A generalization of the graph multicoloring problem [1]

Isabel Méndez-Díaz* Paula Zabala**

*Departamento de Computación, FCEyN, Universidad de Buenos Aires
Pab. I, Ciudad Universitaria, Buenos Aires, Argentina

*Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

Abstract

This paper focuses on a generalization of the graph multicoloring problem. We propose an approach based on integer programming formulations that avoid to certain extent, symmetric solutions. A Branch-and-Cut algorithm based on these formulations is proposed. We report on computational experience with our models on random instances.

Keywords: *Graph Multicoloring, Integer Programming, Branch-and-Cut*

1 Introduction

The problems related to frequency assignment to transmitters can be naturally modeled by generalizations of graph coloring problems. The vertices of a graph represent the transmitters, and an edge between any pair of vertices means that they have the potential to interfere. The frequencies correspond to colors assigned to the vertices such that adjacent vertices must receive different colors. In the classical frequency assignment problem, frequencies are assigned to transmitters with the objective of either minimizing the number of used frequencies, or the total interference.

These models do not consider up-to day technologies like frequency-hopping. Frequency-hopping spread spectrum (FHSS) is a method of transmitting radio signals by rapidly switching a carrier among frequency channels, using a pseudorandom sequence known to both transmitter and receiver. One of the advantages of this method is that transmissions can share a frequency band with many types of conventional transmissions with minimal interference. As a result, bandwidth can be used more efficiently. If we assume that the transmission occurs only on a small portion of a bandwidth at any given time, the effective interference bandwidth is almost the same.

However, with the aim of having some control of possible interferences, we are interested in assigning lists of frequencies to each transmitter such that the cardinal of the intersection between the frequency lists of adjacent vertices should not exceed a maximum threshold.

In this work we propose a generalization of the graph multicoloring problem that is able to model this situation. The graph multicoloring problem is, in turn, a generalization of the well known graph coloring problem. Given a graph, the multicoloring problem is to assign a preset number of colors to each vertex such that no two adjacent vertices may have any colors in common. The objective is to make this using the fewest possible number of colors.

More formally, consider an undirected graph $G = (V, E)$, R a set of colors, $ncol = |R|$ and k_v the demanded number of colors of vertex v. A multicoloring of G is a mapping $f : V \to 2^R$ such that $|f(v)| = k_v$ and $|f(u) \cap f(v)| = 0$ for all $(uv) \in E$. The graph multicoloring problem is to find a multicoloring using as few colors as possible.

In this work we deal with a generalization of the multicoloring problem by relaxing the condition that no two adjacent vertices may share a common color. For all $(uv) \in E$, we consider c_{uv} the maximum

[1]This research was partially supported by UBACYT Grant X143, PICT 920, PICT 1600 and PICT Raices 01070

number of colors that can be shared by adjacent vertices u, v. We define a *relaxed multicoloring* of G as a mapping $f : V \to 2^R$ such that $|f(v)| = k_v$ and $|f(u) \cap f(v)| \leq c_{uv}$ for all $(uv) \in E$. A *p-relaxed multicoloring* is a *relaxed multicoloring* that uses p colors. The *relaxed graph multicoloring problem* is to find a *relaxed multicoloring* using as few colors as possible.

The relaxed graph multicoloring problem is NP-Hard because the coloring problem (which is a special case) is a known NP-Hard problem.

Like most optimization problems on graphs, the relaxed multicoloring graph problem can be formulated as a linear integer programming problem. Approaches in this way were proposed for the coloring problem [1, 5, 8, 10, 11], multicoloring problem [12] and list coloring [2].

Since colors are indistinguishable, many symmetric colorings typically exist for the same given number of colors. Taking this into account, we propose an approach based on integer programming formulations that reduces the number of symmetric feasible solutions in order to have a more tractable computational model.

LP-based Branch-and-Cut algorithms are currently the most important tool to deal with linear integer programming problems computationally. The basic structure is a Branch-and-Bound algorithm which may call a cutting plane algorithm at each node of the search tree. Therefore, the main elements of a Branch-and-Cut algorithm are the formulation and the separation procedures, which we will describe in the next sections. We also include some additional features, such as heuristics based on rounding LP-solutions.

The structure of the remainder of the paper is as follows. In the next section, we give an integer programming formulation and present some valid inequalities. In Section 3 we describe the ingredients of our Branch-and-Cut algorithm. Computational results are given in Section 4. The paper closes with final remarks.

2 Integer programming formulation and valid inequalities

We consider binary variables x_{vj}, with $v \in V$ and $j \in R$, where $x_{vj} = 1$ if color j is assigned to vertex v and $x_{vj} = 0$ otherwise. We also define $ncol$ binary variables w_j ($1 \leq j \leq ncol$), that indicate whether color j is used in some vertex, i.e. $w_j = 1$ if $x_{vj} = 1$ for some vertex v. Finally, binary variables y_{uvj}, with $(uv) \in E$, $j \in R$, where $y_{uvj} = 1$ if color j is assigned to both endpoints u, v. The relaxed graph multicoloring problem can be formulated as:

$$\text{Min} \sum_{j=1}^{ncol} w_j$$

subject to

$$\sum_{j \in R} x_{vj} = k_v \quad \text{for all } v \in V \tag{1}$$

$$\sum_{j \in R} y_{uvj} \leq c_{uv} \quad \text{for all } (uv) \in E \tag{2}$$

$$x_{uj} + x_{vj} - y_{uvj} \leq 1 \quad \text{for all } (uv) \in E, j \in R \tag{3}$$

$$x_{vj} \leq w_j \quad \text{for all } v \in V, j \in R \tag{4}$$

$$x_{vj} \in \{0,1\} \quad \text{for all } v \in V, j \in R$$

$$y_{uvj} \in \{0,1\} \quad \text{for all } (uv) \in E, j \in R$$

$$w_j \in \{0,1\} \quad \text{for all } j \in R$$

Constraints (1) assert that each vertex v must receive exactly k_v colors, and constraints (2) say that every pair of adjacent vertices u, v may not share more than c_{uv} colors. Constraints (3) force $y_{uvj} = 1$ if color j is shared by the adjacent vertices u, v. If color j is assigned to some vertex, constraints (4) impose $w_j = 1$.

This formulation admits symmetric solutions since all color permutations yield feasible solutions with the same objective function value. The indistinguishability of the variables makes very difficult to use the model in practice except small instances.

Following the approach we proposed in [9] for the classical graph coloring problem, we consider two ways to remove symmetry in order to have a more tractable computational model.

In Model 1, we add the constraints $w_j \geq w_{j+1}$ for all $1 \leq j \leq ncol - 1$ to remove symmetry. In this way, color j can be assigned to a vertex provided color $j-1$ has already been assigned. Any *p-relaxed multicoloring* is stood for a feasible solution using the first p colors. All symmetric *p-relaxed multicoloring* using colors with label greater than p are eliminated by these constraints. There are still equivalent solutions arising from permutations of the first p colors. In Model 2, we eliminate some of these solutions by adding the constraints $\sum_{v \in V} x_{vj} \geq \sum_{v \in V} x_{v(j+1)}$ for all $1 \leq j \leq ncol - 1$. These constraints impose that the number of vertices colored by j must be greater than or equal to the number of vertices colored by $j+1$.

Our main focus is on developing a Branch-and-Cut algorithm that takes advantage of the particular structure and exploits the properties of the problem. Although Model 2 is more strengthened than Model 1, it is not clear that a Branch and Cut algorithm will have a better performance on Model 2. To compare and to evaluate the models, we experiment and we analize computational results.

From now on, we restrict our attention to the particular case where $k_v = k$ for all $v \in V$ and $c_{uv} = c$ for all $(uv) \in E$. The complexity of the general case makes very difficult to study the polytope. Even though this restriction we find the problem interesting on its own and it is still hard.

Since our goal is to develop a Branch and Cut algorithm, we look for valid inequalities to strength the formulations. As a result of our polyhedral investigations, the inequalities that follow are proven to be valid for the polyhedra. The cuts are based on the idea of how many vertices of a subset $V' \subset V$ with a particular structure, such as cliques, can share $c+1$ colors. This idea leads to the following inequalities.

Proposition 1. *Let $C = (V', E')$ be a clique of size $p \geq 2$ of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$\sum_{i=1}^{c+1} \sum_{v \in V'} x_{vj_i} \leq p \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$$

is a valid inequality.

Proof. The proof is performed by induction on p.

The case $p = 2$ is trivial.

Assume that this is a valid inequality for $2 \leq p \leq s$, we have to prove that it is also valid for $p = s + 1$.

Let $C = (V', E')$, $|V'| = s + 1$ and $u \in V'$. We have to show that

$$\sum_{i=1}^{c+1} \sum_{v \in V'} x_{vj_i} = \sum_{i=1}^{c+1} \sum_{v \in V' \setminus \{u\}} x_{vj_i} + \sum_{i=1}^{c+1} x_{uj_i} \leq s \sum_{i=1}^{c} w_{j_i} + \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}.$$

If $\sum_{i=1}^{c+1} x_{uj_i} \leq c$, then $\sum_{i=1}^{c+1} x_{uj_i} \leq \sum_{i=1}^{c} w_{j_i}$ and by the inductive hyphotesis $\sum_{i=1}^{c+1} \sum_{v \in V' \setminus u} x_{vj_i} \leq s \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$. The result follows.

If $\sum_{i=1}^{c+1} x_{uj_i} = c+1$, then $\sum_{i=1}^{c+1} x_{uj_i} \leq \sum_{i=1}^{c} w_{j_i} + w_{j_{c+1}}$. Since $uv \in E' \ \forall v \in V' \setminus \{u\}$, this also implies that $\sum_{i=1}^{c+1} x_{vj_i} \leq c = \sum_{i=1}^{c} w_{j_i}$. We conclude that the inequality is valid.

□

Due to the space limitation, we omit the proofs of the following valid inequalities.

Proposition 2. *Let $C = (V', E')$ be a clique of size $p \geq 2$ of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$p \sum_{v \in V'} x_{vj_1} - \sum_{uv \in E'} y_{uvj_1} + \sum_{i=2}^{c+1} \sum_{uv \in E'} y_{uvj_i} \leq (\frac{cp(p-1)}{2} + p) w_{j_1}$$

is a valid inequality.

Proposition 3. *Let $C = (V', E')$ be a clique of size p of G and $j_1 < j_2 < \cdots < j_{c+1} \in R$, then*

$$p \sum_{v \in V'} x_{vj_{c+1}} - \sum_{uv \in E'} y_{uvj_{c+1}} + \sum_{i=1}^{c} \sum_{uv \in E'} y_{uvj_i} \leq \frac{p(p-1)}{2} \sum_{i=1}^{c} w_{j_i} + p w_{j_{c+1}}$$

The next inequality is a transitive consequence of using color j_0 among three adjacent vertices.

Proposition 4. *Let $\{u_1, u_2, u_3\}$ be adjacent vertices and $j_0 \in R$. Then*

$$y_{u_1 u_2 j_0} + y_{u_2 u_3 j_0} \leq x_{u_2 j_0} + y_{u_1 u_3 j_0}$$

is a valid inequality.

3 Branch-and-Cut Algorithm

Given an integer programming problem, the idea of a Branch-and-Cut is recursively to partition the solution set into subsets and solving the problem over each subset. This procedure generates an enumeration tree where offsprings of a node correspond to the partition of the set associated with the parent node. In each node of the tree, a linear relaxation of the problem is considered by dropping integrality requirements and adding valid inequalities which cut off the fractional solution.

To reduce the number of nodes of the tree, it is important to have good lower and upper bounds, good rules to partition the feasible set, good strategies to search on the tree and a good strengthening of the linear relaxations.

In the following we describe the different aspects we consider in our implementation.

Initial Upper Bound

An upper bound is obtained with a greedy heuristic, similar to DSATUR for the classical graph coloring [3], which is run once at the beginning of the algorithm.

The sequential procedure starts with an ordered color list $L = \{1, \ldots, k\}$. In every step, a vertex is chosen and it is assigned the first k colors from the list L such that they are compatible with its already colored neighbors. If there are not enough colors in L, new colors are added to it. After the assignment, the used colors are moved to the end of the list and the procedure goes to the next step. The procedure runs two times by considering different vertex orderings to perform the assignment:

- The highest number of colors used by its neighbors. Ties break by the highest number of non-colored neighbors.
- The highest number of colors used by its neighbors. Ties break by the lowest number of colored neighbors.

Initial Lower Bound

The lower bound is initialized with a simple heuristic to find a clique K, as large as possible. Then, we apply a complete enumeration procedure to find an optimal relaxed multicoloring for K.

We do not preassign colors to the vertices of K, as it is usually done on classic coloring, because it could be a no feasible assignment for an optimal relaxed multicoloring of G.

The enumeration of feasible solutions for instances associated with cliques up to 10 vertices is very fast to carry out. However, it takes a prohibitive amount of time for greater cliques if we consider that it is an initial phase of a Branch-and-Cut algorithm. Then, we impose this limit to the size of the clique.

Primal Heuristic

The availability of good feasible solutions may reduce the size of the Branch-and-Cut tree significantly. During the overall procedure we solve many linear problems and obtain fractional solutions. Every time we have a fractional solution we try to get an integer solution from it by using a rounding procedure. A simple idea consists in choosing a vertex v and ordering the fractional variables x_{vj} in a nonincreasing order of values. We take the first variable on the list and round it up to 1 if the corresponding color is feasible and repeat the step while there are still colors to be assigned to v. The criteria we mentioned for the initial heuristic are used for choosing the vertices.

Cutting plane generation

In order to make the inequality separation, we developed a heuristic procedure for the first three inequalities.

For each color k_1, we consider the list of non-zero variables in decreasing order of the $x^*_{vk_1}$ values, where x^* denotes the current fractional solution.

We initialize a clique with vertex v for each $x^*_{vk_1}$ in the list. For each v, we do several trials bounded by an input parameter. In trial j, we choose the fractional variable $x_{v'k_1}$ such that vertex v' is the *j-th* adjacent vertex to v in the list. We add this vertex to the clique. Then, the clique is grown in size trying to add other adjacent vertices following the ordered list.

Once this is done, it remains to determine the other colors. Since it does not have an expensive computational cost, we attempt all possibilities to find violated inequalities which are added to the LP relaxation.

The last inequalities are handled by brute-force.

4 Computational Experiments

We report in this section on computational experiences with our Branch-and-Cut algorithm. The code was implemented in C++ using the CPLEX 10.1 LP solver.

We have performed the experiments on a SUN UltraSparc III workstation with a CPU running at 1GHz and 2GB of RAM memory. CPU times are reported in seconds. We impose a CPU time limit of 1 hour. The algorithm is tested on random graphs $G(n, p)$ of 20 and 30 vertices and an edge between each pair of vertices with independent probability p. We use random graphs with low ($p \leq 30\%$), medium ($40\% \leq p \leq 66\%$) and high ($p \geq 70\%$) density.

Experiments were carried out over 30 instances per density considered and three different k values. For each model, we report the average CPU time and the average number of tree nodes explored. When the average is taken over less than 30 instances because the others could not be solved within the time limits, we indicate the number of instances solved in brackets. If the time limit was exceeded for all the instances, we report the average final gap. For the root node of the enumeration tree 10 cutting plane rounds were implemented. For tree nodes other than the root, a limit of 2 cutting plane rounds were considered.

As it can be appreciated from the results reported in Table 1, the Branch and Cut algorithm based on Model 2 was clearly dominated by the Branch and Cut algorithm based on Model 1. This conclusion applies despite the fact that Model 1 has more symmetric solutions than Model 2.

To appreciate the benefits of using our cutting planes and other specific aspects we have implemented, we compare our Branch-and-Cut algorithm with the general purpose IP-solver CPLEX based on Model 1. Tests were performed using all the advantages of preprocessing, cutting, etc. that CPLEX offers. Our strategies attains a better performance than CPLEX. CPLEX could not solve instances that could be managed by our Branch-and-Cut within the CPU time limit imposed.

In many cases, our algorithms find the optimal solution very early in the enumeration process but CPLEX requires too much time to obtain the optimality certification. However, our cutting planes improve the lower bounds and this improvement allows to fathom a large percentage of nodes earlier in the tree. The use of cutting planes seems to be essential for solving most of the instances.

We can see from the table that the results do not show a clear pattern. Some instances could be solved to proven optimality in a few CPU seconds while others could not be solved within the time limit imposed. No obvious explanation seems to exist for that behavior.

For any density, instances with $c = k - 1$ are the easiest to solve and we can tackle instances with 30 vertices which are reported in Table 2.

5 Final Remarks

We deal with a generalization of the graph multicoloring problem. We propose two integer programming approaches and a Branch-and-Cut algorithm. We have characterized several valid inequalities. The usefulness of these inequalities has been proved through computational experiments.

k-c		Low		Medium		High	
		Time	Nodes	Time	Nodes	Time	Nodes
2-1	B&C Model 1	0,06	1,26	1,81	9,11	2,1	5,74
	B&C Model 2	0,11	3,12	9,04	54,4	25,61	94,55
	CPLEX	0,31	26,03	31,04	1098,89	2156,76	28878,8
3-1	B&C Model 1	1,01	7,13	16,71	38,86	559,06 (20)	519,31
	B&C Model 2	4,01	14,5	181,54	237,96	809,99 (13)	494,36
	CPLEX	9,8 (29)	305,8	1258,8 (14)	15678,4	Final Gap 25,66%	
3-2	B&C Model 1	0,1	0,06	0,52	0,9	5,59	18,3
	B&C Model 2	0,54	1,7	2,04	4,57	38,66	73,1
	CPLEX	0,45	32,7	13,53	185,5	107,35	4239,63
4-2	B&C Model 1	3,99	15,26	412,69 (27)	475,03	116,85 (13)	116,63
	B&C Model 2	5,97	12,17	612,38 (23)	648,83	562,67 (13)	379
	CPLEX	65,46 (29)	1120,9	884,3 (26)	1744,1	1487,8 (1)	2934

Table 1: Graphs of 20 vertices

k		Low		Medium		High	
		Time	Nodes	Time	Nodes	Time	Nodes
2	B&C Model 1	0,67	3,13	15,75 (23)	31,34	105,5 (24)	165,79
	B&C Model 2	2,66	4,47	121,48 (23)	140,26	770,62 (15)	727,3
	CPLEX	3,59	90,66	31,58 (10)	510,88	Final Gap 38,26%	
3	B&C Model 1	2,81	4,07	21,59	28,46	925,79(22)	885,19
	B&C Model 2	9,12	9,37	120,96	68,57	1790,87 (18)	916
	CPLEX	21,98	152,38	296,47 (29)	742,36(29)	Final Gap 29,53%	

Table 2: Graphs of 30 vertices

The proposed cutting plane algorithm, operating at the nodes of a Branch and Bound tree, has shown to be particularly useful in substantially reducing the number of Branch and Bound nodes, as well as CPU time.

According to the computational experience, the first model outperformed the other. A comparison with version 10.1 of CPLEX shows that the Branch-and-Cut algorithm proposed here reduces both the CPU time and the number of nodes explored in the Branch-and-Bound tree and that it is capable of solving instances that are out of the reach of CPLEX.

References

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

[2] A. Billionnet, *Using Integer Programming to Solve the Train-Platforming Problem*, Transportation Science **37 (2)** (2003) pp. 213–222.

[3] D. Brélaz, *New methods to color the vertices of a graph*, Communications of the ACM **22** (1979) pp. 251–256.

[4] M. Halldórsson and G. Kortsarz, *Multicoloring: Problems and Techniques*, Lecture Notes in Computer Science 3**15** (2004) pp. 25–41.

[5] P. Hansen, M. Labbé and D. Schindl, *Set covering and packing formulations of graph coloring: algorithms and first polyhedral results*, Tech. Rep. G-2005-76, Cahier du GERAD, Montreal, 2005.

[6] T. Jensen and B. Toft, *Graph coloring problems*, Wiley-Interscience, New York, 1995.

[7] D.S. Johnson, A. Mehrotra and M.A. Trick, *Special Issue on Computational Methods for Graph Coloring and its Generalizations*, Discrete Applied Mathematics **156**, 2008.

[8] J. Lee and F. Margot, *On a binary-encoded ILP coloring formulation*, INFORMS J. Computing **19(3)** (2007) pp. 406-415.

[9] I. Méndez-Díaz and P. Zabala, *A polyhedral approach for graph coloring*, Electronic Notes in Discrete Mathematics **7**, 2001.

[10] I. Méndez-Díaz and P. Zabala, *A Branch-and-Cut algorithm for graph coloring*, Discrete Applied Mathematics **154(5)** (2006) pp. 826–847.

[11] A. Mehrotra and M. Trick, *A column generation approach for Graph Coloring*, INFORMS Journal on Computing **8** (1996) pp. 344–354.

[12] A. Mehrotra and M. Trick, *A Branch and Price Approach for Graph Multicoloring*, Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, Springer Operations Research/Computer Science Interfaces Series **37** (2007) pp. 15-30.

[13] L. Narayan, *Channel assignment and graph multicoloring*, In Handbook of wireless networks and mobile computing, Wiley Series On Parallel And Distributed Computing (2002) pp. 71–94.