM80a	80	8859155916	4149659761	761	30	389	85	5
GEOM80b		3687195162	1739030200	480	29	303	138	0
GEOM90		3841482	1748958	471	2	223	31	9
GEOM90a	90	8424930433	3953124503	143	3	411	349	0
GEOM90b		6454145085	3036820947	317	11	417	93	0
GEOM100		33141115	15198269	702	11	221	54	27
GEOM100a	100	6622094014	3084753118	330	2	1367	2162	245
GEOM100b		5409742123	2511274478	130	301	0	567	136
GEOM110		12496119255	5760860721	426	16	320	54	2
GEOM110a	110	5930484572	2724545532	151	501	0	619	79
GEOM110b		4177753606	1922426902	118	754	0	460	0
GEOM120		637959908	289378147	1015	19	357	128	8
GEOM120a	120	5560296354	2542244856	172	782	0	678	0
GEOM120b		4003420813	1841115383	273	1258	0	959	0

We also detected a mistake in [2], where the heuristic result presented in it for constraint matrix C^1_{25} and demand vector D^4_{25} is better (with objective function value 121) than the exact solutions obtained by both CP and IP (with objective function value 200). In fact, no other work in the literature obtained a solution with span lower than 200.

Finally, the numbers of branches and paths that do not reach a solution in the CP enumeration and generated cuts of each type in the IP enumeration are given in Tables 6 and 7 for instances with unit color demands and in Tables 8 and 9 for instances with multicoloring demands. We note that there is not a clear correlation between the instance size and this data, indicating that these parts of the algorithms are sensitive to the individual

TABLE 7. Number of branches and fails (for CP) and of global cuts of each type (for IP) applied to MS-CAP instances (Philadelphia, Helsinki and Artificial) without multicoloring demands.

Instance	Num. Vert.	Const. Prog. (CP Optimizer)		Integer Progr. (CPLEX)				
		# Branches	# Fails	# Clique cuts	# Implied bound cuts	# MIP rounding cuts	# Zero-half cuts	# Gomory fractional cuts
C_{21}^1, C_{21}^3, C_{21}^5 and C_{21}^7	21	49177	24602	3	5	0	9	0
C_{21}^2, C_{21}^4, C_{21}^6 and C_{21}^8		6033	2970	5	5	0	58	3
C_{25}^1	25	176565	87496	0	0	0	0	0
C_{55}^1	55	49096	24214	0	0	0	0	0

TABLE 8. Number of branches and fails (for CP) and of global cuts of each type (for IP) applied to GEOM instances with multicoloring demands.

Instance	Num. Vert.	Const. Prog. (CP Optimizer)		Integer Progr. (CPLEX)				
		# Branches	# Fails	# Clique cuts	# Implied bound cuts	# MIP rounding cuts	# Zero-half cuts	# Gomory fractional cuts
GEOM20		1847656377	823996186	26	33	0	22	0
GEOM20a	20	2060224899	926128135	35	378	0	92	91
GEOM20b		23433594	11236673	19	28	0	47	49
GEOM30		1594445634	702092983	345	1	7	29	0
GEOM30a	30	778845728	343378790	1234	6	189	3	0
GEOM30b		4336696005	2043720645	23	96	0	17	53
GEOM40		1101215203	477955956	186	287	0	158	0
GEOM40a	40	853750393	376905743	131	7	322	4	0
GEOM40b		2915214186	1378502269	361	1	4	2	0
GEOM50		858223844	366277155	575	0	79	31	0
GEOM50a	50	373860395	159472193	183	446	0	235	4
GEOM50b		2205618883	1031493835	1498	2	295	7	0
GEOM60		884613825	374341633	468	1	34	38	0
GEOM60a	60	327100218	135221648	1109	335	0	1213	0
GEOM60b		1625337918	741993923	143	3	456	7	0
GEOM70		480579106	200225715	497	1363	0	275	0
GEOM70a	70	178618063	71625162	0	0	0	0	0
GEOM70b		1153010252	520892432	133	2	645	165	29
GEOM80		491579072	131225337	786	1842	0	1343	85
GEOM80a	80	206713326	84260206	0	0	0	0	0
GEOM80b		972753631	434139515	328	1065	0	690	97
GEOM90		289804704	110251804	1321	0	0	1894	0
GEOM90a	90	203904382	82252836	0	0	0	0	0
GEOM90b		677312582	300211365	344	481	0	877	0
GEOM100		219953270	80390988	0	0	0	0	0
GEOM100a	100	134018202	52245146	0	0	0	0	0
GEOM100b		511570190	224573165	0	0	0	0	0
GEOM110		170668054	64299043	0	0	0	0	0
GEOM110a	110	113908674	41331333	0	0	0	0	0
GEOM110b		409339582	177985395	0	0	0	0	0
GEOM120		207980564	81060477	0	0	0	0	0
GEOM120a	120	82058712	30432596	0	0	0	0	0
GEOM120b		386743755	164997551	0	0	0	0	0

TABLE 9. Number of branches and fails (for CP) and of global cuts of each type (for IP) applied to MS-CAP instances (Philadelphia, Helsinki and Artificial).

Const. Matr.	Demd. Vect.	Num. Vert.	Const. Prog. (CP Optimizer)		Integer Progr. (CPLEX)				
			# Branches	# Fails	# Clique cuts	# Implied bound cuts	# MIP rounding cuts	# Zero-half cuts	# Gomory fractional cuts
C_{21}^1	D_{21}^1	21	2555	504	0	0	0	0	0
C_{21}^1	D_{21}^2		469	4	0	0	0	0	0
C_{21}^2	D_{21}^1		6610	1505	0	0	0	0	0
C_{21}^2	D_{21}^2		822377	309790	0	0	0	0	0
C_{21}^3	D_{21}^1		2451	504	0	0	0	0	0
C_{21}^3	D_{21}^2		2874	505	0	0	0	0	0
C_{21}^4	D_{21}^1		20982	5811	0	0	0	0	0
C_{21}^4	D_{21}^2		44845827	17108454	0	0	0	0	0
C_{21}^5	D_{21}^1		2305	504	0	0	0	0	0
C_{21}^5	D_{21}^2		158368	50239	0	0	0	0	0
C_{21}^6	D_{21}^1		180812651	65917318	160	4	210	116	0
C_{21}^6	D_{21}^2		347622756	121153772	170	1	37	159	0
C_{21}^7	D_{21}^1		4106	505	0	0	0	0	0
C_{21}^7	D_{21}^2		352244757	132070514	4	0	0	8	35
C_{21}^8	D_{21}^1		199948906	75399919	163	1	379	224	0
C_{21}^8	D_{21}^2		451729822	146753868	225	204	0	190	0
C_{25}^1	D_{25}^3	25	2435656356	1064961796	2	134	0	81	143
C_{25}^1	D_{25}^4		374343491	115700707	0	0	0	0	0
C_{55}^1	D_{55}^5	55	7809104	2371461	0	0	0	0	0
C_{55}^1	D_{55}^6		27897	9552	0	0	0	0	0

values of distances and demands in the instances. We also conjecture that the input graph layout also impacts in the enumeration, since some known cuts applied to coloring problems, such as clique cuts [23], are derived according to the graph to be colored.

5. Concluding remarks

In this paper, we addressed channel assignment in wireless networks as special graph coloring with distance constraints, and explored some feasibility properties on them, by proving some specific graph classes which admit or do not admit solutions. The special coloring problems with distance constraints were modeled by distance geometry being considered as the general problem. We have assigned the vertices on the real line, according to the distances between adjacent vertices. Beyond that, we have described feasibility conditions for some classes of graphs.

We employed constraint and integer programming formulations, which were implemented using computational OR tools, and applied them to instances from the literature in order to verify which mathematical modeling tool is best for these distance coloring models. Since the constraints from the problems are naturally transported to constraint programming, its implementation reaches the optimal solution much faster than the integer programming one for BCP instances. However, for BMCP, due to needing the expansion of color demands into

additional variables, constraint programming becomes slower than integer programming, which does not need such expansion.

Ongoing and future works include improving the CP formulation by domain reduction and more specific constraints, and also use hybrid methods, combining both CP and IP, as well as heuristics, in order to solve the distance coloring models faster. The study of the problems posed to specific classes of graphs, in order to establish the characterization of feasibility conditions for them, is subject of the research in progress.

Acknowledgements. Supported by CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior)*, CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*), FAPEAM (Fundação de Amparo a Pesquisa do Estado do Amazonas) and Méliuz – Brazil.

References

[1] G. Audhya, K. Sinha, S. Ghosh and B. Sinha, A survey on the channel assignment problem in wireless networks. *Wireless Commun. Mobile Comput.* **11** (2011) 583–609.

[2] G. Chakraborty, An efficient heuristic algorithm for channel assignment problem in cellular radio networks. *IEEE Trans. Veh. Technol.* **50** (2001) 1528–1539.

[3] D. Costa, On the use of some known methods for T-coloring of graphs. *Ann. Oper. Res.* **41** (1999) 343–358.

[4] R. de Freitas, B. Dias, N. Maculan and J. Szwarcfiter, Distance geometry approach for special graph coloring problems. Preprint arXiv:1606.04978 (2016).

[5] R. de Freitas, B. Dias, N. Maculan and J. Szwarcfiter, On distance graph coloring problems. *Int. Trans. Oper. Res.* (2019). DOI: 10.1111/itor.12626.

[6] B. Dias, *Theoretical models and algorithms for optimizing channel assignment in wireless mobile networks*. Master's thesis, Universidade Federal do Amazonas, Brazil (2014). In Portuguese.

[7] B. Dias, R. de Freitas, N. Maculan and P. Michelon, Solving the bandwidth coloring problem applying constraint and integer programming techniques. *Optim. Online (e-print)* (2016). Available at http://www.optimization-online.org/DB_HTML/2016/06/5514.html.

[8] R. Dorne and J. Hao, Tabu search for graph coloring, T-coloring and set T-colorings, edited by S. Voss, S. Martello, I.H. Osman and C. Roucairol. In: Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization. Springer, New York, NY (1999) 77–92.

[9] P. Galinier and A. Hertz, A survey of local search methods for graph coloring. *Comput. Oper. Res.* **33** (2006) 2547–2562.

[10] M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co, New York, NY (1979).

[11] A.I. Giortzis and L.F. Turner, Application of mathematical programming to the fixed channel assignment problem in mobile radio networks. *IEE Proc. Commun.* **144** (1997) 257–264.

[12] A. Gueham, A. Nagih, H.A. Haddadene and M. Masmoudi, Graph coloring approach with new upper bounds for the chromatic number: team building application. *RAIRO:OR* **52** (2018) 807–818.

[13] W. Hale, Frequency assignment: theory and applications. *Proc. IEEE* **25** (1980) 1497–1514.

[14] R. Karp, Reducibility among combinatorial problems, edited by R.E. Miller and J.W. Thatcher. In: Complexity of Computer Computations. Plenum Press, New York, NY (1972) 85–103.

[15] G. Kendall and M. Mohamad, Solving the fixed channel assignment problem in cellular communications using an adaptive local search, edited by E. Burke and M. Trick. In: Vol. 3616 of *5th International Conference for the Practice and Theory of Automated Timetabling (PATAT 2004). Lecture Notes in Computer Science*. Springer, Heidelberg (2005).

[16] S. Kim, A.E. Smith and J. Lee, A memetic algorithm for channel assignment in wireless fdma systems. *Comput. Oper. Res.* **34** (2007) 1842–1856.

[17] A. Koster, *Frequency assignment: models and algorithms*. Ph.D. thesis, Universiteit Maastricht (1999).

[18] X. Lai and Z. Lü, Multistart iterated tabu search for bandwidth coloring problem. *Comput. Oper. Res.* **40** (2013) 1401–1409.

[19] V. Mak, Polyhedral studies for minimum-span graph labelling with integer distance constraints. *Int. Trans. Oper. Res.* **14** (2007) 105–121.

[20] E. Malaguti and P. Toth, An evolutionary approach for bandwidth multicoloring problems. *Eur. J. Oper. Res.* **189** (2008) 638–651.

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

[22] A. Mehrotra and M. Trick, A branch-and-price approach for graph multi-coloring, edited by E.K. Baker, A. Joseph, A. Mehrotra and M. Trick. In: Vol. 37 of *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies. Operations Research/Computer Science Interfaces Series*. Springer, New York, NY (2007) 15–29.

[23] I. Méndez-Díaz and P. Zabala, A Branch-and-Cut algorithm for graph coloring. *Dis. Appl. Math.* **154** (2006) 826–847.

[24] V. Phan and S. Skiena, Coloring graphs with a general heuristic search engine. In: Proceedings of the Computational Symposium on Graph Coloring and Its Generalizations. Ithaca, NY (2002) 92–99.

[25] J.B. Saxe, Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proceedings of 17th Allerton Conference in Communications, Control and Computing. (1979) 480–489.

[26] M. Trick, A. Mehrotra, and D. Johnson. COLOR02/03/04: Graph coloring and its generalizations. In: Proceedings of the Computational Symposium on Graph Coloring and Its Generalizations. Ithaca, NY (2002).

[27] S. Varone and N. Zufferey, Three tabu search methods for the MI-FAP applied to 802.11 networks. *RAIRO:OR* **42** (2008) 501–514.

[28] M. Vasquez, A. Dupont and D. Habet, Consistency checking within local search applied to the frequency assignment with polarization problem. *RAIRO:OR* **37** (2003) 311–323.