Industrial Engineering and Applications
L.-C. Tang (Ed.)

doi:10.3233/ATDE230065

A Constraint Programming Model for the B-Coloring Problem

Roberto MONTEMANNI [a,1], Gabriel Henrique CARRARETTO [b], Umberto Junior MELE [b], and Luca Maria GAMBARDELLA [b]

[a] *Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy*

[b] *Faculty of Informatics, Universitá della Svizzera Italiana, Lugano, Switzerland*

ORCiD ID: Roberto Montemanni https://orcid.org/0000-0002-0229-0465, Umberto Junior Mele https://orcid.org/0000-0002-8464-1889, Luca Maria Gambardella https://orcid.org/0000-0002-3736-0419

Abstract. B-coloring is a theoretical optimization problem on a graph that, on top of being used to model some real-world applications, is exploited by some bounding techniques embedded into solvers for the classical graph coloring problem. This implies that improved solutions for the b-coloring problem have an impact on an even larger pool of practical applications modelled by graph coloring in several different fields such as scheduling, timetabling and telecommunications.

The b-coloring problem aims to maximize the number of colors used to provide a complete coloring for a graph $G = (V, E)$, while preventing adjacent vertices from receiving the same color. Moreover, each color used is associated to a so-called a b-vertex. A vertex can be a *b-vertex* only if the set of colors assigned to its adjacent vertices includes all the colors used, apart from the one assigned to the vertex itself.

In this work we discuss a new Constraint Programming model for the b-coloring problem and we show how such a paradigm can improve state-of-the-art results for several benchmarks instances commonly adopted in the literature.

Keywords. B-Coloring, Optimization, Constraint Programming, Modelling, Heuristic Solutions

1. Introduction

In the last decades, Operations Research and Optimization have been providing planning tools capable of improving our everyday life in virtually all contexts. At the basis of such tools there are mathematical models able to describe real-life problems in abstract terms, on which it is possible to apply more formal reasonings. In this paper we focus the b-coloring problem, that can be formally described below. Several real applications of b-coloring to real-world contexts, both direct and indirect, exist. We direct the interested reader to [1] for further details.

Given an undirected graph $G = (V, E)$, a b-coloring with K colors is a function that assigns a color $c(i) \in C = \{1, 2, \ldots, K\}$ to each vertex i of V, so that $c(i) \neq c(j)$ for every

[1] Corresponding Author: Roberto Montemanni, roberto.montemanni@unimore.it.

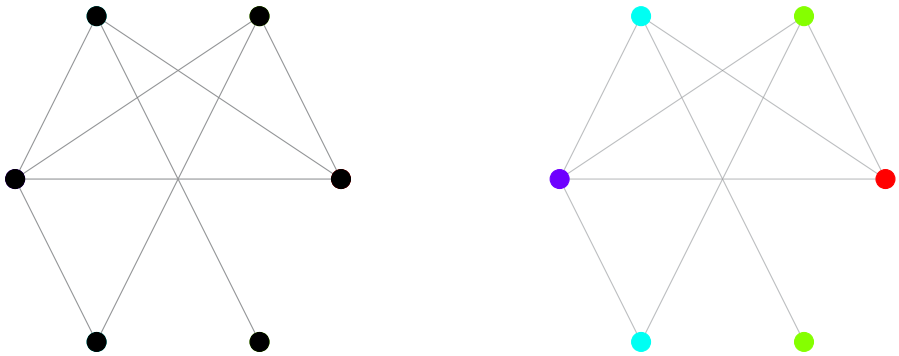

Figure 1. Example of a graph with an associated optimal b-coloring with 4 colors. The b-vertices are the two on the top row and the two in the middle row.

$(i, j) \in E$. Let $N(i) = \{j|(i, j) \in E\}$ be the neighborhood of i. For each $k \in C$ there must be a vertex $i \in V$ with $c(i) = k$ and with $N(i) \cap \{j \in V|c(j) = h\} \neq \emptyset \ \forall h \in C \backslash \{k\}$. In other words, for each color k used, a vertex assigned to color k (called *b-vertex*) must exist such that for every other color used h, there is at least one of its neighbors assigned to h. The optimization targer is to find a b-coloring using the maximum possible number of colors. The *b-chromatic number* of a graph G is defined as the maximum number of colors for which G admits a b-coloring, and is normally indicated as $X_b(G)$.

An example of an optimal b-coloring for a given graph is provided in Figure 1.

2. Literature Review

The b-coloring problem is NP-hard, since it is proven in [2] that providing an estimation for $X_b(G)$ NP-hard itself. Some theoretical properties of the problem are also known: the difference between the optimal solution values of the classical coloring problem ([3]) and b-coloring for the same graph G can be arbitrarily large [4]; the girth (length of a shortest cycle) of the graph can influence the b-coloring problem heavily [5]; Given a value for which a b-coloring exists for a graph G and the b-chromatic number $X_b(G)$, a b-coloring with k colors does not necessarily exist for all the possible values of k ranging within such an interval, instead gaps might exist [6].

Algorithmic contributions to the b-coloring problem can be classified as follows. In [7] a hybrid evolutionary algorithm is discussed, while a method for the calculation of the b-chromatic index $X_b(G)$ based on an integer linear programming formulation is introduced in [8]. Later on, the same model is used within a branch and cut algorithm [9]. An advanced mixed integer linear programming model is presented in [10], where some matheuristic approaches [11] are also derived. The latter paper also contributed the testbed commonly adopted for the b-coloring problem, which is composed of instances originally proposed for other graph problems in [12]. A further matheuristic method, based on an iterative schema, is discussed in [13]. Finally, improvements on both lower and upper bounds generated by integer programming models are documented in [1].

Concerning applications of b-coloring to real-world problems, in [14] and [15] a postal mail sorting systems based on b-coloring is introduced to model a new approach for address block localization, with the aim is of assisting the software for address recognition. A clustering technique using some b-coloring concepts is used by the French healthcare system to identify and formalize a new typology of hospital stays, as presented in [16]. Finally, as discussed in [17], the b-coloring problem can provide viable bounds for the classical coloring problem. Note that due to this, enhancement in the methods to solve b-coloring may lead in turns to benefits for several important practical applications such as scheduling [18], timetabling [19] and telecommunications [20].

The rest of the paper is organized as follows. In Section 3 a Constraint Programming model for b-coloring is discussed. Section 4 presents comprehensive computational results related to the performance of the model while solved by state-of-the-art solvers. The instances traditionally used in the b-coloring literature were considered. Section 5 finally contains some conclusions. Note that the approach presented in this paper is an extension of that discussed in the Master thesis [21].

3. A Constraint Programming model

In this section a Constraint Programming model for the b-coloring problem, baed on the idea of the Integer Programming model originally proposed in [10], is discussed. The rationale behind the use of Constraint Programming [22] is that such paradigm appears to be well-suited for problems that can be described by Boolean variables. Moreover, recent advances in solvers, make Constraint Programming an efficient tool, thanks to the high scalability on moderns computers characterized by multi-core architectures.

In the model we present there is a set of boolean variables x such that $x_{ij} = 1$ (or equivalently *True*) if vertex j is colored with the color of the representative vertex i, 0 (or equivalently *False*) otherwise. With such a notation, a vertex i is a b-vertex (sometimes referred to as a representative) if and only if $x_{ii} = 1$. Let $\bar{N}(i) = V \setminus \{\{i\} \cup N(i)\}$ be the anti-neighborhood of i.

$$\max \sum_{i \in V} x_{ii} \tag{1}$$

$$\sum_{j \notin N(i)} x_{ji} = 1 \qquad \forall i \in V \tag{2}$$

$$\neg x_{ii} \implies \bigwedge_{j \in \bar{N}(i)} \neg x_{ij} \qquad \forall i \in V \tag{3}$$

$$\neg x_{ij} \vee \neg x_{ik} \qquad i \in V; j,k \in \bar{N}(i); (j,k) \in E \tag{4}$$

$$x_{ii} \wedge x_{jj} \implies \bigvee_{\substack{k \in N(j) \\ k \in \bar{N}(i)}} x_{ik} \qquad \forall i,j \in V; (i,j) \notin E \tag{5}$$

$$x_{ij} \in \{0,1\} \qquad i,j \in V \tag{6}$$

The objective function (1) maximizes the number of b-vertices selected. Constraints (2) ensure that every vertex is assigned exactly one color (observe that, by definition, j

can take value i in the summation). Constraints (3) states that a vertex can give its color only when it is a representative (note the ose of the logical operator *AND*). Constraints (4) guarantees a proper coloring, by imposing that two adjacent nodes must be assigned different colors. Note the ose of the logical operator *OR*, and that these constraints works in conjunction with the previous ones to guarantee proper b-coloring. Constraints (5) formalize the proper b-coloring restrictions. They imply that if both vertices i and j are b-vertices, then there must be at least a neighbor of j which is represented by color i. The domain definition for the variables is provided by constraints (6).

4. Experimental results

The model described in Section 3 will be tested from an empirical viewpoint, to understand its potential.

4.1. Datasets and Settings

The instances considered in this work are those commonly adopted in the literature to validate b-coloring approaches have been originally proposed in [10]. They are based on the DIMACS benchmark set originally proposed for the minimum coloring and the maximum clique problems in [12]. Being the approach we discuss a model-based one, and given the performance of the solver currently available, it is not likely that instances with more than 500 vertices will be handled effectively. Therefore, the instances of the dataset with more than 500 vertices are left out of the present study. Moreover, all the instances for which an optimal solution has been proven in previous studies are left out as well, in consideration of our objective, which is the improvement of state-of-the-art results. This leaves us with a total of 72 instances: 29 of which originally proposed for the graph coloring problem [3] and 43 originally proposed for the maximum clique problem [23].

The Constraint Programming model discussed in Section 3 has been implemented in Python and solved with the CP-SAT solver from Google OR-tools 9.5.2237 [24] with default settings. All the experiment reported in this section has been carried out on the processors of a cluster, each equipped with an x86 Intel Xeon Platinum 8276 processor with 24 cores running at 2.4 GHz. For each instance, one run with a maximum computation time of 24 hours is consider, Once the allowed time is reached, the run is interrupted and the best lower and upper bounds are saved. Note that the powerful hardware and the longer computation time give to the approach a clear advantage over the methods previously appeared in the literature. However, we do not expect the previous methods to be able to fully take advantage of such experimental conditions, due to substantially limited scalability of the Linear Programming solvers on which they are based. The files containing the solutions retrieved are available upon request to the authors.

4.2. Results and Discussion

The results obtained by solving the Constraint Programming model of Section 3 are reported in Tables 1 and 2, divided by origin of the instances composing the benchmark set. The columns of the tables contain the following information:

Table 1. Results for the relevant coloring instances from [12].

Instance			Best Known		CP Model	
Name	$\|V\|$	$\|E\|$	LB	UB	LB	UB
dsjc125.5	125	3891	45	63	*39*	*72*
dsjc125.9	125	6961	68	72	68	*73*
DSJC250.1	250	3218	24	33	24	*137*
DSJC250.5	250	31336	57	126	**63**	*155*
DSJC250.9	250	55794	128	150	**129**	*158*
DSJC500.1	500	12458	38	59	38	*308*
DSJC500.5	500	62624	88	251	**106**	*392*
DSJC500.9	500	112437	249	443	*245*	**339**
DSJR500.1c	500	242550	150	221	**156**	**162**
DSJR500.5	500	58862	221	234	-	*329*
flat300_20_0	300	21375	61	144	**69**	*199*
flat300_26_0	300	21633	64	146	**68**	*210*
flat300_28_0	300	21696	57	146	**69**	*202*
le450_15a	450	8168	40	57	*39*	*275*
le450_15b	450	8169	40	56	*34*	*281*
le450_15c	450	16680	54	93	*51*	*277*
le450_15d	450	16750	54	92	*52*	*277*
le450_25a	450	8260	54	63	-	*253*
le450_25b	450	8263	52	60	*47*	*261*
le450_25c	450	17343	59	101	*53*	*275*
le450_25d	450	17425	60	99	*57*	*277*
le450_5a	450	5714	28	34	*24*	*282*
le450_5b	450	5734	28	34	*24*	*280*
le450_5c	450	9803	35	52	35	*282*
le450_5d	450	9757	36	52	*34*	*286*
R250.1c	250	30227	86	89	86	**86**
R250.5	250	14849	116	119	*107*	*151*
school1	385	19095	70	117	70	*219*
school1_nsh	352	14612	59	101	**61**	*195*

- *Instance* contains three subcolumns reporting the name, the number of vertices and the number of edges of each instance considered;
- *Best Known* reports for each instance the best known lower bound (heuristic solution cost) and upper bound. The results summarise the achievement of the previous relevant literature [1], [10] and [13];
- *CP Model* reports for each instance the lower and upper bounds retrieved by solving he Constraint Programming model described in Section 3, according to the descriptions provided in Section 4.1.

Entries in italics in the tables denote suboptimal bounds for the Constraint Programming model-based method, while bold entries indicate improved best-known bounds. Entries with a dash mean that no feasible solution has been retrieved in the given time. The new heuristic solutions are available upon request to the authors.

The results of Table 1 suggest that the Constraint Programming model is fairly effective in providing good heuristic solutions (LBs), and in fact 8 new best-known results emerged (over 29 instances). Note that suboptimal results are always close to best-known results. Different is the situation concerning the upper bounds: in several cases the new solver provides estimations that lie very far from best-known results, notwithstanding 3 improved results are retrieved, and one of them also led to proven optimality for one instance (*R250.1c*). The poor upper bounds suggest difficulties for the new model to fully capture the characteristics of the problem on these graph coloring instances.

The results of Table 2 are on maximum clique instances, and denote a much better behaviour of the Constraint Programming model we propose with respect to what seen for the coloring instances of Table 1. In details, 34 lower bounds are improved (over 43 instances), together with 8 upper bounds. Note that in some cases emarkable improvements in lower bounds are achieved. The new model is still denoting difficulties on the upper bound side, but the many improved heuristic solutions provided indicate that the new method can be used as a very viable method for generating high quality solutions, notwithstanding being based on a general solver.

Upon the undoubted success of the present results, future research will try to understand and explain why the new model performs better on the second set of instances. This might depend on weaknesses of the new approach, or by the fact that previous methods were tailored towards the first set of instances, and therefore performing better on them. A further question that should be addressed by future research, while looking for remedies, is why some of the upper bounds provided by Constraint Programming are so far from the best known ones.

5. Conclusion

This paper described a Constraint Programming model for the b-coloring problem, a problem on graphs arising in optimization and used to describe in mathematical terms several real applications and is used, in turns, as a bounding technique for the traditional graph coloring problem.

The experimental results presented to validate the new model produced several improvements for the benchmarks commonly adopted in the literature. On top of the theoretical results, there are potential implications for solution methods of the many real-world applications that can be modelled as b-coloring or traditional graph-coloring.

Future research will try to address the limitations on the quality of some upper bounds provided by Constraint Programming on some of the instances. A possible strategy could be to consider modifications to the basic model as done recently for Integer Programming models of the problem, or to integrate on a more tailored fashion Constraint Programming and Integer Programming.

Acknowledgment

We acknowledge the CINECA award under the Italian SuperComputing Resource Allocation (ISCRA) initiative, for the availability of high performance computing resources and support.

Table 2. Results for the relevant maximum clique instances from [12].

Instance			Best Known		CP Model	
Name	$\|V\|$	$\|E\|$	LB	UB	LB	UB
brock200_1	200	14834	76	127	**77**	*128*
brock200_2	200	9876	48	100	**51**	*123*
brock200_3	200	12048	60	120	**62**	*125*
brock200_4	200	13089	66	124	**68**	*130*
brock400_1	400	59723	123	254	**135**	*306*
brock400_2	400	59786	121	254	**134**	*291*
brock400_3	400	59681	123	254	**134**	*293*
brock400_4	400	59765	125	254	**135**	*294*
C125.9	125	6963	68	71	**69**	*73*
C250.9	250	27984	127	152	**129**	*158*
C500.9	500	112332	250	327	*246*	*345*
gen200_p0.9_44	200	17910	104	118	**105**	*122*
gen200_p0.9_55	200	17910	105	119	105	*122*
gen400_p0.9_55	400	71820	200	261	**201**	*262*
gen400_p0.9_65	400	71820	200	262	**201**	**261**
gen400_p0.9_75	400	71820	200	262	**201**	262
hamming6-4	64	704	15	22	**16**	*23*
hamming8-2	256	31616	144	160	*143*	**159**
hamming8-4	256	20864	56	144	**66**	*153*
johnson8-4-4	70	1855	28	35	28	*36*
johnson16-2-4	120	5460	38	46	38	*51*
johnson32-2-4	496	107880	42	262	**158**	*305*
keller4	171	9435	52	101	**55**	**98**
p_hat300-1	300	10933	48	91	48	*181*
p_hat300-2	300	21928	85	149	**93**	*187*
p_hat300-3	300	33390	117	190	**123**	*198*
p_hat500-1	500	31569	72	152	**75**	*321*
p_hat500-2	500	62946	127	252	**134**	*376*
p_hat500-3	500	93800	152	351	**193**	*366*
san200_0.7_1	200	13930	82	124	**85**	**116**
san200_0.7_2	200	13930	60	115	**65**	**98**
san200_0.9_1	200	17910	105	111	105	**110**
san200_0.9_2	200	17910	106	119	106	119
san200_0.9_3	200	17910	104	119	*105*	*121*
san400_0.5_1	400	39900	41	204	**61**	*232*
san400_0.7_1	400	55860	113	253	**143**	**246**
san400_0.7_2	400	55860	108	251	**132**	251
san400_0.7_3	400	55860	82	248	**117**	*265*
san400_0.9_1	400	71820	203	259	**205**	**257**
sanr200_0.7	200	13868	68	125	**73**	*130*
sanr200_0.9	200	17863	104	119	**105**	*123*
sanr400_0.5	400	39984	80	201	**87**	*292*
sanr400_0.7	400	55869	106	251	**128**	*302*

References

[1] Montemanni R, Chou X, Smith DH. Upper and lower bounds based on linear programming for the b-coloring problem. EURO Journal on Computational Optimization. 2022;10:100049.

[2] Irving RW, Manlove DF. The b-chromatic number of a graph. Discrete Applied Mathematics. 1999;91(1):127-41.

[3] Malaguti E, Toth P. A survey on vertex coloring problems. International Transactions in Operational Research. 2010;17(1):1-34.

[4] Kratochvíl J, Tuza Z, Voigt M. The b-chromatic number of a graph. Lecture Notes in Computer Science. 2002;2573:310-20.

[5] Campos VA, Lima CV, Silva A. B-Coloring Graphs with Girth at Least 8. Proceedings of the Seventh European Conference on Combinatorics, Graph Theory and Applications. 2013:327-32.

[6] Barth D, Cohen J, Faik T. On the b-continuity property of graphs. Discrete Applied Mathematics. 2007;155(13):1761-8.

[7] Fister I, Peterin I, Mernik M, Črepinšek M. Hybrid evolutionary algorithm for the b-chromatic number. Journal of Heuristics. 2015;21:501-21.

[8] Koch I, Peterin I. The b-chromatic index of direct product of graphs. Discrete Applied Mathematics. 2015;190-191:109-17.

[9] Koch I, Marenco J. An integer programming approach to b-coloring. Discrete Optimization. 2019;32:43-62.

[10] Melo RA, Queiroz MF, Santos MC. A matheuristic approach for the b-coloring problem using integer programming and a multi-start multi-greedy randomized metaheuristic. European Journal of Operational Research. 2021;295(1):66-81.

[11] Toklu NE, Montemanni R, Gambardella LM. An ant colony system for the capacitated vehicle routing problem with uncertain travel costs. In: The IEEE Swarm Intelligence Symposium (SIS); 2013. p. 32-9.

[12] Johnson DS, Trick MA. Cliques, coloring, and satisfiability: second DIMACS implementation challenge. American Mathematical Society. 1996;26.

[13] Montemanni R, Smith DH. An Iterative Matheuristic Algorithm for the B-Coloring Problem. In: The 3rd International Conference on Industrial Engineering and Industrial Management (IEIM 2022); 2022. p. 265-70.

[14] Gaceb D, Eglin V, Lebourgeois F, Emptoz H. Improvement of postal mail sorting system. International Journal of Document Analysis and Recognition. 2008;11:67-80.

[15] Gaceb D, Eglin V, Lebourgeois F, Emptoz H. Robust Approach of Address Block Localization in Business Mail by Graph Coloring. International Arab Journal of Information Technology. 2009;6(3):221-9.

[16] Elghazel H, Deslandres V, Hacid MS, Dussauchoy A, Kheddouci H. A New Clustering Approach for Symbolic Data and Its Validation: Application to the Healthcare Data. Lecture Notes in Computer Science. 2006;4203:373-482.

[17] Campos VA, Lima CV, Martins NA, Sampaio L, Santos MC, Silva A. The b-chromatic index of graphs. Discrete Mathematics. 2015;338(11):2072-9.

[18] Montemanni R, Smith DH, Rizzoli AE, Gambardella LM. Sequential ordering problems for crane scheduling in port terminals. International Journal of Simulation and Process Modelling. 2009;5(4):248-61.

[19] Dandashi A, Al-Mouhamed M. Graph Coloring for class scheduling. In: ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010; 2010. p. 1-4.

[20] Montemanni R, Smith DH, Allen SM. An improved algorithm to determine lower bounds for the fixed spectrum frequency assignment problem. European Journal of Operational Research. 2004;156(3):736-51.

[21] Carraretto GH. A SAT approach to generate Lower and Upper Bounds for the b-coloring problem [Master thesis]. Università della Svizzera Italiana; 2023.

[22] Rossi F, van Beek P, Walsh T. Handbook of Constraint Programming. Elsevier Science. New York City, USA; 2006.

[23] Smith DH, Montemanni R, Perkins S. The Use of an Exact Algorithm within a Tabu Search Maximum Clique Algorithm. Algorithms. 2020;13(10).

[24] Perron L, Furnon V. Google OR-Tools; 2023. Https://developers.google.com/optimization/.