Exact and heuristic algorithms for MinMax Channel Assignment Problems modeled as distance geometry graph coloring

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

[1] PPGI - Instituto de Computação - Universidade Federal do Amazonas, Brazil
[2] COPPE - IM - NCE - Universidade Federal do Rio de Janeiro, Brazil
[3] CERI/LIA - Université d'Avignon et des Pays de Vaucluse, Francel

{bruno.dias,rosiane}@icomp.ufam.edu.br, {maculan,jayme}@cos.ufrj.br,
{philippe.michelon@univ-avignon.fr}

ABSTRACT

One of the most important combinatorial optmization problems is graph coloring. There are several versions of this problem, involving additional constraints on the vertices or edges. They constitute models for real applications, such as channel assignment in mobile wireless networks. In this work, we consider some of these graph coloring models, involving distance constraints on the edges. The vertices of the graphs are considered as embedded on the real line and the coloring is treated as an assignment of integers to the vertices, while the distances correspond to intersections of line segments. We formulate different such coloring problems in integer and constraint programming, proposing implicit enumeration methods for some of these optimization problems. An empirical analysis was undertaken, considering equality and inequality constraints, comparing the performance of the proposed solutions, with that of some existing methods.

Keywords: bandwidth coloring; integer and constraint programming; distance geometry; graph theory.
Main Area: Combinatorial Optimization.

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 that 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 graph coloring problem, which consists in finding the chromatic number of a graph, is one of the most important combinatorial optimization problems and it is known to be NP-hard karp:1972.

There are several versions of this classic vertex coloring problem, involving additional constraints, in both edges as vertices of the graph, with a number of practical applications as well as theoretical challenges. 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, 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. There are three main types of interferences: *co-channel*, among calls of two transmitters using the same channels; *adjacent channel*, among calls of two transmitters using adjacent channels and *co-site*, among calls on the same cell that do not respect a minimal separation. It is necessary to assign channels to the calls such that interference is avoided and the spectrum usage is minimized audhya:2011,koster:2010,koster:1999.

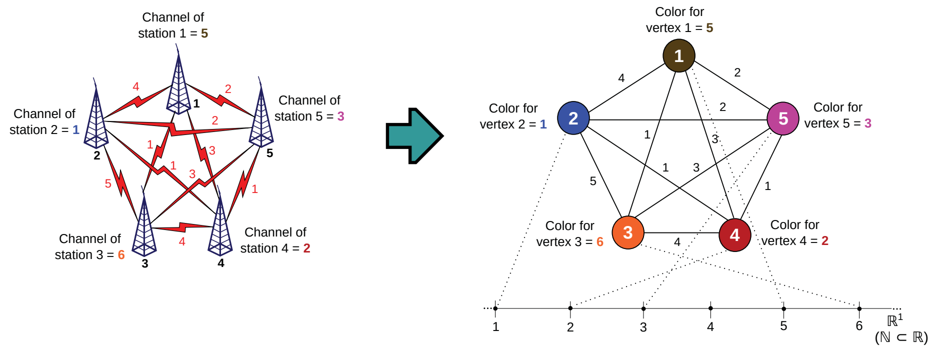


Figure 1: Example of channel assignment with distance constraints, where the separation is given by the weight in each edge. The image on the right shows the network as an undirected graph and the projection of vertices on the real number line, but considering only natural numbers.

Thus, the channel assignment scenario is modeled as a graph coloring problem by considering each transmitter as a vertex in a undirected graph and the channels to be assigned as the colors that 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) where, for each edge, the absolute difference between colors assigned to each vertex must not be in a given forbidden set hale:1980. Also, the Bandwidth Coloring Problem, a special case of T-coloring where the absolute difference between colors assigned to each vertex must be greater or equal a certain value malaguti:2010,lai:2013, and the coloring problem with restrictions of adjacent colors (COLRAC), where there is a restriction graph for which adjacent colors in it cannot be assigned to adjacent vertices akihiro:2002.

The separation among channels is 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, as can be seen on Figure 1. One method to solve DG problems is the branch-and-prune approach proposed by lavor:2012:1,lavor:2012:2, where a solution is constructed and, if at some point a distance constraint is violated, then we stop the building of the current solution (prune) and try another option in the search space. See also dias:2014, dias:2013:1, dias:2012. For graph theoretic concepts and terminology, see the book by bondy:2008.

The main contribution of this paper consists of a distance geometry approach for special cases of T-coloring problems with distance constraints, involving a study of graph classes for which some of these distance coloring problems are unfeasible, and branch-prune-and-bound algorithms, combining concepts from the branch-and-bound method and constraint programming, for the considered problems.

The remainder of this paper is organized as follows. Section 2 defines the distance geometry models for some special graph coloring problems. Section 3.1 formulates the branch-prune-and-bound (BPB) algorithms proposed for the problems. Section 4 shows results of some experiments done with the BPB algorithms. Finally, Section 5 concludes the paper and states the next steps of ongoing research.

2. Distance geometry and graph colorings

We are proposing an approach in distance geometry for special vertex coloring problems with distance constraints, based on the Discretizable Molecular Distance Geometry Problem (DMDGP), which is a special case of the Molecular Distance Geometry Problem, introduced by lavor:2012:1, where the set of V vertices from the input graph G are ordered such that the 4-cliques of the graph appear consecutively in the ordering $(\forall i \in \{4, \dots, n\}\ \forall j, k \in \{i-3, \dots, i\}\ (\{j, k\} \in E))$. Furthermore, a strict triangular inequality holds $(\forall i \in \{2, \dots, n-1\}\ d_{i-1,i+1} < d_{i-1,i} + d_{i,i+1})$. All coordinates are given in $\mathbb{R}^3$ space. The position for a point i (where $i \geq 4$) can be determined using the positions of the previous three points $i-1, i-2$ and $i-3$ by intersecting three spheres with radii $d_{i-3,i}, d_{i-2,i}$ and $d_{i-1,i}$, obtaining two possible points that are checked for feasibility.

A similar reasoning can be used in vertex coloring problems with distance constraints, where the distances that must be respected involve the absolute difference between two values $x(i)$ and $x(j)$ (respectively, the color points attributed to i and j), but for these problems the space considered is actually unidimensional. The positioning of a vertex i can be determined by using a neighbor j that is already positioned. Thus, we have a *0-sphere*, consisting of a projection of a 1-sphere (a circle), which itself is a projection of a 2-sphere (the three-dimensional sphere), as shown in Figure 2. The 0-sphere is a line segment with a radius $d_{i,j}$, and feasible colorings consist of treating the intersections of these 0-spheres. Figure 3 exemplifies the correlations between these types of spheres, and shows the example from Figure 1 as the positioning of these line segments.

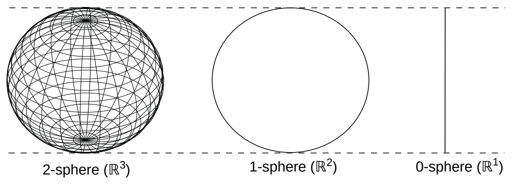

2-sphere ($\mathbb{R}^3$)
1-sphere ($\mathbb{R}^2$)
0-sphere ($\mathbb{R}^1$)

Figure 2: Some types of n-spheres. A $(n-1)$-sphere is a projection of a n-sphere on a lower dimension.

In this work, we focus on problems with exact distances between colors, and also in the analysis of different types of branch-prune-and-bound algorithms and integer programming models.

Based on DMDGP, which is a decision problem involving equalities distance constraints, the basic distance graph coloring model we consider involves also equality constraints between colors of two neighbor vertices i and j. That is, the absolute difference between them must be exactly equal an arbitrary weight imposed on the edge (i, j), and consists of finding a solution which satisfies all given constraints. We can formally define as follows.

Given a graph $G = (V, E)$, we define $x(i)$ as a positive integer value, corresponding to a color of the vertex $i \in V(G)$, and $d_{i,j}$ is a positive integer weight associated to an edge $(i, j) \in E(G)$. A variation of the classic graph coloring problem consists in finding the minimum *span* of G, that is, in determining that the maximum $x(i)$, or used color, be the minimum possible. Based on these preliminar definitions, we describe the following distance geometry vertex coloring problems.

Definition 1. ***Coloring Distance Geometry Problem (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 $x : V \to \mathbb{R}$ (that is, an embedding of G on the real number line) such that $|x(i) - x(j)| = d_{i,j}$ for each $(i, j) \in E$.*

CDGP involves equality constraints and, thus, named as Equal Coloring Distance Geometry Problem and labelled as **EQ-CDGP**. A solution for this problem, consists of a tree, whose

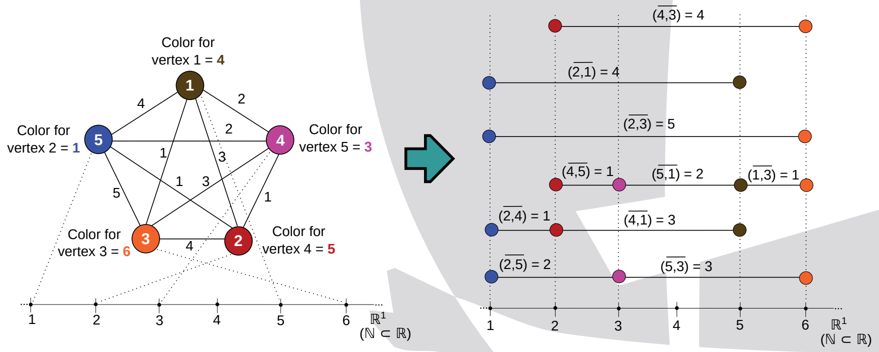


Figure 3: Example from Figure 1 using 0-spheres (line segments).

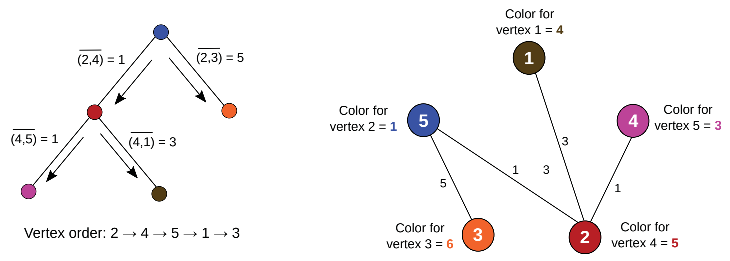
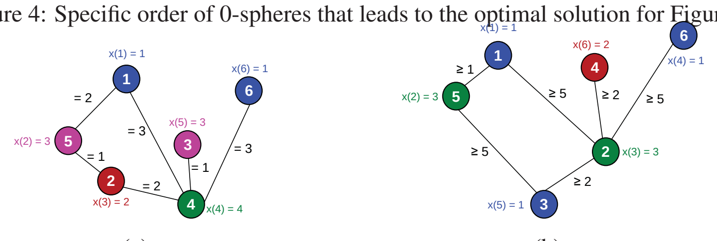


Figure 4: Specific order of 0-spheres that leads to the optimal solution for Figure 1.



(a) (b)

Figure 5: Examples of instances for distance coloring models and feasible solutions for them.

vertices are colored with colors that respect the inequality constraints involving the weighted edges (see Figure 4). Since CDGP (or EQ-CDGP) is a *decision problem*, only a feasible solution is required. Therefore, since most graph coloring problems in the literature and in real world applications are optimization problems, we define an optimization version of this basic distance geometry graph coloring problem, as shown below.

Definition 2. ***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 $x : V \rightarrow \mathbb{R}$ 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.*

Thus, in this case, a solution is the best possible feasible solution, that is, a tree of the graph G that satisfies the constraints with the minimum span.

By the other hand, instead of equalities, we can consider inequalities, such that the weight $d_{i,j}$ on an edge (i, j) is actually a lower bound for the distance to be respected between the color points $x(i)$ and $x(j)$, that is, $|x(i) - x(j)| \geq d_{ij}$. Thus, we can modify MinEQ-CDGP to deal with this scenario, which becomes the following model.

Definition 3. ***Minimum Greater than Equal Coloring Distance Geometry Problem (MinGEQ-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 $x : V \rightarrow \mathbb{R}$ such that $|x(i) - x(j)| \geq d_{i,j}$ for each $(i, j) \in E$ whose span $(\max\limits_{i \in V} x(i))$ is the minimum possible.*

MinGEQ-CDGP is equivalent to the bandwidth coloring problem (BCP) lai:2013, which itself is equivalent to the minimum span frequency assignment problem (MS-FAP) koster:1999,audhya:2011.

Figure 5 shows examples of the proposed distance coloring models.

2.1. Special cases

For the models previously stated, we can identify some specific scenarios for which additional properties can be identified. The first special case is for MinEQ-CDGP, where all distances are the same, a graph with uniform edge weights, as stated below.

Definition 4. ***Minimum Equal Coloring Distance Geometry Problem with Constant Distances (MinEQ-CDGP-Const):*** *Given a simple weighted undirected graph $G = (V, E)$, and a nonnegative integer φ, find an embedding $x : V \rightarrow \mathbb{R}$ such that $|x(i) - x(j)| = \varphi$ for each $(i, j) \in E$ whose span $(\max_{i \in V} x(i))$ is the minimum possible.*

In this model, an input graph can be stated by its sets of vertices and edges and the φ value, instead of a set of weights for each edge. A similar special case exists for MinGEQ-CDGP, as stated in the following definition.

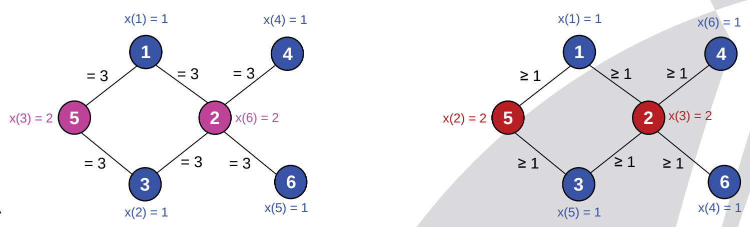
Definition 5. ***Minimum Greater than Equal Coloring Distance Geometry Problem with Constant Distances (MinGEQ-CDGP-Const):*** *Given a simple weighted undirected graph $G = (V, E)$, and a nonnegative integer φ, find an embedding $x : V \to \mathbb{R}$ such that $|x(i) - x(j)| \geq \varphi$ for each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

A summary of all distance coloring models, including special cases, is given in Table 1.

Table 1: Summary of distance coloring models.

Problem	**Constraint type**	**Distance type**
CDGP and MinEQ-CDGP	$\forall (i,j) \in E,\ \lvert x(i) - x(j)\rvert = d_{i,j}$	$\forall (i,j) \in E,\ d_{i,j} \in \mathbb{N}$
MinGEQ-CDGP	$\forall (i,j) \in E,\ \lvert x(i) - x(j)\rvert \geq d_{i,j}$	$\forall (i,j) \in E,\ d_{i,j} \in \mathbb{N}$
CDGP-Const and MinEQ-CDGP-Const	$\forall (i,j) \in E,\ \lvert x(i) - x(j)\rvert = d_{i,j}$	$\forall (i,j) \in E,\ d_{i,j} = \varphi$ $(\varphi \in \mathbb{N})$
MinGEQ-CDGP-Const	$\forall (i,j) \in E,\ \lvert x(i) - x(j)\rvert \geq d_{i,j}$	$\forall (i,j) \in E,\ d_{i,j} = \varphi$ $(\varphi \in \mathbb{N})$

When $\varphi = 1$,MinGEQ-CDGP-Const is equivalent to the classic graph coloring (Figure 6).



(a) MinEQ-CDGP-Const. (b) MinGEQ-CDGP-Const.

Figure 6: Examples of instances for the special cases of distance coloring models with constant edge weights and feasible solutions for them.

3. Algorithmic techniques and methods to solve CDGP models

In this section, we show some algorithmic strategies to solve , and discuss about some algorithmic strategies, for our distance geometry graph coloring models, considering the most general combinatorial optimization problem proposed, MinGEQ-CDGP.

3.1. Branch-and-prune methods

For solving the three models shown in Section 2, we developed two algorithms that combine the branch-and-bound methodology with concepts from constraint programming, such as pruning.

In Lavor *et al.* lavor:2012:1, a branch-and-prune (BP) algorithm was proposed for the Discretizable Molecular Distance Geometry Problem, which proceeds by enumeration possible positions for the vertices that must be located in three-dimensional space ($\mathbb{R}^3$). The position for a vertex i, where $i \in [4, n]$ and n is the number of vertices that must be placed in $\mathbb{R}^3$, is determined with respect to the last three that have already been positioned, following the ordering and sphere intersection cited in Section 2. However, a distance between the currently positioned vertex and a previous one that was placed before the last three can be violated, which requires feasibility tests to guarantee that the solution being built is valid. The authors used the Direct Distance Feasibility (DDF) pruning test, where $\forall (i, j) \in E\ \ \lvert\lvert\lvert x(i) - x(j)\rvert\rvert - d_{i,j}\rvert < \epsilon$, where ϵ is a given tolerance.

In our work, we have transported these concepts to the studied distance coloring problems. One of the first observations that can be made is that there is no explicit vertex ordering to

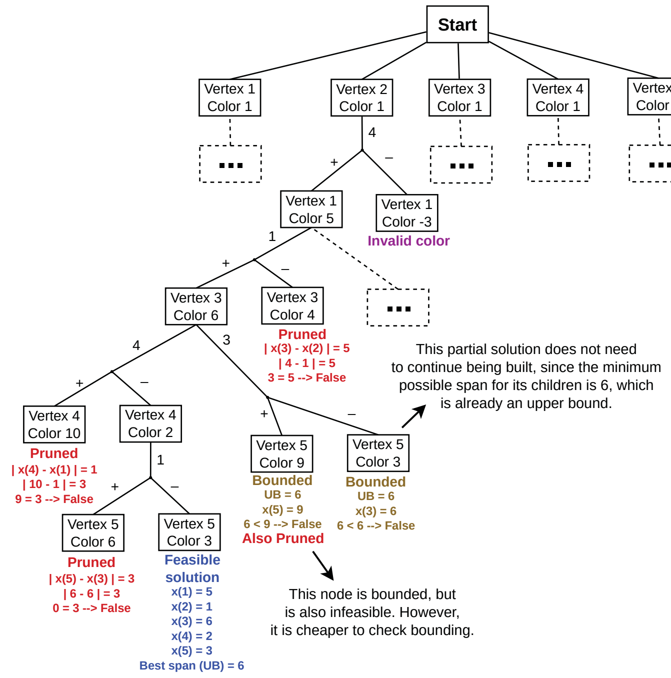


Figure 7: Partial enumeration of solutions starting from vertex 2 for the MinGEQ-CDGP instance defined by Figures 1 and 2 using the first BPB algorithm.

be considered, so we will build the ordering by enumerating. Using the branch-and-bound method, we can obtain partial solutions (sequences of vertices that have already been colored) that cannot improve on the current best solution. Two algorithms were developed and are described below.

A - First version: using previous colored neighbors

The first algorithm, denoted by BPB-Prev, colors a vertex j by calculating the lowest feasible color with respect to the last colored vertex i, where, in most cases, it is a neighbor of j (if all neighbors of i are already colored, then no previous vertex is taken into account. It is a similar to how the BP algorithm from works for the DMDGP lavor:2012:1.

BPB-Prev works as follows. First, a vertex i whose demands have not been satisfied yet is selected as a starting point. This vertex receives the color 1, which is the lowest available (since all colors are positive integers). Then a neighbor j of i that has not been colored yet is selected. There are two color possibilities for j that satisfy the constraint $|x(i) - x(j)| = d_{i,j}$. The first one is $x(i) + d_{i,j}$ and the second one is $x(i) - d_{i,j}$ (which will only be used if $x(i) > d_{i,j}$. This method is recursively applied to the neighbors of j, that is, j becomes i. If, when selecting a neighbor, all possible vertices are colored, another non-colored vertex among all currently present in the graph is selected and receives color 1, where the recursive procedure is restarted. When all vertices are colored, we have a feasible solution for the distance coloring problem considered. In order to guarantee that all solutions that can be reached using this procedure are explored, each vertex is considered as a starting point for the algorithm. Pseudocode for this procedure is given in Algorithm 1, and an example with the graph from Figure 1 is given in Figure 7.

In early tests with this algorithm, we noticed that some results obtained were not optimal for the considered problems. This characteristic is also present in the BP for DMDGP, since its authors do not consider it an exact algorithm.

B - Second version: check feasibility only of full colorings

In BPB-Prev, at each node in the enumeration tree, a feasibility check is made to ensure that the color assigned to the current vertex does not violate the distance to each neighbor. Since this can be costly, a modification can be done to the algorithm, which will result in a second version denoted by BPB-CheckFull, in order to make feasibility checks only at leaf nodes of the enumeration

Algorithm 1 Branch-prune-and-bound, first version: BPB-Prev

Require: graph G (with set V of vertices and set E of edges), matrix d of distances for each edge, binary edge function $f : E \to \{0,1\}$, current vertex i to be colored, previous vertex j, current coloring x, best coloring found x_{best}, upper bound ub and enumeration tree depth dpt.

```
1: function BPB-PREV($G = (V,E), d, f, i, j, x, x_{best}, ub$)
2:     $numTestedColors = 0$
3:     while $numTestedColors < 2$ do
4:         if $j = -1$ then
5:             $color \leftarrow 1$
6:             $numTestedColors = 2$
7:         else
8:             if $numTestedColors = 0$ then
9:                 $color \leftarrow x(j) + d_{i,j}$
10:            else
11:                $color \leftarrow x(j) - d_{i,j}$
12:                if $color < 1$ then continue          ▷ Invalid color, can be discarded
13:                $numTestedColors = numTestedColors + 1$
14:            $x(i) \leftarrow color$
15:            if $\max_{v \in V \mid v \text{ is colored}} x(v) \geq ub$ then
16:                Remove color from $i$
17:                return          ▷ Discard partial solution by bounding
18:            if FEASIBILITYTEST($G, d, f, x, i$) = false then
19:                Remove color from $i$
20:                return          ▷ Distance violation, discard partial solution by pruning
21:            if $dpt = |V|$ then          ▷ If true, then all vertices are colored
22:                if $\max_{v \in V} x(v) < \max_{v \in V} x_{best}(v)$ then
23:                    $x_{best} \leftarrow x$
24:            else
25:                $hasNeighbor \leftarrow$ false
26:                for each neighbor $k$ of $i$ do
27:                    if $k$ is not colored then
28:                        $hasNeighbor \leftarrow$ true
29:                        BPB-PREV($G, d, f, k, i, x, x_{best}, ub, dpt + 1$)
30:                if $hasNeighbor =$ false then
31:                    for each vertex $k$ of $G$ do
32:                        if $k$ is not colored then
33:                            BPB-PREV($G, d, f, k, -1, x, x_{best}, ub, dpt + 1$)
34:            Remove color from $i$
35:    return $x_{best}$
```

tree, that is, only when all vertices are colored.

Two important differences are present in BPB-CheckFull. First, to ensure that a solution is feasible before possibly accepting it, the feasibility check must traverse the entire graph instead of only the neighbors of the current vertex as in BPB-Prev. Secondly, whenever an unfeasibility is met, the enumeration tree must be backtracked to the point where the violation has ocurred and not to the immediately father node of the considered leaf node, since all nodes below the point of unfeasibility contain it as a part of the solution and can safely be discarded. Let i and j be two vertices for which there is an equality constraint $|x(i) - x(j)| = d_{i,j}$ such that the assigned colors $x(i)$ and $x(j)$ violate it (for inequalities, the operator $=$ is changed to $\leq$). Then the algorithm must backtrack through the path it descending in the enumeration tree until either i or j is found. Since not all distances are taken into account at the time of choosing a color for a vertex, this version of BPB is also not exact. Pseudocode is omitted.

C - Third version: calculating the color using all neighbors

In order to find the optimal solutions for the coloring problems, we developed another algorithm, denoted by BPB-Select, which uses the same concepts of bounding and pruning, but in

Algorithm 2 Color selection on BPB-Select

```
Require: graph $G$ (with set $V$ of vertices, set $E$ of edges and matrix $d$ of distances for each edge), vertex $i$
    to be colored, current coloring $x$, upper bound $ub$ and binary edge function $f : E \to \{0, 1\}$
 1: function SELECTCOLOR($G = (V, E, d), i, x, ub$)
 2:     $sumColors \leftarrow$ array of $ub$ elements (all initialized to zero)
 3:     $numColoredNeighbors \leftarrow 0$
 4:     for each neighbor $j$ of $i$ do
 5:         if $j$ is colored then
 6:             $numColoredNeighbors \leftarrow numColoredNeighbors + 1$
 7:             $resultFirstIneq \leftarrow x(j) - d_{i,j}$
 8:             if $resultFirstIneq \geq 1$ then
 9:                 if $f((i, j)) = 0$ then                                  ▷ Inequality constraint
10:                     for $k \leftarrow 1$ to $resultFirstIneq$ do
11:                         $sumColors[k] \leftarrow sumColors[k] + 1$
12:                 else                                                     ▷ Equality constraint
13:                     $sumColors[resultFirstIneq] \leftarrow sumColors[resultFirstIneq] + 1$
14:             $resultSecondIneq \leftarrow x(j) + d_{ij}$
15:             if $resultSecondIneq \leq ub$ then
16:                 if $f((i, j)) = 0$ then                                  ▷ Inequality constraint
17:                     for $k \leftarrow resultSecondIneq$ to $ub$ do
18:                         $sumColors[k] \leftarrow sumColors[k] + 1$
19:                 else                                                     ▷ Equality constraint
20:                     $sumColors[resultSecondIneq] \leftarrow sumColors[resultSecondIneq] + 1$
21:     for $k \leftarrow resultSecondIneq$ to $ub$ do
22:         if $sumColors[k] = numColoredNeighbors$ then
23:             return $k$
24:     return -1                                          ▷ Partial solution can by discarded
```

a different way.

Instead of using information from the previous colored vertex to determine the color for the current vertex, a system of absolute value inequalities (or equalities, if the constraints are of this type) is used to calculate such color. Those inequalities arise from the distance constraints for the edges. Let j be the vertex that must be colored. The color $x(j)$ must be the solution of a system of absolute value inequalities where there is one for each colored neighbor i and each one of them is as follows:

$$|x(j) - x(i)| \begin{cases} = d_{i,j} & \text{if the constraint is an equality } (f((i,j)) = 1) \\ \geq d_{i,j} & \text{if the constraint is an inequality } (f((i,j)) = 0) \end{cases}$$

To solve the system, we use an algorithm described as follows. An array $sumColors$ of ub elements, where ub is an upper bound for the coloring span, is created with all elements initialized to zero. Then, for each colored neighbor i of j, we have to decompose the corresponding inequality into two sub-inequalities: $|x(j) - x(i)| \geq d_{i,j}$ and $|x(j) - x(i)| \leq d_{i,j}$ (if the constraints are equalities, then the symbols $\geq$ and $\leq$ must be substituted by $=$). The solution for the inequality is given, then, by $x(i) + d_{i,j} \leq x(j) \leq x(i) - d_{i,j}$ (if the constraints are equalities, then we have $x(j) = x(i) + d_{i,j}$ or $x(j) = x(i) - d_{i,j}$). The next step is increasing by one all elements of $sumColors$ whose indices satisfy the solutions for the system. The color that will be assigned to j is the lowest index whose value in $sumColors$ is equal to the number of colored neighbors of j.

For BPB-Select, lines 6 to 17 must be substituted by a call to a function SELECTCOLOR(), whose returned value is stored in variable $color$. Pseudocode for the function is given in 2.

Feasibility test

When building a partial solution, we must verify if it is feasible when not all distances are taken into account at the same time, especially on BPB-Prev. We used a similar feasibility test to the Direct Distance Feasibility (DDF) used on the BP algorithm for the DMDGP.

Algorithm 3 Direct Distance Feasibility (DDF) check

```
Require: graph $G$ (with set $V$ of vertices and set $E$ of edges), matrix $d$ of distances for each edge, binary
    edge function $f : E \to \{0, 1\}$, current coloring $x$ and vertex $i$.
 1: function SELECTCOLOR($G, d, f, x, i$)
 2:     for each neighbor $k$ of $i$ do
 3:         if $k$ is colored then
 4:             if $f((i, j)) = 0$ then                              ▷ Inequality constraint
 5:                 if $|x(k) - x(i)| > d_{i,j}$ then
 6:                     return false
 7:             else                                                 ▷ Equality constraint
 8:                 if $|x(k) - x(i)| \neq d_{i,j}$ then
 9:                     return false
10:     return true
```

Let i be the vertex that has just been colored. Then we must check, for each neighbor j that has already been colored, if the condition $|x(i) - x(j)| \geq d_{i,j}$ (if $f((i, j)) = 0$) or $|x(i) - x(j)| = d_{i,j}$ (if $f((i, j)) = 1$). This test can be seen as a variation of DDF setting ϵ to zero and allowing inequalities in the test. Pseudocode for DDF is given in Algorithm 3.

4. Computational experiments

The constraint and integer programming formulations (MinGEQ-CDGP-CP and MinGEQ-CDGP-IP) were implemented in C++ using IBM/ILOG CPLEX solver 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), 112GB 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 limites to 48 hours of CPU time (172800 seconds). For the IP formulation, we also made experiments where all cutting planes generated by CPLEX were disabled, resulting in a branch-and-bound method.

We used instances from the literature and generated new random ones. Since some of the distance coloring models are equivalent to coloring problems present in the literature, specifically, MinGEQ-CDGP being equivalent to BCP (and MS-FAP), instances for these problems were explored. The first set of literature instances is known as GEOM, generated by trick:2002 for BCP and its multicoloring variant, and consists of 33 instances of three types: GEOM*n* are sparse graphs, while GEOM*n*a and GEOM*n*b are denser graphs (where n is the number of vertices in the instance). Since we do not consider multicoloring in the models, we used only the pure BCP instances. Other sets of instances are the classical Philadelphia (21 vertices) and Helsinki (25 vertices) problems for MS-FAP, based on cellular networks from the cities of the same names, and an artificial problem (55 vertices) that extends a Philadelphia instance chakraborty:2001,dias:2013,kendall:2005. Since these instances also use multicoloring, we applied the suggestion by the authors of the GEOM instances in these as well. The last set of instances are randomly generated graphs with $|V|$ = 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18 and 20 and where the number of edges is a random number between $|V| - 1$ and $\frac{|V|(|V| - 1)}{2}$, which ensures that the graph is always connected and the number of edges does not exceed the one from a complete graph. Edges are then randomly generated with weights ranging from 1 to 6 dias:2014.

Table 4 shows the results for the GEOM instances. Since the BCP variants are also used in the literature, we compared our results with the Discropt heuristic framework by phan:2002 and the multistart iterated tabu search heuristic by lai:2013 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 for all of them. However, we emphasize that, for some instances, neither method achieved the best solution presented by phan:2002. As noted by lai:2013, no other work has obtained the same results, while our exact approaches reached the same best solutions for these instances obtained by other authors (Table 4).

Table 2: Results for the constraint and integer programming formulations applied on literature BCP instances proposed by Michael Trick (GEOM set).

Instance	Num. Vert.	Phan and Skiena (2002)	lai:2013	MxCDGP-CP (CP Optimizer)		MxCDGP-IP (CPLEX)			
						B&C		B&B	
		Best Reported	Best Reported	Best Found	Time (sec)	Best Found	Time (sec)	Best Found	Time (sec)
GEOM20		20[†]	21	21	0.00	21	0.97	21	2.08
GEOM20a	20	20	20	20	0.00	20	3.35	20	6.33
GEOM20b		13	13	13	0.00	13	1.31	13	2.56
GEOM30		27[†]	28	28	0.05	28	7.85	28	39.76
GEOM30a	30	27	27	27	0.06	27	65.48	27	213.82
GEOM30b		26	26	26	0.23	26	22.31	26	95.99
GEOM40		27[†]	28	28	0.23	28	54.74	28	141.89
GEOM40a	40	38	37	37	2.53	37	901.93	37	2085.24
GEOM40b		36	33	33	3.08	33	693.22	33	2077.01
GEOM50		29	28	28	0.26	28	112.59	28	797.82
GEOM50a	50	54	50	50	378.55	50	9079.44	≤ 50	172832.85
GEOM50b		40	35	35	377.52	35	15117.44	≤ 35	172878.43
GEOM60		34	33	33	0.29	33	540.03	33	72711.57
GEOM60a	60	54	50	50	614.81	50	72707.10	50	148030.40
GEOM60b		47	41	41	76850.68	≤ 41	172800.07	≤ 44	172855.01
GEOM70		40	38	38	3.03	38	2504.25	≤ 38	172889.47
GEOM70a	70	64	61	61	33583.75	≤ 63	172800.26	≤ 563	173207.41
GEOM70b		54	47	47	534.65	≤ 63	172800.40	≤ 127	172960.59
GEOM80		44	41	41	8.18	41	7069.52	41	32435.96
GEOM80a	80	69	63	63	87770.77	≤ 1731	172803.55	≤ 1283	173045.09
GEOM80b		70	60	60	54320.89	≤ 81	172800.25	≤ 2052	173113.15
GEOM90		48	46	46	55.18	≤ 46	172834.46	≤ 288	172920.76
GEOM90a	90	74	63	63	130050.12	None	173100.57	≤ 4191	173034.18
GEOM90b		83	69	≤ 70	172800.00	None	172802.83	≤ 4450	173027.02
GEOM100		55	50	50	545.79	50	99218.94	≤ 700	173070.60
GEOM100a	100	84	67	≤ 70	172800.01	None	172826.84	None	172895.87
GEOM100b		87	72	≤ 71	172800.02	None	172838.38	None	172888.12
GEOM110		59	50	50	2982.24	None	172800.62	≤ 111	173264.16
GEOM110a	110	88	72	≤ 73	172800.01	None	172917.07	None	173078.27
GEOM110b		87	78	≤ 79	172800.01	None	173008.89	None	173153.95
GEOM120		67	59	59	147572.50	None	173296.11	≤ 4051	173102.15
GEOM120a	120	101	82	≤ 84	172800.01	None	173181.91	None	173109.76
GEOM120b		103	84	≤ 85	172800.01	None	173187.16	None	173112.29

[†]Results lower than the obtained optimum - possibly wrong in the corresponding work.

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 and new random ones 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. However,

Table 3: Results for the constraint and integer programming formulations applied on literature MS-FAP instances (Philadelphia, Helsinki and Artificial) without multicoloring demands.

Instance	Num. Vert.	Set	MxCDGP-CP (CP Optimizer)		MxCDGP-IP (CPLEX)			
					B&C		B&B	
			Best Found	Time (sec)	Best Found	Time (sec)	Best Found	Time (sec)
C21-1	21	Philadelphia	7	0.40	7	0.87	7	2.62
C21-2			9	0.06	9	2.66	9	4.67
C21-3			7	0.40	7	0.81	7	2.62
C21-4			9	0.06	9	2.49	9	4.68
C21-5			7	0.40	7	0.84	7	2.62
C21-6			9	0.06	9	2.47	9	4.67
C21-7			7	0.40	7	0.83	7	2.63
C21-8			9	0.06	9	2.62	9	4.67
C25-1	25	Helsinki	8	4.71	8	1.90	8	10.28
C55-1	55	Artificial	7	0.79	7	30.63	7	173.34

both are still exponential and cannot solve big instances by themselves.

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.

References

Akihiro, U., Hiro, I., Hideyuki, U., e Mitsuo, Y. (2002). A Coloring Problem With Restrictions of Adjacent Colors. *International Transactions in Operational Research*, 9(2):183–194.

Audhya, G., Sinha, K., Ghosh, S., e Sinha, B. (2011). A survey on the channel assignment problem in wireless networks. *Wireless Communications and Mobile Computing*, 11:583–609.

Bondy, J. e Murty, U. (2008). *Graph Theory: Graduate Texts in Mathematics*. Springer, 3rd edition.

Chakraborty, G. (2001). An Efficient Heuristic Algorithm for Channel Assignment Problem in Cellular Radio Networks. *IEEE Transactions on Vehicular Technology*, 50(6):1528–1539.

Dias, B. (2014). Theoretical models and algorithms for optimizing channel assignment in wireless mobile networks. Master's thesis, Federal University of Amazonas, Brazil. In portuguese.

Dias, B., Freitas, R., e Maculan, N. (2012). Alocaï¿½ï¿½o de canais em redes celulares sem fio: algoritmos e modelos teï¿½ricos em grafos e escalonamento. In *Proceedings of the XVI Latin-Ibero-American Conference on Operations Research / XLIV Brazilian Symposium of Operations Research*. Brazilian Society of Operations Research (SOBRAPO). In portuguese.

Dias, B., Freitas, R., e Maculan, N. (2013a). *Simulated annealing* para a alocaï¿½ï¿½o de canais em redes mï¿½veis celulares. In *Anais do XLV Simpï¿½sio Brasileiro de Pesquisa Operacional*. Sociedade Brasileira de Pesquisa Operacional (SOBRAPO). In portuguese.

Dias, B., Freitas, R., e Szwarcfiter, J. (2013b). On graph coloring problems with distance constraints. In *Proceedings of I Workshop on Distance Geometry and Applications (DGA 2013)*.

Hale, W. (1980). Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 25:1497–1514.

Karp, R. (1972). Reducibility Among Combinatorial Problems. In Miller, R. E. e Thatcher, J. W., editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

Kendall, G. e Mohamad, M. (2005). Solving the Fixed Channel Assignment Problem in Cellular Communications Using an Adaptive Local Search. In Burke, E. e Trick, M., editors, *5th International Conference for the Practice and Theory of Automated Timetabling (PATAT 2004)*, Lecture Notes in Computer Science, vol. 3616. Springer, Heidelberg.

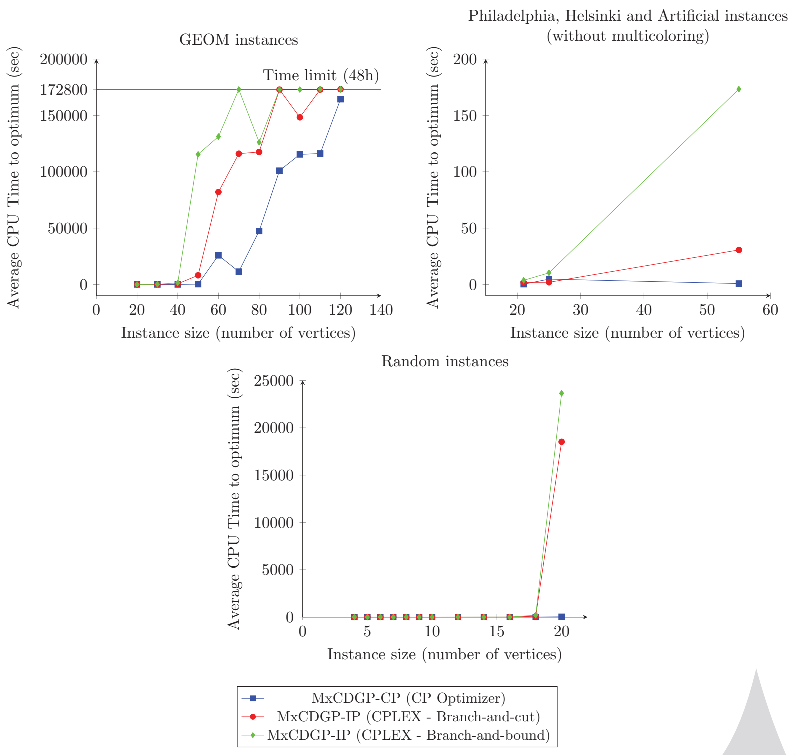


Figure 8: Number of vertices × CPU time needed to prove optimality (if found) for each method.

Koster, A. (1999). *Frequency assignment: models and algorithms*. PhD thesis, Universiteit Maastricht, the Netherlands.

Koster, A. e Munhoz, X. (2010). *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks*. Texts in Theoretical Computer Science. Springer.

Lai, X. e Lu, Z. (2013). Multistart Iterated Tabu Search for Bandwidth Coloring Problem. *Computers & Operations Research*, 40:1401–1409.

Lavor, C., Liberti, L., Maculan, N., e Mucherino, A. (2012a). The discretizable molecular distance geometry problem. *European Journal of Operational Research*, 52(1):115–146.

Lavor, C., Liberti, L., Maculan, N., e Mucherino, A. (2012b). Recent advances on the discretizable molecular distance geometry problem. *Computational Optimization and Applications*, 219(3):698–706.

Malaguti, E. e Toth, P. (2010). A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1–34.

Phan, V. e Skiena, S. (2002). Coloring graphs with a general heuristic search engine. In *Computational Symposium on Graph Coloring and Its Generalizations*, pages 92–99.

Trick, M., Mehrotra, A., e Johnson, D. (2002). COLOR02/03/04: Graph Coloring and its Generalizations. `http://mat.gsia.cmu.edu/COLOR02/`.