A LINEAR INTEGER PROGRAMMING APPROACH FOR THE EQUITABLE COLORING PROBLEM*

Isabel Méndez Díaz[♭], Graciela Nasini[†] and Daniel Severín[†]

[♭]*Fac. de Ciencias Exactas y Naturales, UBA, Argentina (imendez@dc.uba.ar)*
[†]*Fac. de Ciencias Exactas, Ingeniería y Agrim., UNR, Argentina ({nasini, daniel}@fceia.unr.edu.ar)*

Abstract: Branch & Cut algorithms based on the polyhedral study of linear integer programming models have proved to be an important tool for solving coloring problems. The Equitable Coloring Problem is a coloring problem where color class sizes must differ by at most one. In this work, we propose and evaluate integer programming formulations for the Equitable Coloring Problem. The best formulation is then used in the context of a Branch & Cut algorithm which achieves a competitive perfomance.

Keywords: *equitable graph coloring, integer programming, branch and cut*
2000 AMS Subject Classification: 90C27 - 05C15

1 INTRODUCTION AND PREVIOUS WORKS

Given a graph G, the *Equitable Coloring Problem* (ECP) consists of finding the minimum number of colors needed in order to have a coloring of G such that any pair of color classes differ in size by at most one. Several applications such as parallel memory systems [1] and load balancing in task scheduling [3] can be modeled as an ECP (for more details, see for example [4]).

Let $G = (V, E)$ be a simple graph, where $V = \{1, \ldots, n\}$ and E are the sets of vertices and edges of G respectively. Given a k-coloring of G, we denote by C_j the set of vertices painted with color j, for each $j = 1, \ldots, n$. An *equitable k-coloring* of G (or just k-eqcol) is a k-coloring of G that satisfies the *equity constraints*: $||C_i| - |C_j|| \leq 1$, for $i, j = 1, \ldots, k$. Alternatively, a k-coloring is a k-eqcol if and only if $\lfloor n/k \rfloor \leq |C_j| \leq \lceil n/k \rceil$ for each $j = 1, \ldots, k$. The *equitable chromatic number* of G, $\chi_{eq}(G)$, is the minimum k for which there exists a k-eqcol in G. Finding this number is an NP-Hard problem [5].

Exact algorithms based on the polyhedral study of linear integer programming (IP) models have proved to be an important tool for solving the traditional coloring problem (CP) [2, 6, 8]. Clearly, IP models for CP can be adapted for ECP by modeling the equity constraints as linear inequalities in terms of variables used in the IP model. In these cases, valid inequalities for CP become valid also for ECP and the cutting-plane stage of a Branch & Cut (B&C) algorithm that solves the CP can be reused in a B&C algorithm for solving ECP. This is the idea behind the B&C algorithm presented in [9], where ECP is modeled by adding equity constraints to the CP-model presented in [2]. We refer to that algorithm as B&C-LF_2.

In this work, we consider the CP-model presented in [8], which has shown a good performance in the context of a B&C algorithm. We propose three different ways of expressing equity constraints and we choose the one with best behavior according to our computational experiences. We develop a B&C algorithm that uses valid inequalities proposed in [8] for the cutting-plane stage, and we compare our algorithm against B&C-LF_2, concluding that our algorithm presents a better performance.

2 MODELS FOR THE ECP

In the CP-model given in [8], colorings are represented by using binary variables x_{vj} and w_j for each vertex $v \in V$ and color $1 \leq j \leq n$ as follows: $x_{vj} = 1$ if and only if color j is assigned to vertex v and

*This research was partially supported by PICT 38036 ANPCyT.

$w_j = 1$ if and only if color j is used by some vertex. The formulation is:

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} w_j & & \\
s.t. \quad & \sum_{j=1}^{n} x_{vj} = 1, & \forall\, v \in V, & \qquad (1)\\
& x_{uj} + x_{vj} \le w_j, & \forall\, uv \in E,\ j = 1, \ldots, n, & \qquad (2)\\
& w_{j+1} \le w_j, & \forall\, j = 1, \ldots, n-1, & \qquad (3)\\
& x_{vj}, w_j \in \{0, 1\}, & \forall\, v \in V,\ j = 1, \ldots, n, &
\end{aligned}
$$

where constraints (1) assert that each vertex has to be painted by an unique color, constraints (2) guarantee that two adjacent vertices can not share the same color, and constraints (3) remove some symmetric solutions by forbidding to use color $j+1$ if color j is not used.

Unlike the CP, we have to consider isolated vertices of G in the ECP [7]. This fact forces us to add the constraints

$$x_{ij} \le w_j, \qquad \forall\, i \in I,\ j = 1, \ldots, n, \tag{4}$$

where $I \subset V$ is the set of isolated vertices of G, imposing that, if color j is not used, no isolated vertex can be painted with color j.

In [7], we presented a model that avoids a class of symmetric colorings, imposing that $|C_j| \ge |C_{j+1}|$ for $j = 1, \ldots, n-1$. In a such k-eqcol, if t_k^j denotes the size of color class j and $p = n \mod k$, we have that $t_k^j = \lfloor n/k \rfloor + 1$ if $j \le p$, and $t_k^j = \lfloor n/k \rfloor$ if $p < j \le k$. By using this property and the fact that a solution is a k-eqcol if and only if $w_k - w_{k+1} = 1$, equity constraints are modeled as:

$$\sum_{v=1}^{n} x_{vj} = w_n + \sum_{k=j}^{n-1} t_k^j (w_k - w_{k+1}), \qquad \forall\, j = 1, \ldots, n-1. \tag{5}$$

Moreover, with the addition of (5) some constraints can be deleted: (2) and (4) for $j = \lfloor n/2 \rfloor + 1, \ldots, n$ become redundant. We refer to the formulation comprised of (1), (3), (2) and (4) for $j = 1, \ldots, \lfloor n/2 \rfloor$, and (5) as $M1$.

In the following two models, we eliminate some symmetric colorings by imposing that a vertex v should be painted by a color j such that $j \le v$. For attaining this elimination, we only need to fix

$$x_{vj} = 0, \qquad \forall\, j = v+1, \ldots, n. \tag{6}$$

However, (6) can be incompatible with constraints (5). Then, in order to use (6), we must remodel the equity constraints. We first propose to express the restriction $\lfloor n/k \rfloor \le |C_j| \le \lceil n/k \rceil$, for each $j = 1, \ldots, k$, by the following constraints:

$$\sum_{v \in V} x_{vj} \ge w_n + \sum_{k=j}^{n-1} \left\lfloor \frac{n}{k} \right\rfloor \big(w_k - w_{k+1}\big), \qquad \forall\, j = 1, \ldots, n-1, \tag{7}$$

$$\sum_{v \in V} x_{vj} \le w_n + \sum_{k=j}^{n-1} \left\lceil \frac{n}{k} \right\rceil \big(w_k - w_{k+1}\big), \qquad \forall\, j = 1, \ldots, n-1. \tag{8}$$

We refer to the formulation comprised of (1), (2), (3), (4), (6), (7) and (8) as $M2$.

The second way is to strengthen the constraint $\lfloor n/k \rfloor \leq |C_j| \leq \lceil n/k \rceil$ by the disjunction $|C_j| = \lfloor n/k \rfloor \vee |C_j| = \lceil n/k \rceil$. We add binary variables y_j for $j = 1, \ldots, n-1$ and the constraints

$$\sum_{v \in V} x_{vj} = y_j + w_n + \sum_{k=j}^{n-1} \left\lfloor \frac{n}{k} \right\rfloor (w_k - w_{k+1}), \qquad \forall\, j = 1, \ldots, n-1. \tag{9}$$

The formulation that uses (1), (2), (3), (4), (6) and (9) is referred as $M3$. Let us notice that we can not delete (2) and (4) for $j = \lfloor n/2 \rfloor + 1, \ldots, n$ in $M2$ and $M3$, as we did with $M1$.

We compared the performance of the three formulations by using a Cut & Branch algorithm based on the standard Branch & Bound algorithm of CPLEX 10.1 plus a custom cutting-plane algorithm that generates strong cuts in the root node, in orden to speed up the optimization. The cuts added in the root node are called *clique inequalities* and are separated by a greedy heuristic (more details of its implementation can be found in [8]).
The test consisted of comparing the behavior of $M1$, $M2$ and $M3$ on 252 randomly generated graphs, and was performed on a Sun UltraSparc workstation. The following table summarizes the results. The first two columns display the number of vertices and the graph density: *low* means 0-33%, *medium* means 33-66% and *high* means 66-100%. Columns 3-5 give the percentages of succesfully solved instances (s.s.i.) for each formulation. An instance is not solved whether the optimizer exceeds the time limit (2 hours). Columns 6-8 give the evaluated nodes in the format "a (b)", where a is the average of the evaluated nodes over instances solved by all formulations and b is the average of the evaluated nodes over instances solved by each formulation. Columns 9-11 give the averages of the elapsed time with the same format as columns 6-8.

n	Dens.	% of s.s.i.			Evaluated nodes			Time in sec.		
		$M1$	$M2$	$M3$	$M1$	$M2$	$M3$	$M1$	$M2$	$M3$
	low	100	100	100	3(3)	2(2)	2(2)	0(0)	0(0)	0(0)
25	med.	100	100	100	55(55)	15(15)	16(16)	3(3)	0(0)	0(0)
	high	100	100	100	1293(1293)	125(125)	23(23)	27(27)	1(1)	0(0)
	low	100	100	100	11(11)	20(20)	39(39)	1(1)	0(0)	1(1)
30	med.	100	100	100	3175(3175)	1184(1184)	1323(1323)	317(317)	51(51)	58(58)
	high	84	100	100	3603(3603)	338(347)	244(302)	577(577)	14(17)	11(15)
	low	100	100	100	182(182)	12(12)	8(8)	6(6)	1(1)	1(1)
35	med.	81	100	100	4381(4381)	1633(2360)	802(1755)	696(696)	104(181)	57(164)
	high	67	90	90	2109(2109)	3994(5713)	1298(3116)	895(895)	603(745)	131(409)

As it can be appreciated from the table, $M3$ outperforms $M1$ and $M2$, both in nodes and time. In particular, the CPU time difference increases with graph size. This behavior shows that $M3$ has a good potential for developing a competitive B&C algorithm for the ECP, based on this formulation.

3 A Branch and Cut algorithm for ECP

From the results given above, we decided to implement a B&C for the ECP based on $M3$. We implemented several routines related to the development of a B&C algorithm, such as initial heuristics and cutting-plane algorithms. Details of these routines are not given due to lack of space.

In this section, we compare the performance between B&C-LF_2 and our algorithm (called B&C-$M3$). We use the same instances as [9] in order to make the comparison fair. The instances were taken from the DIMACS database, except the *kneser* instances corresponding to the well known Kneser family of graphs. B&C-$M3$ were executed on an Intel 1.6Ghz using CPLEX 11 as the LP-solver, while B&C-LF_2 were executed on an AMD-Athlon 1.8Ghz using XPRESS 2005.
The following table resumes the results. The first three columns display the name of the instance, the number of vertices and the equitable chromatic number, respectively. Columns 4-6 give the evaluated nodes and columns 7-9 give the time elapsed. An asterisk indicates that the current instance was solved by the initial heuristics.

Name	n	χ_{eq}	Evaluated nodes		Time in sec.	
			B&C-$M3$	B&C-LF_2	B&C-$M3$	B&C-LF_2
miles750	128	31	0	6	1	171
miles1000	128	42	0*	13	0*	267
miles1500	128	73	0*	1	0*	13
zeroin.i.1	211	49	0	1	0	50
zeroin.i.2	211	36	7	23	6	510
zeroin.i.3	206	36	7	28	7	491
queen6-6	36	7	205	1	13	1
queen7-7	49	7	2	1	4	0
queen8-8	64	9	2332	27	1327	441
myciel3	11	4	0	7	0	0
myciel4	23	5	179	237	0	5
jean	80	10	0*	1	0*	4
anna	138	11	0*	2	0*	26
david	87	30	0	1	0	13
games120	120	9	0*	1	0*	30
kneser5-2	10	3	0*	1	0*	0
kneser7-2	21	6	628	357	1	6
kneser7-3	35	3	4	4	0	2
kneser9-4	126	3	0	4	0	809
1-FullIns-3	30	4	0	34	0	2
2-FullIns-3	52	5	0	84	0	25
3-FullIns-3	80	6	0	38	0	85
4-FullIns-3	114	7	0	3	1	72
5-FullIns-3	154	8	0	5	1	268

We conclude that B&C-$M3$ evaluates fewer nodes than B&C-LF_2, and seems to consume less time. The only exception is the subset of *queen* graphs, where B&C-LF_2 shows a better behavior.

Clearly, our algorithm should be improved when we incorporate strong valid inequalities specific to the ECP (and not to the CP) to the cutting-plane stage. The next step in our research is to perform a polyhedral study of $M3$ in order to find families of valid inequalities for the ECP, and make separation routines for them. Finally, we also will incorporate other elements (such as primal heuristics and variable branching selection) that enrich the algorithm.

REFERENCES

[1] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling Computer and Manufacturing Processes*. Springer Verlag, 1997.

[2] Manoel Campêlo, Ricardo Corrêa, and Victor Campos. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, 156(7):1097–1111, 2008.

[3] Sajal K. Das, Irene Finocchi, and Rossella Petreschi. Conflict-free star-access in parallel memory systems. *J. Parallel Distrib. Comput.*, 66(11):1431–1441, 2006.

[4] Marek Kubale et al. *Graph Colorings*. American Mathematical Society, Providence, Rhode Island, 2004.

[5] Hannah Furmańczyk and Marek Kubale. The complexity of equitable vertex coloring of graphs. *Journal of Applied Computer Science*, 13:97–107, 2005.

[6] A. Mehrotra and M. Trick. A column generation approach for graph coloring. *INFORMS J. Comput.*, 8:344–353, 1996.

[7] Isabel Méndez-Díaz, Graciela Nasini, and Daniel Severín. A polyhedral approach for the graph equitable coloring problem. *VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, 2008.

[8] Isabel Méndez-Díaz and Paula Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159–179, 2008.

[9] L. Bahiense and Y. Frota and N. Maculan and T. Noronha and C. Ribeiro. A Branch-and-cut for equitable coloring. *International Network Optimization Conference*, 2009.