RAIRO-Oper. Res. 57 (2023) 1821–1841
https://doi.org/10.1051/ro/2023062

RAIRO Operations Research
www.rairo-ro.org

A PARALLEL LAGRANGIAN HEURISTIC FOR THE FRACTIONAL CHROMATIC NUMBER OF A GRAPH

Paulo Henrique Macêdo de Araújo[1,*], Ricardo C. Corrêa[2] and Manoel Campêlo[3]

Abstract. We propose a new integer programming formulation for the Fractional Chromatic Number Problem. The formulation is based on representatives of stable sets. In addition, we present a Lagrangian heuristic from a Lagrangian relaxation of this formulation to obtain a good feasible solution for the problem. Computational experiments are presented to evaluate and compare the upper and lower bounds provided by our approach.

Mathematics Subject Classification. 05C15, 05C75, 90C11, 90C27.

Received March 3, 2023. Accepted April 28, 2023.

1. Introduction

A stable set of a graph is a subset of its vertices in which the induced subgraph has no edges. This concept is often associated with problems that require the selection of a group of unrelated elements from a given set. Such a situation occurs, for instance, in the assignment of radio frequencies to antennas. In this problem, the network can be described by a conflict graph, with the set of vertices representing the antennas and the edges illustrating possible interference between them when transmitting at the same frequency. As such, a stable set of the conflict graph can be seen as a set of antennas that do not interfere with one another, which is a restriction of every assignment. Stable set based constraints are also present in a variety of other problems, such as school timetabling [9], frequency assignment [12], computer register allocation [7], and iterative solution of linear systems [29].

A typical scenario in which stable set based constraints appear is the description of colorings of a graph. Consider an undirected, simple, nonempty, and connected graph $G = (V, E)$. A *k-fold coloring* of G, for some integer $k > 0$, is an assignment of at least k (distinct) colors to each vertex of G such that the endpoints of any edge are assigned different colors. An example of a 2-fold coloring is illustrated in Figure 1. As each color defines a stable set of G, a k-fold coloring can be viewed as a covering of V by stable sets, where each vertex is covered at least k times.

Colorings are usually associated with measures on the number of colors used. The (*integral*) *chromatic number* $\chi(G)$ is the minimum number of colors needed in a 1-fold coloring of a graph G. The problem of determining

Keywords. Fractional chromatic number, Integer programming, Lagrangian heuristic.

[1] Federal University of Ceará, Quixadá, Brazil.
[2] Digital Humanities Program, Federal Rural University of Rio de Janeiro, Nova Iguaçu, Brazil.
[3] Federal University of Ceará, Fortaleza, Brazil.
*Corresponding author: phmacedoaraujo@ufc.br

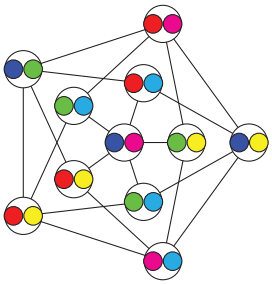


FIGURE 1. A 2-fold coloring with a total number of colors of 6.

the chromatic number (CNP) has been employed to solve various applications and is a strongly NP-hard problem [13]. Closely related to CNP is the problem of finding the *fractional chromatic number* $\chi_F(G)$, to be referred to as FCNP. The value $\chi_F(G)$ is the smallest ratio (total number of colors)/k, taken over all k-fold colorings of G, for all integer $k > 0$ [18,21]. It has been proven to be an NP-hard problem for arbitrary graphs [23]. Furthermore, efficient upper bounds have been established recently for some classes of graphs [10, 11, 14, 25].

A relevant motivation for studying the Fractional Chromatic Number Problem (FCNP) is the celebrated relation $\chi_F(G) \leq \chi(G)$, which provides a lower bound that can be used in enumerative algorithms for the Chromatic Number Problem (CNP). See, for instance, [19,24]. In particular, the algorithm proposed by Mehrotra and Trick showed good performance on moderately sized instances of CNP and was observed to be more effective than previous branch-and-bound algorithms that used an approximation of the maximum clique as a lower bound [24]. As FCNP is in NP-hard, the use of a good approximation of $\chi_F(G)$ (instead of its exact value) is essential for dealing with large instances.

Another important application related to FCNP can be found in the field of telecommunications. In wireless networks, where communication is performed via radio waves and is subject to interference, the *Round Weighting Problem* (RWP) concerns to the demand-weighted communications on the network nodes [20].

The standard approach to FCNP is to solve its classical linear programming formulation (CLP), *i.e.* the linear relaxation of the set covering formulation of CNP with one variable for each stable set of the graph [17]. This demands the use of a column generation method. It typically yields a good lower bound on $\chi_F(G)$ (after many iterations of the method) at a considerable computational cost, since the pricing problem is hard. Unfortunately, this approach may suffer from rounding errors. Besides, it does not provide a k-fold coloring giving the fractional chromatic number, as required in some applications (for instance, RWP mentioned above). An attempt to circumvent these drawbacks would be the use of exact linear solvers [15]. Nevertheless, exact LP-solvers may be significantly slower than column generation methods. Moreover, the number of colors (the denominator of the rational solution) obtained by these solvers is usually very high, which can be impractical for many applications of the problem.

In this paper, we propose a different approach to FCNP that simultaneously provides lower and upper bounds resulting in a narrow gap. In addition, the upper bound is given by a k-fold coloring where the value of k is usually smaller than that one provided by the classical fractional coloring heuristic when comparing solutions with same objective value [1]. The approach is based on a novel linear program for FCNP that employs a polynomial number of variables but an exponential number of constraints. The cutting-plane method is then used to iteratively improve the lower bound of $\chi_F(G)$. This method has the advantage of providing a lower bound in each iteration and can be stopped once an approximation with the desired precision is obtained.

The new model has a linear number of covering constraints that couple stable set polytopes of subgraphs of G. So, a good lower bound relies on a considerably tight partial description of these polytopes. This is made possible by the existence of various structural results and algorithmic techniques for stable set polytopes in the literature, as seen in [22, 26] and references therein. The known valid inequalities generally provide a good

approximation of the polytope. However, the separation heuristics used in such an approach can face a significant challenge if it has to handle stable sets of the entire graph, which can potentially be exponential in number.

The use of representative vertices can mitigate the drawback mentioned above. Originally proposed for formulating the integer coloring problem in [4], this modeling strategy involves partitioning the set of stable sets of G into $|V|$ subsets, with each subset having a vertex of G as its representative. In this paper, we propose the application of representative vertices for FCNP. By doing so, separation heuristics can be applied to subgraphs that are typically much smaller than G, albeit at the cost of increasing the number of variables by a factor that may be as large as n. To the best of our knowledge, this is the first linear programming formulation for FCNP with a polynomial number of variables (disregarding the dual of formulation CLP).

The representatives formulation constraints define a polytope that is closely related to the stable set polytope, such that the separation heuristics for rank inequalities can also be applied [26, 28]. Actually, by relaxing the covering constraints in a Lagrangian way, we obtain separable stable set subproblems that can be solved in parallel. It is the basis for the proposed parallel Lagrangian heuristic.

The text is described as follows. Section 2 shows classical greedy heuristics for CNP and FCNP, and some basic notation used in this text. An explanation of the representatives model is shown in Section 3, along with representatives formulations for the cited coloring problems. In Section 4, a Lagrangian relaxation of the representatives formulation for FCNP is presented, while its resolution technique, based on a Lagrangian heuristic, is showed in Section 5. Computational experiments are reported in Section 6.

2. Greedy heuristics for coloring problems

In this section, we present greedy heuristics to obtain approximate solutions for both CNP and FCNP. These heuristics are later utilized to construct colorings as part of a Lagrangian heuristic, which is discussed in subsequent sections. The discussion in this section pertains to the integer and fractional coloring of arbitrary graphs.

2.1. Definitions and notation

We denote an arbitrary simple graph (without loops and multiple edges) by $G = (V, E)$. We define $V = [n] = \{0, 1, \ldots, n-1\}$. For each $u \in V$, $N(u) = \{v \in V \mid uv \in E\}$ is the neighborhood of vertex u in G. In the description of the coloring algorithms presented in this paper, we use the terms "color", "color class", and "stable set" interchangeably.

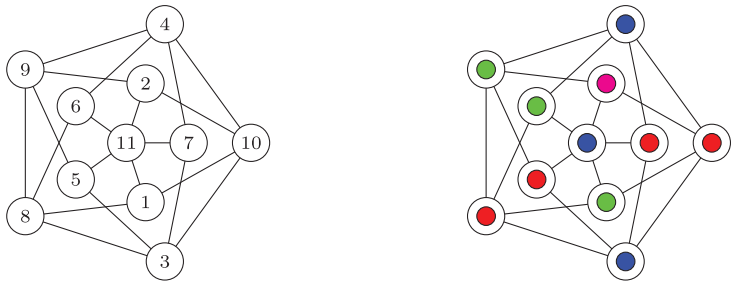
As mentioned in the Introduction, a *k-fold coloring* in G, for some $k \in \mathbb{N}$, is a coloring assignment to the vertices of G such that each vertex is assigned to at least k distinct colors and adjacent vertices receive disjoint color sets. Thus, a k-fold coloring can be seen as a multiset of stable sets where each vertex belongs to at least k of them. The minimum number of distinct colors used in a k-fold coloring of G is denoted $\chi_k(G)$.

Particularly for $k = 1$, a k-fold coloring is also called an *integer coloring* or simply a *coloring*. Besides, $\chi(G) = \chi_1(G)$ is known as the *chromatic number* of G, and the problem of determining $\chi(G)$ is known as the *Chromatic Number Problem* (*CNP*). A solution to CNP is given by a minimum-size set of stable sets covering V. It is a well-known NP-hard problem [13].

The *Fractional Chromatic Number Problem* (*FCNP*) in G consists in determining the *fractional chromatic number* $\chi_F(G)$, *i.e.* the minimum ratio $\chi_k(G)/k$ over all integer $k > 0$. This minimum is indeed achieved [8]. Then, the value k and a k-fold coloring giving $\chi_F(G)$ are also desirable. It is known that FCNP is NP-hard [23]. A solution to FCNP is given by a *fractional coloring*, *i.e.* a set of stable sets, each one assigned a fractional weight, such that each vertex belongs to stable sets whose weights add up to at least 1. Precisely, $\chi_F(G)$ is the minimum total weight of a fractional coloring [17].

2.2. Greedy heuristic for the chromatic number

An iterative greedy heuristic is commonly used to solve CNP, generating an integer coloring for a given graph. The heuristic works by first establishing a vertex order and then assigning colors to each vertex one at a time.



(A) (B)

FIGURE 2. Example of a graph and a coloring produced by `DSATUR`. (a) A graph and (b) a coloring with 4 colors.

Specifically, when traversing the vertices in the established order, each vertex is assigned the color with smallest label that has not been assigned to any of its neighbors. A well-known example of this approach is `DSATUR` (for *Degree SATURation*) [2], which dynamically adjusts the vertex ordering. The `DSATUR` algorithm initializes the number of available colors to $\ell = 1$ and sets the corresponding color class to $S_\ell = \emptyset$. In each iteration $k = 1, \ldots, n$, it proceeds in the following manner:

- Each vertex is assigned a *saturation degree*, which is the number of distinct colors assigned to vertices in its neighborhood in previous iterations. The assignment of a color to a vertex, say v, at the end of an iteration can cause a change in the saturation degree of the neighbors of v.
- The non-colored vertices are ordered in non-increasing order of their current saturation degrees. To break ties, priority is given to the vertex with the greatest degree (number of vertices in its neighborhood).
- The first vertex v in the established ordering is chosen, and the possible color with smallest label j is assigned to it, chosen among those in $S_1, \ldots, S_\ell$. If the chosen color is $S_j = S_\ell$, then ℓ is incremented by 1 and S_ℓ initializes as the empty set. Then, for all $w \in N(v)$ such that $N(w) \cap S_\ell = \{v\}$, the saturation degree of w is also incremented by 1.

An execution of `DSATUR` for the graph in Figure 2a is summarized in Table 1. Initially, the saturation degrees are all zero. In the first column, k stands for the index of the current iteration in the execution. The second column gives the vertex v with the lowest cost chosen in iteration k, and the color S_j modified in this iteration appears in the fourth column, with index j in the third column. The resulting saturation degrees are indicated in the last column.

The strategy of assigning the possible color with smallest label to each vertex in each iteration is very common in coloring algorithms. The $O(|V|^3)$ time complexity of `DSATUR` is mainly due to the vertex ordering maintenance and the calculation of the possible color with smallest label for each vertex to be colored [2].

2.3. Greedy heuristic for the fractional chromatic number

The pseudocode described by Algorithm 1 is referred to as `FCP` (for *Fractional Coloring Procedure*). Its purpose is to generate k-fold colorings, for some values of k, and select, among these colorings, the one with the lowest ratio (total number of colors/k) [1].

For $k = 1$, a 1-fold coloring (integer coloring) $\mathcal{S}_1 = \{S_1, \ldots, S_q\}$ of G is heuristically obtained. The coloring $\mathcal{S}_1$ partitions the vertices of G, where some color classes may not be maximal with respect to set inclusion. Then, we proceed to $k = 2$ and try to extend the color classes of $\mathcal{S}_1$ (lines 3–7). We attempt to assign one more color to each vertex, from the already used colors. Let U be the set of vertices that do not receive a second color after this step. If we use any integer coloring heuristic for the subgraph $G[U]$ to obtain a coloring, say $\mathcal{S}_2$, (line 8), we can thus find a second color for all vertices in U. If $|\mathcal{S}_1| + |\mathcal{S}_2|/2 < |\mathcal{S}_1|$, then the fractional coloring

TABLE 1. Execution of DSATUR on the graph of Figure 2a.

k	v	j	S_j	Sat. degrees										
1	11	1	{11}	0	0	0	0	0	1	1	1	1	1	–
2	5	2	{5}	0	1	0	1	0	1	1	–	1	1	–
3	9	3	{9}	1	–	1	1	0	2	1	–	1	1	–
4	2	4	{2}	1	–	1	1	1	–	1	–	1	1	–
5	4	1	{11,4}	–	–	1	1	2	–	1	–	1	1	–
6	10	2	{5,10}	–	–	1	1	–	–	1	–	2	1	–
7	1	3	{9,1}	–	–	1	1	–	–	1	–	–	1	–
8	3	1	{11,4,3}	–	–	2	–	–	–	1	–	–	1	–
9	8	2	{5,10,8}	–	–	–	–	–	–	2	–	–	1	–
10	6	3	{9,1,6}	–	–	–	–	–	–	–	–	–	1	–
11	7	2	{5,10,8,7}	–	–	–	–	–	–	–	–	–	1	–
			Vertices	4	9	8	3	10	2	6	5	1	7	11

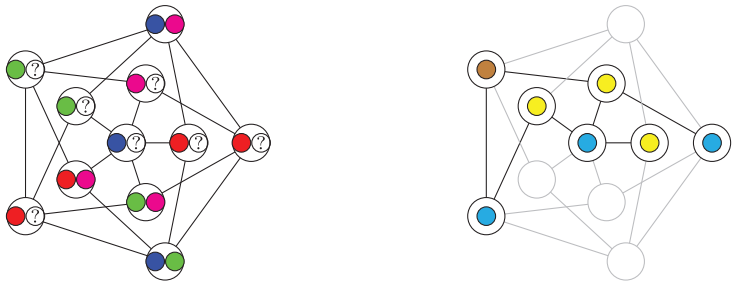
value of $\mathcal{S}_1 \cup \mathcal{S}_2$ is less than that of $\mathcal{S}_1$. The idea is to repeat this procedure in each iteration, incrementing the number of colors that each vertex receives by 1 until the fractional coloring value becomes worse than the one of the previous iteration, or the number of color classes reaches a given limit N_{lim}.

Algorithm 1: FCP– Fractional coloring greedy procedure.

Data: Graph instance G.
Result: Fractional Coloring of G.

Initialize: $\chi \leftarrow \infty$, $\mathcal{S}^* \leftarrow \mathcal{S} \leftarrow \emptyset$, $U \leftarrow V$, $c \leftarrow 0$, $k \leftarrow 1$, $stop \leftarrow false$
repeat
  $U \leftarrow V$
  forall $v \in U$ **do**
    if *there is* $S \in \mathcal{S}$ *such that* $v \notin S$ *and* $N(v) \cap S = \emptyset$ **then**
      $S \leftarrow S \cup \{v\}$
      $U \leftarrow U \setminus \{v\}$
  Apply integer coloring heuristic to $G[U]$, giving $\mathcal{S}'$ as the returned coloring
  $\mathcal{S} \leftarrow \mathcal{S} \uplus \mathcal{S}'$
  $c \leftarrow c + |\mathcal{S}'|$
  if $c/k \leq \chi$ **then**
    if $c/k < \chi$ **then**
      $\chi \leftarrow c/k$
      $\mathcal{S}^* \leftarrow \mathcal{S}$
    $k \leftarrow k + 1$
  else
    $stop \leftarrow true$
until $c > N_{\text{lim}}$ *or* $stop = true$
return $\mathcal{S}^*$

Variable $\mathcal{S}$ ($\mathcal{S}^*$ for the best solution) stores the collection of all generated color classes in the algorithm. The operator $\uplus$ represents the union of collections, allowing for repetition of elements in the resulting collection ($\{S\} \uplus \{S\} = \{S, S\} \neq \{S\}$). Any color class generated at an iteration can be extended at any iteration. Variable



(A) (B)

FIGURE 3. Iteration $k = 2$ of FCP for the graph of Figure 2a. (a) Extension with second color and (b) Coloring produced by DSATUR.

χ contains the value of the best fractional coloring obtained so far. The DSATUR algorithm is used as the integer coloring heuristic in line 8 in our implementation.

To illustrate the algorithm's performance, consider the example shown in Figure 2. At iteration $k = 1$, the set of uncolored vertices is $U^1 = \{1, 2, \ldots, 11\}$, and the first generated coloring can be observed in Figure 2b. This particular coloring leads to $\chi = |\mathcal{S}| = 4$. At $k = 2$, it is possible to add vertices $1, 4, 5$ to color class S_4, and vertex 3 to S_3, as indicated in Figure 3a. As a result, U^2 becomes $U^2 = \{2, 6, 7, 8, 9, 10, 11\}$. The integer coloring heuristic generates $\mathcal{S}'$ as depicted in Figure 3b. Hence, $(|\mathcal{S}| + |\mathcal{S}'|)/2 = 3.5 < 4$, and, consequently, χ becomes 3.5, while $\mathcal{S}$ is updated to $\mathcal{S} \uplus \mathcal{S}'$ and $\mathcal{S}^*$ is updated to the best solution so far. After 4 iterations, the best fractional coloring that is obtained is a 2-fold coloring that contains 12 colors in total, with $\chi = 3$.

The complexity of the FCP heuristic is mainly determined by the execution of lines 3–7 and line 8 in each iteration, which yields $O\big(|V|^2 \cdot (N_{\text{lim}})^2 + |V|^3 \cdot N_{\text{lim}}\big)$ [1].

3. Representatives model

This section presents a new Lagrangian heuristic for the FCNP, which is based on a formulation using the notation of the asymmetric representatives model proposed in [4]. Unlike the classical formulation for FCNP that has a variable for each stable set of the graph [17], the representatives formulation involves a polynomial number of variables. Therefore, the column generation technique is not applicable.

The representatives formulation has two important features that are exploited in the proposed Lagrangian heuristic. The first feature is the presence of stable set polytope constraints, which enables the use of known techniques to separate several of its facets. The second exploited feature is a decomposition of the Lagrangian relaxation formulation into independent subproblems, thus allowing for parallel resolution and faster computation. We initiate the model presentation with the CNP, and then present its adaptation to the FCNP.

3.1. Notation and terminologies

The *complement* of $G = (V, E)$ is denoted by $\bar{G} = (V, \bar{E})$, where $\bar{E}$ is the set of all edges not in E. The *neighborhood* and *anti-neighborhood* of a vertex $u \in V$ are given by $N(u) = \{v \in V \mid uv \in E\}$ and $\bar{N}(u) = V \backslash (N(u) \cup \{u\})$, respectively. Given an order on V, we define the *positive anti-neighborhood* of u by $\bar{N}^+(u) = \big\{v \in \bar{N}(u) \mid u < v\big\}$ and the *negative anti-neighborhood* by $\bar{N}^-(u) = \big\{v \in \bar{N}(u) \mid v < u\big\}$. We also denote $\bar{N}^+[u] = N^+(u) \cup \{u\}$ and $\bar{N}^-[u] = N^-(u) \cup \{u\}$.

For a subset $H \subseteq V$, the subgraph induced by H in $\bar{G}$ is denoted by $\bar{G}[H] = \big(H, \bar{E}[H]\big)$, where $\bar{E}[H]$ is the set of edges in $\bar{G}$ with both endpoints in H. We write $\bar{G}^-(u)$ to represent $G\big[\bar{N}^-(u)\big]$ and $\bar{G}^+(u)$ to represent $G\big[\bar{N}^+(u)\big]$. Similarly, $\bar{G}^-[u] = G\big[\bar{N}^-[u]\big]$ and $\bar{G}^+[u] = G\big[\bar{N}^+[u]\big]$. Figure 4 illustrates the notation for a graph G, where u is the smallest vertex in order.

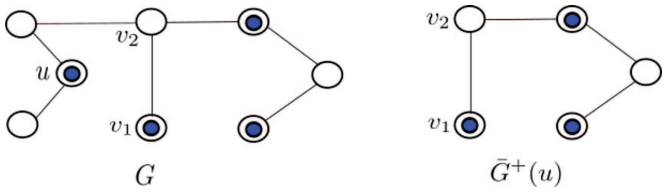


FIGURE 4. An example of the representatives model notation. Vertex u is the smallest in the vertex order.

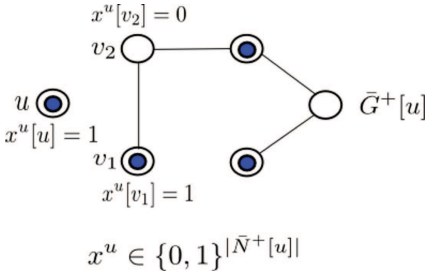


FIGURE 5. Example of the representatives model notation. The marked vertices define a stable set represented by u.

3.2. A representatives formulation for CNP

In the representatives model, each color has a representative vertex that represents the remaining vertices of the stable set assigned to this color. The variables for the proposed formulation comprise vectors $x^u \in \{0,1\}^{|\bar{N}^+[u]|}$, for all $u \in V$. Each vector contains binary variables indexed by vertices in $\bar{N}^+[u]$. For $u, v \in V$, $u \leq v$ and $uv \in \bar{E}$, $x^u[v] = 1$ if, and only if, u represents v (more precisely, u represents the color assigned to v). Moreover, $x^u[u] = 1$ if, and only if, u is the representative of some color. Besides integrality, the constraints defining an integer coloring in G are shown below:

$$\sum_{u \in \bar{N}^-[v]} x^u[v] \geq 1, \qquad v \in V \tag{1}$$

$$x^u[v] + x^u[w] \leq x^u[u], \qquad u \in V,\ vw \in E,\ v, w \in \bar{N}^+(u) \tag{2}$$

$$0 \leq x^u[v] \leq x^u[u], \qquad u \in V,\ v \in \bar{N}^+(u), \tag{3}$$

$$x^u[u] \leq 1, \qquad u \in V. \tag{4}$$

Constraints (1) ensure that each vertex gets at least one color, while constraints (2) prevents two adjacent vertices to be assigned to the same color. We call constraints (2) *edge constraints*. A binary vector $x^u \subseteq \{0,1\}^{|\bar{N}^+[u]|}$ satisfying constraints (2) and (3) is the *characteristic vector* of a stable set $S \subseteq V$ represented by u, where, for all $v \in \bar{N}^+[u]$, $x^u[v] = 1$ if, and only if, $v \in S$. In case of $x^u[u] = 0$, the stable set represented by u is empty. Figure 5 illustrates the notation of the variables expressed by x for the example of Figure 4. The marked vertices form a stable set represented by u.

3.3. A representatives formulation for FCNP

It can be easily seen that a feasible solution for the above formulation is a coloring of the graph. As a preliminary step towards achieving the ultimate goal of adapting the previous formulation to FCNP, the binary

variables x are converted to integer variables and the constraints (1)–(4) are adjusted as

$$\sum_{u\in\bar{N}^-[v]} x^u[v] \geq k, \qquad v \in V \tag{5}$$

$$x^u[v] + x^u[w] \leq x^u[u], \qquad u \in V,\ vw \in E,\ v,w \in \bar{N}^+(u) \tag{6}$$

$$0 \leq x^u[v] \leq x^u[u], \qquad u \in V,\ v \in \bar{N}^+(u) \tag{7}$$

$$x^u[u] \leq k, \qquad u \in V \tag{8}$$

$$k \in \mathbb{N}. \tag{9}$$

Given vertices u and $v \in \bar{N}^+(u)$, variables $x^u[u]$ and $x^u[v]$ now represent the number of stable sets that are represented by u (with potential repetitions), and the number of those stable sets that contain v, respectively. Additionally, variable k specifies the lower bound on the number of colors that each vertex must be assigned to. Nonetheless, it should be noted that these constraints alone are insufficient to fully capture the requirements of FCNP.

The formulation's laxity is attributable to constraints (6) and (7). This is due to the fact that converting variables x from binary to integer renders these constraints no longer sufficient to describe stable sets. To illustrate, consider a vertex u and a clique v, w, z in its positive anti-neighborhood. Although the point $x^u[u] = 2$, $x^u[v] = 1$, $x^u[w] = 1$, and $x^u[z] = 1$ satisfies constraints (6) and (7), it implies that each vertex of the clique belongs to half of the stable sets represented by u. Accordingly, at least one of these sets contains two vertices of the clique. Therefore, to accurately describe a representative formulation for FCNP, we must incorporate additional constraints such as maximal cliques, holes, anti-hole constraints, and so on.

Let

$$\mathcal{Y}(G) = \left\{ y \in \{0,1\}^{|V|} \middle| y[v] + y[w] \leq 1, \quad \forall\, vw \in E \right\}$$

be the set of characteristic vectors of stable sets of G. Considering a fixed value $s \in \mathbb{N}$, we also define the set

$$\text{STAB}_s^I(G) = \left\{ \sum_i \alpha_i y_i \middle| \sum_i \alpha_i = s,\ \alpha_i \in \mathbb{N},\ y_i \in \mathcal{Y}(G),\ 1 \leq i \leq |\mathcal{Y}(G)| \right\}.$$

Note that the coefficients $\alpha_1, \ldots, \alpha_{|\mathcal{Y}(G)|}$ define an integer linear combination of the stable sets of G. Since all of these coefficients are natural numbers and their sum is equal to s, $\text{STAB}_s^I(G)$ is the set of all possible choices of s stable sets of G (with repetitions allowed).

Let x_+^u denote the subvector of x^u with entries indexed by $\bar{N}^+(u)$, which consists of all entries of x^u except $x^u[u]$. Then, constraints (5), (8), and (9) together with the requirement that $x^u[u] \in \mathbb{N}$ and $x_+^u \in \text{STAB}^I_{x^u[u]}(\bar{G}^+(u))$, for each $u \in V$, represent the integer points of the k-fold coloring polytope [5]. The minimization of $\sum_{u\in V} x^u[u]/k$ over these points gives a program for FNCP.

To formulate a linear programming model, we apply a variable transformation where each variable x is divided by k. It results in the following representatives formulation for FCNP:

$$(FC) \quad \min \sum_{u\in V} x^u[u] \tag{10}$$

$$\text{s.t.} \sum_{u\in\bar{N}^-[v]} x^u[v] \geq 1, \qquad v \in V \tag{11}$$

$$x_+^u \in \text{STAB}_{x^u[u]}(\bar{G}^+(u)), \qquad u \in V \tag{12}$$

$$x^u[u] \leq 1, \qquad u \in V \tag{13}$$

In this transformation, given a fixed $s = \frac{s'}{k} \in \mathbb{Q}_+$, the set of all possible choices of s' stable sets of G, *i.e.* $\text{STAB}_s^I(G)$, gives way to

$$\text{STAB}_s(G) = \text{conv}\left\{ \frac{z'}{k} \in \mathbb{Q}_+^{|V|} \middle| z' \in \text{STAB}_{s'}^I(G) \right\}$$

$$= \text{conv}\left\{ \sum_i \alpha_i y_i \Big| \sum_i \alpha_i = s,\ \alpha_i \in \mathbb{Q}_+,\ y_i \in \mathcal{Y}(G),\ 1 \le i \le |\mathcal{Y}(G)| \right\}.$$

Constraints (11)–(13) describe the fractional coloring representatives polytope. It should be noted that the polyhedron is rational, and therefore its vertices are rational points. Consequently, we can disregard the rationality constraints on the variables x in the formulation.

For a scalar $\sigma \in \mathbb{R}$ and a set $X \subseteq \mathbb{R}^n$, let $\sigma \cdot X = \{\sigma x | x \in X\}$. Observe that $\text{conv}(\sigma \cdot X) = \sigma \cdot \text{conv}(X)$. By applying the multiplier transformation $\beta_i = \frac{\alpha_i}{s}$, $1 \le i \le |\mathcal{Y}(G)|$, in the expression of $\text{STAB}_s\ (G)$, we can rewrite it as

$$\begin{aligned} \text{STAB}_s\ (G) &= s \cdot \text{conv}\left\{ \sum_i \beta_i y_i \Big| \sum_i \beta_i = 1,\ \beta_i \in \mathbb{Q}_+,\ y_i \in \mathcal{Y}(G),\ 1 \le i \le |\mathcal{Y}(G)| \right\} \\ &= s \cdot \text{conv}(\mathcal{Y}(G)) = \text{conv}(s \cdot \mathcal{Y}(G)). \end{aligned}$$

Then, we get the expression

$$\text{STAB}_s\ (G) = \text{conv}\Big\{ y \in \{0, s\}^{|V|} \big| y[v] + y[w] \le s,\ v, w \in V,\ vw \in E \Big\}. \tag{14}$$

It is worth noting that $\text{STAB}_1\ (G) = \text{STAB}\ (G)$ is the extensively studied stable set polytope [22]. Additionally, $\text{STAB}_s\ (G) = s \cdot \text{STAB}\ (G)$, indicating that $\text{STAB}_s\ (G)$ and $\text{STAB}\ (G)$ have a two-way relationship, where each element of $\text{STAB}_s\ (G)$ corresponds to an element of $\text{STAB}\ (G)$ multiplied by s. It follows that checking whether a non-null point x^u satisfies constraint (12) is equivalent to checking whether $x^u_+ / x^u[u]$ belongs to $\text{STAB}(\bar{G}^+(u))$. Furthermore, $a^\top x \le a_0$ is a valid inequality for $\text{STAB}\ (G)$ if and only if $a^\top x \le a_0 s$ is valid for $\text{STAB}_s\ (G)$.

4. Lagrangian relaxation

The Lagrangian relaxation of a mathematical programming formulation consists in moving some of the constraints into the objective function, with their violations penalized by a corresponding Lagrangian multiplier. A key point is how to determine the multipliers leading to the best dual bounds. A classical approach is the so-called subgradient method.

4.1. Lagrangian relaxation for *FC*

To perform the Lagrangian relaxation of FC, we opt to relax constraints (11), which are referred to as *covering constraints*, following the approach proposed in [3]. Each of these constraints is associated with a vertex of G, and link variables that correspond to different representatives. By relaxing these constraints, the model can be decomposed into $|V|$ independent subproblems, each one corresponding to a representative vertex.

The relaxation of FC for a given vector of fixed multipliers λ, $\lambda \ge \mathbf{0}$, becomes

$$\begin{aligned} L(\lambda) = \min\ & \sum_{u \in V} \left(x^u[u] + \lambda[u] \left(1 - \sum_{v \in \bar{N}^-[u]} x^v[u] \right) \right) \\ \text{s.t.}\ & x^u_+ \in \text{STAB}_{x^u[u]}(\bar{G}^+(u)), && u \in V \\ & 0 \le x^u[u] \le 1, && u \in V. \end{aligned}$$

By rearranging the objective function, we obtain the problem $\text{FC}(\lambda)$, written as

$$L(\lambda) = \min \sum_{u \in V} \left(\lambda[u] + (1 - \lambda[u]) x^u[u] - \sum_{v \in \bar{N}^+(u)} \lambda[v] x^u[v] \right) \tag{15}$$

$$\text{s.t.} \quad x_+^u \in \text{STAB}_{x^u[u]}(\bar{G}^+(u)), \qquad u \in V \tag{16}$$

$$0 \leq x^u[u] \leq 1, \qquad u \in V. \tag{17}$$

Upon analyzing the objective function (15), it becomes apparent that, for every $u \in V$, constraint (17) can be substituted by $x^u[u] \in \{0,1\}$. This is because setting $x^u[u] = 1$ results in an objective function value that is at least as good as setting $x^u[u] = \epsilon$, where $0 < \epsilon < 1$. With this observation, we can decompose $FC(\lambda)$ into a set of maximum weight stable subproblems, one for each $u \in V$, as outlined below.

Let $FC_u(\lambda)$ be the following *maximum weight stable set subproblem* associated with $u \in V$:

$$\alpha_u(\lambda) = \max \sum_{v \in \bar{N}^+(u)} \lambda[v] x^u[v]$$

$$\text{s.t.} \quad x_+^u \in \text{STAB}\big(\bar{G}^+(u)\big). \tag{18}$$

Suppose that $\bar{x}_+^u$ is an optimal solution for $FC_u(\lambda)$. Then, an optimal solution x for $FC(\lambda)$ is obtained as follows: if $\alpha_u(\lambda) > 1 - \lambda[u]$, then set $x^u[u] = 1$ and $x_+^u = \bar{x}_+^u$; otherwise, set $x^u[u] = 0$ and $x_+^u = 0$. Thus, we can define the set of representatives in an optimal solution as $\text{Rep}(\lambda) = \{u \in V \mid \alpha_u(\lambda) > 1 - \lambda[u]\}$. It follows that the optimal value of $FC(\lambda)$ is equal to

$$L(\lambda) = \sum_{u \in V \setminus \text{Rep}(\lambda)} \lambda[u] + \sum_{u \in \text{Rep}(\lambda)} (1 - \lambda[u] - \alpha_u(\lambda)).$$

The Lagrangian dual problem of FC, denoted by LD, is defined as

$$L(\lambda^*) = \max \; L(\lambda)$$
$$\text{s.t.} \quad \lambda \in \mathbb{R}_+^{|V|}.$$

It is a concave piecewise linear program.

4.2. 2-Phase Subgradient Algorithm for the Lagrangian dual problem

To solve the Lagrangian dual, we propose a 2-*Phase Subgradient Algorithm* (`2-PSA`) that is based on the three-phase algorithm presented in [6]. The essential component is the known *Subgradient Method (SM)*, a widely employed iterative approach to solve dual problems derived from Lagrangian relaxations [27]. The main purpose of the SM is to provide tight lower bounds for the original problem. Starting with initial values, the method entails updating Lagrangian multipliers in each iteration using the subgradient of the corresponding piecewise linear objective function. In the sequel, we provide an in-depth explanation of the method for our specific case. Additionally, our proposed algorithm not only produces an effective lower bound but also generates a good feasible solution.

4.2.1. Subgradient phase

The first phase of the proposed algorithm is referred to as the *subgradient phase*, with its primary objectives consisting of determining an appropriate lower bound for the Lagrangian dual problem and identifying a suitable vector of Lagrangian multipliers to be used in the second phase. Both objectives are achieved through the utilization of the SM.

Let λ^t denote the value of λ at the beginning of iteration $t \geq 0$. The execution of SM commences at iteration $t = 0$, with arbitrary initial values for λ^0. We use initial value 1 for all multipliers for the sake of simplicity. At any iteration $t \geq 0$, the value λ^t defines problem $FC(\lambda^t)$, whose optimal value is $L(\lambda^t)$. Let $\bar{x}$ be an optimal solution of this problem. The subgradient of $L(\lambda)$ at $\lambda = \lambda^t$ is the vector $\Delta(\lambda^t)$, whose components are

$$\Delta(\lambda^t)[u] = 1 - \sum_{v \in \bar{N}^-[u]} \bar{x}^v[u], \quad u \in V. \tag{19}$$

The update of the Lagrangian multipliers is accomplished through

$$\lambda^{t+1}[u] = \max\left\{\lambda^t[u] + \psi \cdot \Delta(\lambda^t)[u] \cdot \frac{(UB - L(\lambda^t))}{\| \Delta(\lambda^t) \|^2},\ 0\right\}, \quad u \in V, \tag{20}$$

where UB is an upper bound for FC and ψ is a parameter used to regulate the stepsize taken along the subgradient direction. To improve convergence, we use the objective value of a feasible solution provided by a Lagrangian heuristic as an upper bound. This heuristic is described in Section 5.

Typically, if the lower bound fails to improve after several iterations, it may indicate that the stepsize along the subgradient is too large. In such a case, the stepsize is gradually decreased until it falls below a certain threshold. Specifically, if the best lower bound found so far is not improved after N_{stuck} consecutive iterations, then the value of ψ is halved. The procedure ends when the stepsize becomes smaller than a parameter denoted by S_ψ or when $\| \Delta(\lambda^t) \|$ approaches zero. The choice of parameters N_{stuck}, S_ψ, and the initial value of ψ can significantly affect the quality of the solution.

In each iteration of the SM, the value $L(\lambda^t)$ provides a lower bound for the FCNP. However, these values may oscillate over the iterations.

4.2.2. Heuristic phase

The subsequent phase of the `2-PSA` is intended to produce a feasible solution to the original problem. In this *heuristic phase*, a sequence of Lagrangian multipliers is determined. Multipliers selected from the outcomes of the preceding subgradient phase serve as the initial basis for the procedure, which is then executed repetitively until it attains convergence. Each iteration requires the use of the Lagrangian heuristic `FCPLagRel` of Section 5 to address a subproblem, where each of the identified multipliers is employed as input, ultimately producing a feasible solution for FCNP.

Using (15), we can observe that the Lagrangian multipliers perturb the coefficients of the objective function of FC. The Lagrangian heuristic leverages the potential of multipliers that generate tight lower bounds to produce good feasible solutions. Although it is merely a heuristic approach, it has been successfully applied in various applications [6].

4.2.3. Algorithm overview

The entire procedure is precisely described in Algorithm 2. The outer loop comprises two phases introduced in the previous subsections, each specified in consecutive inner loops. To elaborate on the details, we introduce additional notation: $bestLB$ and $bestUB$ refer to the best lower and upper bounds, respectively, found so far; $bestLB'$ refers to the best lower bound found by the current iteration of the outer loop; x^* denotes the solution associated with $bestUB$, and λ^* denotes the multipliers associated with $bestLB'$. These variables are initialized in line 1.

The inner loop between lines 5 and 19 refers to the execution of the subgradient phase. In the first run of this phase, the Lagrangian multipliers gets 1. In each iteration t of the subgradient phase, the relaxed problem $FC(\lambda^t)$ is solved to provide a lower bound value of $L(\lambda^t)$ at line 6. This lower bound value is used to update $bestLB$, $bestLB'$, and the Lagrangian multipliers. Additionally, λ^* is updated with a small perturbation of the multipliers that provided the best lower bound. This disturbance is achieved by adding a random value, $\frac{0.1}{\text{rand}}$, to each Lagrangian multiplier. Here, rand is an integer randomly selected in the interval $[-100, 100]$ such that rand $\neq 0$. The subgradient is then computed, and the Lagrangian multipliers are updated using (20) with a fixed upper bound UB. In the first iteration of the outer loop, UB is set to the value of the solution obtained from the `FCP` heuristic (Algo. 1). In subsequent iterations of `2-PSA`, UB is updated with the value of the best solution found thus far in the heuristic phase, $bestUB$. If $bestLB'$ is not updated within N_{stuck} consecutive iterations, then the stepsize ψ is halved. First phase terminates when the value of the stepsize ψ becomes smaller than the parameter S_ψ or $\| \Delta(\lambda^t) \|$ approaches zero (with a threshold of 10^{-6} set in our experiments).

The heuristic phase is executed as described in lines 21–36. SM is utilized with modifications in comparison to the subgradient phase. The Lagrangian multipliers start with the same values as those which provided the

Algorithm 2: `2-PSA`– Determining lower and upper bounds for FCNP.

Data: Relaxed program $FC(\lambda)$.
Result: Lower bound, upper bound, and feasible solution for FCNP.

$\lambda^* \leftarrow 1$, $bestLB \leftarrow -\infty$, $bestUB \leftarrow \infty$
$UB \leftarrow$ `FCP(`FC`)`
repeat
  `/*** Subgradient phase ***/`
  $t \leftarrow 1$, $\psi \leftarrow 2$, $bestLB' \leftarrow -\infty$, $\lambda \leftarrow \lambda^*$
  repeat
    Solve a relaxation of $FC(\lambda)$ providing solution of value $\overline{L}(\lambda)$
    if $\overline{L}(\lambda) > bestLB$ **then**
      $bestLB \leftarrow \overline{L}(\lambda)$
    if $\overline{L}(\lambda) > bestLB'$ **then**
      $bestLB' \leftarrow \overline{L}(\lambda)$
      $\lambda^* \leftarrow \lambda + \frac{0.1}{\text{rand}}$
      $t \leftarrow 0$
    else if $t \geq N_{\text{stuck}}$ **then**
      $\psi \leftarrow \psi/2$
      $t \leftarrow 0$
    Calculate subgradient $\Delta(\lambda)$
    Update multipliers λ using UB and $\overline{L}(\lambda)$
    $t \leftarrow t + 1$
  until $\psi < S_\psi$ *or* $\| \Delta(\lambda) \| < 10^{-6}$
  `/*** Heuristic phase ***/`
  $t \leftarrow 1$, $\psi \leftarrow 2$, $\lambda \leftarrow \lambda^*$
  repeat
    Solve a relaxation of $FC(\lambda)$ providing solution of value $\overline{L}(\lambda)$
    if $\overline{L}(\lambda) > bestLB$ **then**
      $bestLB \leftarrow \overline{L}(\lambda)$
    if $\overline{L}(\lambda) > bestLB'$ **then**
      $bestLB' \leftarrow \overline{L}(\lambda)$
      $\lambda^* \leftarrow \lambda + \frac{0.1}{\text{rand}}$
    Calculate subgradient $\Delta(\lambda)$
    $\hat{x} \leftarrow$ `FCPLagRel(`FC`,`λ`)`
    $c\hat{x} \leftarrow$ value of $\hat{x}$ in (10)
    if $c\hat{x} < bestUB$ **then**
      $bestUB \leftarrow c\hat{x}$
      $x^* \leftarrow \hat{x}$
    Update multipliers λ using $c\hat{x}$ and $\overline{L}(\lambda)$
    $t \leftarrow t + 1$
  until $t > N_{p2}$
  $UB \leftarrow bestUB$
until *bestUB is not improved, or the total number of colors reaches a limit*
return $bestLB$, $bestUB$, x^*

best lower bound, $bestLB'$, in the first phase of the current iteration of `2-PSA`. In contrast to the first phase, the value of the stepsize ψ remains constant, and the upper bound UB used in (20) is determined by the objective function (10) for the solution $\hat{x}$ provided by the Lagrangian heuristic `FCPLagRel` with the current λ as input. If the value $c\hat{x}$ of this solution is the best found so far, then $bestUB$ and x^* are updated accordingly. The heuristic phase concludes after N_{p2} iterations. The value of the best feasible solution is utilized as a fixed upper bound on the subgradient phase of the next iteration of the outer loop.

Both phases are executed alternately until the best solution found x^* is not improved, or the total number of colors reaches a limit.

4.3. Cutting planes and pricing

In our preliminary experimentation, we noticed that the resolution time of the subproblems $FC_u(\lambda)$ in each subgradient iteration could significantly differ, depending on the specific instance at hand. These subproblems are maximum weight stable set problems, which are notoriously difficult to solve optimally. To circumvent this difficulty, we utilized a cutting-plane method to determine the optimal value of a relaxation of $FC_u(\lambda)$. The cutting-plane method starts with the linear relaxation of the integer programming formulation in which the fractional constraint (18) is replaced by $x^u_+ \in \mathcal{Y}\big(\bar{G}^+(u)\big)$. Subsequently, we add cuts to strengthen the relaxation. This procedure yields an upper bound $\overline{\alpha}_u(\lambda)$ for $\alpha_u(\lambda)$. More information on the separation strategy is available in Subsection 6.1. The combination of these relaxations for all $u \in V$ still constitutes a relaxed Lagrangian dual, which enables us to compute the subgradient iteration faster. This provides a lower bound

$$\overline{L}(\lambda) = \sum_{u \in V \setminus \mathrm{Rep}(\lambda)} \lambda[u] + \sum_{u \in \mathrm{Rep}(\lambda)} (1 - \lambda[u] - \overline{\alpha}_u(\lambda))$$

that may be less accurate than $L(\lambda)$.

The computation time of the `2-PSA` algorithm in the subgradient phase can be further reduced by adopting a pricing procedure inspired by [6], which resembles the partial pricing technique commonly used in linear programming. This procedure is regularly applied to select a subset of vertices to serve as candidates for the role of representatives in a subsequent sequence of T consecutive subgradient iterations. This results in solving only a portion of the stable set subproblems during these iterations, which saves computation time. When applied in an iteration t of the subgradient phase, the pricing procedure determines whether each subproblem $FC_u(\lambda)$ is active or idle, and active subproblems are solved in all T subgradient iterations until the next pricing execution, while idle subproblems remain unsolved in all such iterations. The set of active vertices is given by $\overline{\mathrm{Rep}}(\lambda^t) = \{u \in V | \overline{\alpha}_u(\lambda^t) > 1 - \lambda^t[u]\}$. The idea behind this choice is to set as idle all the subproblems associated with vertices that are not selected as representatives in the solution of the relaxation of $FC(\lambda^t)$. We speculate that such vertices would also remain as non-representatives in an optimal solution of $FC(\lambda^*)$, for a vector λ^* of Lagrangian multipliers associated with the best lower bound found by `2-PSA`.

The pricing application frequency is managed by the parameter T, which represents the number of subgradient iterations until pricing is applied again. In our implementation, it is initialized to $T = 20$. At every iteration t in which the pricing procedure is applied, we calculate the deviation $\delta = \tilde{L}(\lambda^t) - L(\lambda^t)$ between the true lower bound $L(\lambda^t)$ and the approximate lower $\tilde{L}(\lambda^t)$ given by the relaxation of $FC(\lambda^t)$ with constraint $x^u[v] = 0$ for all $v \in \bar{N}^+[u]$ and $u \in V \setminus \overline{\mathrm{Rep}}(\lambda^t)$. Note that this restricted problem (the one solved after disregarding the idle subproblems) may not provide a valid lower bound. If the deviation δ is small enough, we decrease the frequency of the pricing application by increasing the parameter T; otherwise, we decrease the value of T. Specifically, we update T according to the following rules: $T = 4 \cdot T$ if $\delta \leq 10^{-5}$, $T = 1.5 \cdot T$ if $\delta \leq 10^{-1}$, $T = 1.1 \cdot T$ if $\delta \leq 0.5$, and $T = 20$ otherwise. This approach is similar to the one used in [6].

We replaced lines 5–19 of Algorithm 2 with the pseudocode of the subgradient phase with pricing, which is written in Algorithm 3. Additionally, it should be noted that $bestLB'$ is used to store the best value of such a problem in the current iteration of the outer loop of `2-PSA`, which is not guarantee to be a valid lower bound.

Algorithm 3: Including pricing procedure in the subgradient phase of `2-PSA`.

```
/*** To be included between lines 4 and 5 of 2-PSA ***/
T ← 20
Solve a relaxation of FC(λ) providing a solution of value L̄(λ)
Apply pricing to determine Rep = Rep(λ)
bestLB' ← L̄(λ)
if L̄(λ) > bestLB then
    bestLB ← L̄(λ)
/*** Modified inner loop of the subgradient phase ***/
repeat
    t' ← 1
    repeat
        Solve active subproblems in Rep to get a restricted solution value L̃(λ) of FC(λ)
        Lines 9–18 of 2-PSA to update bestLB', the Lagrangian multipliers, and λ* with L̃(λ) replacing L̄(λ)
        t' ← t' + 1
    until t' > T
    Lines 6–8 of 2-PSA to update bestLB
    Apply pricing to determine Rep = Rep(λ)
    δ ← L̃(λ) − L̄(λ)
    if δ ≤ 10^−5 then
        T ← 4T
    else if 10^−5 < δ ≤ 10^−1 then
        T ← 1.5T
    else if 10^−1 < δ ≤ 0.5 then
        T ← 1.1T
    else if 0.5 < δ then
        T ← 20
until ψ < S_ψ or ‖ Δ(λ) ‖< 10^−6
```

5. Lagrangian heuristic

The proposed heuristic for obtaining a fractional coloring is a modified version of the `FCP` algorithm. The modification involves using a dynamic vertex ordering and changing the heuristic of line 8 of Algorithm 1 by an integer coloring heuristic that is guided by Lagrangian multipliers. The resulting adapted heuristic, denoted as `FCPLagRel(`G`,`λ`)`, takes the graph and Lagrangian multipliers as input. Its objective is to determine a k-fold coloring $\mathcal{S}$ such that the ratio $|\mathcal{S}|/k$ is as close to $\overline{L}(\lambda)$ as possible.

The `FCPLagRel(`G`,`λ`)` algorithm utilizes an integer coloring heuristic referred to as `LagRelCol`, which is a greedy list heuristic. When applied as `LagRelCol(`$G[R]$`,`λ`)`, it provides a coloring of the subgraph $G[R]$ that is induced by a subset $R \subseteq V$. To determine the vertex ordering used by the heuristic, a cost $c_\lambda[u]$ is assigned to each vertex $u \in R$ based on the Lagrangian multipliers λ. This cost is defined in terms of the coefficients in (15) of the variables indexed by vertices that can still represent u. Initially, $c_\lambda[u]$ is given by the sum of the coefficients of $x^u[u]$ and $x^w[u]$, where $w \in N^-(u) \cap R$. That is,

$$c_\lambda[u] = 1 - \lambda[u] - \sum_{w \in \bar{N}^-(u) \cap R} \lambda[u] = 1 - \lambda[u] - \left|\bar{N}^-(u) \cap R\right| \lambda[u]. \tag{21}$$

As the `LagRelCol(`$G[R]$`,`λ`)` iterations progress, the colors assigned to some vertices may restrict the ability of other vertices to act as representatives. For example, suppose that there are pairwise non-adjacent ordered vertices $v_1,\ v_2,\ v_3$. At first, vertex v_2 could represent vertex v_3. However, if vertex v_2 is assigned the same color

as v_1, then v_3 can no longer be represented by v_2 in that integer coloring because v_1 would be the representative of v_3 in such a color. Thus, as vertices are being iteratively colored, the costs are updated to remove the term $-\lambda[u]$ for each vertex from R that can no longer represent u.

In each iteration of Algorithm 4, the color assigned to the vertex u with the lowest cost, say S, is the color with the smallest index that does not appear in the neighborhood of u, that is, no neighbor of u was previously assigned color S. If color S is assigned only to vertex u, than no cost need to be updated. Otherwise, suppose that r is the smallest vertex different from u assigned color S. The costs of vertices v selected from $R \cap \left(\bar{N}^+(u) \cup \bar{N}^+(r)\right)$ are updated as $c_\lambda[v] \leftarrow c_\lambda[v] + \lambda[v]$. There are two possible cases, namely:

- $r > u$: u becomes the S's representative and the selected vertices are all $v \in R \cap \bar{N}^+(u)$ with some neighbor with color S, and all $v \in R \cap \bar{N}^+(r)$.
- $r < u$: all $v \in R \cap \bar{N}^+(u)$ and all $v \in N(u) \cap R \cap \bar{N}^+(r)$ are selected.

This definition of costs is derived from the dualized constraints in the Lagrangian relaxation, thus it may provide significant information to find a good fractional coloring. Once the integer coloring determined, a representative is defined for each color as its smallest vertex. This representative is the vertex that represents the entire color in the fractional coloring.

In algorithm `FCPLagRel`, the order in which the vertices are evaluated in line 4 of Algorithm 1 is also determined by their costs. Thus, the same calculation and update of the costs are done at this point in the algorithm but considering $R = V$. In this case, we do not have the possibility of a vertex to be colored with a new color, because at this point we only color vertices that can extend existing colors.

Algorithm 4: `LagRelCol`– Integer coloring heuristic using the Lagrangian multipliers.

Data: Graph $G[R]$, Lagrangian multipliers vector λ.
Result: Integer coloring.

```
1 foreach u ∈ R do
2     Calculate initial costs c_λ[u]
3 repeat
4     Choose vertex u ∈ R with the lowest cost c_λ[u]
5     Assign the color with the smallest possible index to u
6     R ← R − u
7     Update cost c_λ[u], ∀u ∈ R
8 until R = ∅
9 return Coloring
```

An execution of two consecutive iterations (with indices $k = 1, 2$) of the heuristic `FCPLagRel` for the example illustrated in Figure 6 is summarized in Table 2. The Lagrangian multipliers used, as depicted in Figure 6a, correspond to the iteration of the outer loop of `2-PSA` that produced the best fractional coloring for the graph in Figure 2a. The table contains the indices of the iterations of the `LagRelCol` algorithm in the first column. The second column gives the vertex u with the lowest cost selected in each corresponding iteration of `LagRelCol`. The color S_j modified in this iteration appears in the fourth column, with index j in the preceding column. In the next columns, the resulting c_λ is indicated.

For iteration $k = 1$, the initial vertex costs are given in Figure 6b and are dependent on λ according to (21) with $R = V$. The resulting coloring for this iteration is shown in Figure 6c. A second color is assigned to some vertices in iteration $k = 2$, as presented in Figure 6d. The updated costs reflect the removal of vertices from R that received a second color. Subsequently, an integer coloring of $G[R]$ is obtained using the updated costs. These two iterations yield the 2-fold coloring shown in Figure 1, using a total of 6 colors with $\chi = 3$.

TABLE 2. Execution of two iterations of `FCPLagRel` on the graph of Figure 2a.

Iteration	u	j	S_j	c_λ										
$k = 1,\ R = V$														
1	11	1	{11}	−0.2	−0.5	−0.5	0.1	−0.8	0.6	0	0.2	0	0	–
2	10	1	{11,10}	−0.2	−0.5	−0.5	0.1	–	0.6	0	0.2	0	0	–
3	8	2	{8}	−0.2	−0.5	–	0.1	–	0.6	0	0.2	0	0	–
4	9	1	{11,10,9}	−0.2	–	–	0.1	–	0.6	0	0.2	0	0	–
5	4	2	{8,4}	–	–	–	0.1	–	0.6	0	0.2	0	0	–
6	1	3	{1}	–	–	–	0.1	–	0.6	0	0.2	–	0	–
7	6	3	{1,6}	–	–	–	0.1	–	0.6	–	0.2	–	0	–
8	7	3	{1,6,7}	–	–	–	0.1	–	0.6	–	0.2	–	–	–
9	3	4	{3}	–	–	–	–	–	0.6	–	0.2	–	–	–
10	5	2	{8,4,5}	–	–	–	–	–	0.6	–	–	–	–	–
11	2	2	{8,4,5,2}	–	–	–	–	–	–	–	–	–	–	–
$k = 2,\ R = \{8,3,10,2,6,5,1,7\}$														
1	10	5	{10}	–	–	−0.2	0.1	–	0.6	0	0.4	0	0	–
2	8	5	{10,8}	–	–	–	0.1	–	0.6	0	0.4	0	0	–
3	1	6	{1}	–	–	–	0.1	–	0.6	0	0.4	–	0	–
4	6	6	{1,6}	–	–	–	0.1	–	0.6	–	0.4	–	0	–
5	7	5	{10,8,7}	–	–	–	0.1	–	0.6	–	0.4	–	–	–
6	3	6	{1,6,3}	–	–	–	–	–	0.6	–	0.4	–	–	–
7	5	5	{10,8,7,5}	–	–	–	–	–	0.6	–	–	–	–	–
8	2	6	{1,6,3,2}	–	–	–	–	–	–	–	–	–	–	–
			Vertices	4	9	8	3	10	2	6	5	1	7	11

6. Computational experiments and results

This section describes computational experiments carried out to evaluate the effectiveness of `2-PSA` for FCNP. In particular, we compare the performance and efficiency of the Lagrangian heuristic with the greedy heuristic `FCP` discussed in Subsection 2.3.

6.1. Overall setup

We used `C++` as programming language and the optimization solver `CPLEX` 22.1.1 to solve the linear programming subproblems. These subproblems were solved in parallel using the library `pthreads` with 12 threads. The experiments were conducted on a computer with two Intel Xeon E5640 2.67 GHz processors (each with 6 cores), 20 GB of RAM, and a 64-bit Linux OS.

The parameters of `2-PSA` were set as follows: stepsize ψ starts at a value of 2 in the subgradient phase; $N_{\text{stuck}} = 10$; $S_\psi = 0.001$ (with a maximum limit of 200 iterations in the subgradient phase); the stepsize ψ was fixed at 1.5 in the heuristic phase; $N_{p2} = 100$. These values were chosen based on empirical experimentation. The initial value $\psi = 2$ was suggested by [27]. For each execution of the heuristic `FCPLagRel`, we imposed a limit of $N_{\text{lim}} = 1000$ colors. The integer programming formulation where (18) is replaced with $x^u_+ \in \mathcal{Y}(\bar{G}^+(u))$ of each subproblem $FC_u(\lambda)$ is tackled with cuts generated by `CPLEX` (with no branching).

The `CPLEX` parameters to generate cuts were set as follows: `CPX_PARAM_CLIQUES` = 1 (moderately clique cut generation); `CPX_PARAM_FRACCUTS` = 1 (moderately Gomory cut generation); `CPX_PARAM_ZEROHALFCUTS` = 1 (moderately 0–1/2 cut generation). These options were selected after performing several experiments, where we highlighted the cuts most used by `CPLEX` to solve the subproblems.

It is important to note that `CPLEX` uses information from previously solved subproblems (such as the basis of the linear system problem) to improve the resolution of subsequent subproblems. Therefore, the order in

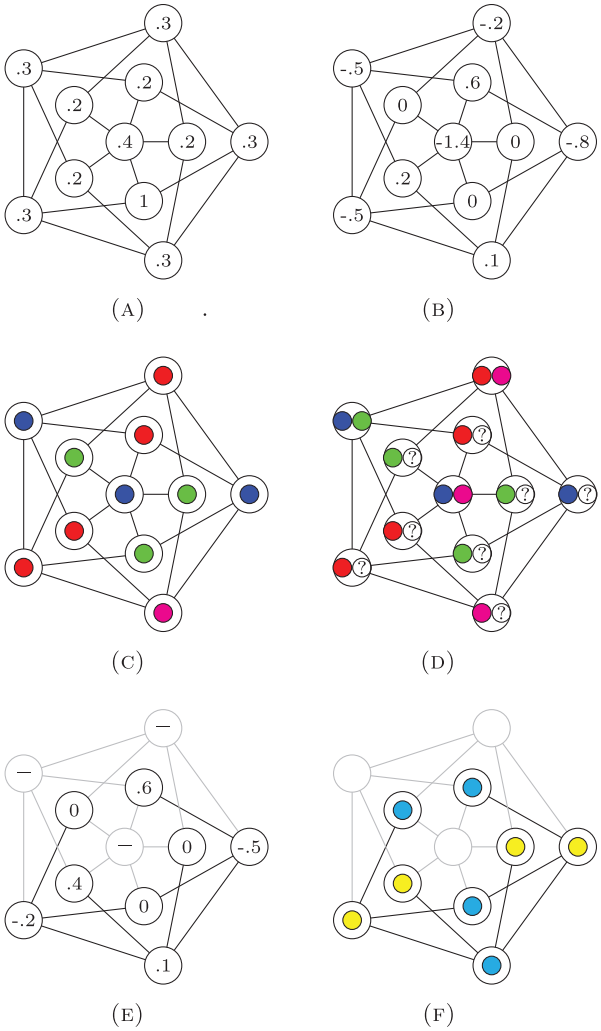


FIGURE 6. Execution of `FCPLagRel` during the outer iteration of `2-PSA` that produced the fractional coloring in Figure 1. (a) Vector λ, (b) vector c_λ from (21) for $R = V$, (c) a coloring with 4 colors, (d) extension with second color, (e) vector c_λ from (21) for $R = \{8, 3, 10, 2, 6, 5, 1, 7\}$ and (f) a coloring with 2 additional colors.

which the subproblems are solved by the threads in each iteration of the subgradient optimization may yield non-deterministic results. For this reason, we conducted 10 executions of `2-PSA` for each instance tested.

6.2. Instances

The majority of experiments were performed on non-random instances acquired from the DIMACS graph coloring challenge dataset, which is accessible at http://mat.gsia.cmu.edu/COLOR/instances.html. The instances were deliberately selected to be relatively challenging in terms of determining their fractional chromatic number [5, 16]. Table 3 displays the general features of these instances, such as the graph instance G (column *graph*), the quantity of vertices in G ($|V|$), the number of undirected edges in G ($|E|$), and G's density.

TABLE 3. Features of the instances for the FCP.

Graph	$\|V\|$	$\|E\|$	Density	χ_F
2-Insertions.3	37	144	11%	2.42
2-Insertions.4	149	541	4%	2.56
4-FullIns.3	114	541	8%	6.17
5-FullIns.3	154	792	6%	7.14ˆ
DSJC125.9	125	6961	90%	42.74
DSJC250.9	250	27897	90%	70.39
myciel4	23	71	28%	3.24
myciel5	47	236	21%	3.55
myciel6	95	755	17%	3.83
queen6.6	36	290	46%	7
queen8.8	64	728	36%	8.44
queen9.9	81	1056	32%	9
g200.10	200	17907	9%	–

Additionally, we present the fractional chromatic number χ_F, where the symbol ˆ denotes a lower bound for this value when the optimal solution is unknown.

We also tested 10 random graphs, denoted as g200.10, where the number 200 represents the number of vertices, and 10 denotes the probability of selecting an edge in the random graph generation. The rows corresponding to g200.10 in Tables 3 and 4 display the mean values of all information for these 10 random graphs. We chose this configuration of vertices and probability based on empirical experiments to create challenging instances for obtaining good feasible solutions.

6.3. Results and analysis

Table 4 presents experimental results. This table includes the graph instance G, the fractional chromatic number χ_F, the upper bound found by the `FCP` heuristic, the best lower bound found among all executions of the `2-PSA` algorithm (column BestLB), the average upper bound found among all executions of the `2-PSA` algorithm (AvgUB), the best upper bound found among all executions of the `2-PSA` algorithm (BestUB), the average number of iterations of the `2-PSA` algorithm (*Iter*), the improvement of the BestUB over `FCP`, calculated as

$$100\left(\frac{(FCP-\chi_F)-(BestUB-\chi_F)}{FCP-\chi_F}\right)=100\left(\frac{FCP-BestUB}{FCP-\chi_F}\right)$$

(column *Improv*, bold for values greater than 50%), and the average running time of the `2-PSA` algorithm in seconds (T (s)).

The results indicate that `2-PSA` outperforms the greedy heuristic `FCP` using `DSATUR` as the integer heuristic in terms of the solution's value. The average improvement of BestUB over `FCP` was 52%, indicating a substantial enhancement. For certain instances, optimal solutions were achieved by `2-PSA`, while for others, an improvement of at least 50% was attained, which is promising for low-density instances. However, we obtained a smaller improvement of less than 20% for instances DSJC250.9 and queen9.9. The non-determinism of the algorithm did not significantly affect the value of the solution. Another advantageous feature of the method proposed is that the value of the best lower bound found is very close to the optimal value. The average number of iterations of `2-PSA` was surprisingly low, ranging from 2 to 3, which implies that each iteration is very costly.

Although effective, the proposed method required significantly more running time compared to other heuristics such as `FCP`, which executed within negligible time for the instances tested. This motivated the use of the pricing technique, which greatly reduced the running time while keeping the quality of the solution value. For some instances, we achieved a reduction of over 30%. There was a very significant time reduction with the use

TABLE 4. Results of FCP and 2-PSA for the FCNP.

Graph	χ_F	FCP	Best LB	Avg UB	Best UB	*Iter*	*Improv*	T (s)
2-Insertions.3	2.42	2.75 (11/4)	2.42	2.50	2.50 (10/4)	2	**76.79%**	8
2-Insertions.4	2.56	3.33 (10/3)	2.43	2.79	2.75 (11/4)	2	**75%**	339
4-FullIns.3	6.17	6.40 (32/5)	6.08	6.17	6.17 (37/6)	2	**100%**	96
5-FullIns.3	7.14ˆ	7.20 (36/5)	7.08	7.19	7.17 (43/6)	2	**50%**	183
DSJC125.9	42.74	49.29 (345/7)	42.70	47.49	47.12 (754/16)	2	33%	21
DSJC250.9	70.39	83.71 (586/7)	70.33	82.08	81.87 (1228/15)	2	16%	117
myciel4	3.24	3.60 (18/5)	3.24	3.45	3.40 (17/5)	2	**56%**	3
myciel5	3.55	4.00 (36/9)	3.54	3.85	3.80 (38/10)	3	44 %	62
myciel6	3.83	4.80 (24/5)	3.56	4.38	4.29 (30/7)	2	**53%**	717
queen6.6	7	7.67 (23/3)	6.99	7.29	7.00 (7002/1000)	2	**100%**	7
queen8.8	8.44	10.75 (43/4)	8.29	9.71	9.67 (58/6)	2	47 %	99
queen9.9	9	11.17 (67/6)	8.99	10.82	10.75 (86/8)	2	19%	126
g200.10	–	71.56 (514/7)	60.56	69.23	69.09 (886/12)	3	22%	61
Avg Num of Colors	–	(237/6)	–	–	(860/71)	–	52%	141

of parallelism, which was 70% on average using 12 threads. It indicates an even better running time reduction if more threads are used since there are as many independent subproblems as the number of vertices in G. This is a powerful scalability resource of our technique. Furthermore, the difference between the best upper bound and the lower bound found by 2-PSA along with the running time of the method reflects the difficulty of the problem. The easiest instances for the problem are generally the ones with low or high densities.

The use of heuristic FCPLagRel for FCNP brings the advantage of a solution with few colors compared to a solution of the formulation (10)–(13), whose value of k is usually very large. This characteristic influences applications of associated coloring problems, like *RWP* [20], where a solution with many colors may become impractical.

7. Concluding remarks

We have developed and implemented a Lagrangian heuristic called 2-PSA for the Fractional Chromatic Number Problem (FCNP). This heuristic is used within a 2-phase subgradient method to obtain a feasible solution for the problem. Our approach is based on a Lagrangian relaxation of a representatives formulation that has two interesting characteristics. The first characteristic is that the relaxed problem can be decomposed into independent subproblems and, therefore, can be solved in parallel, which significantly reduces the running time. The second feature is that these subproblems contain stable sets polyhedron constraints, which allows us to use already known techniques for separating several of its facets. We explore these two characteristics in 2-PSA.

We presented an analysis of the computational results of our method by comparing it with the greedy heuristic called FCP. We obtained better results at the expense of longer running time. Our Lagrangian heuristic provides a good improvement in the solution value compared to the heuristic FCP, and excellent lower bounds. Although the running time of 2-PSA is relatively high when considering it just as a heuristic, unlike other methods used to solve FCNP, 2-PSA gives us not only a good feasible solution but also a lower bound in each iteration of the method.

Practical applications involving FCNP require the value of the optimal solution as well as the coloring itself. Therefore, it is preferable to use 2-PSA rather than a method such as the column generation approach. Moreover, the Lagrangian heuristic used in 2-PSA generates a solution with far fewer colors compared to a solution provided by known formulations for FCNP. This significantly improves the feasibility of the solution in practical applications.

For future work, we consider developing heuristics and specific separation procedures for the maximum weighted stable set subproblems to accelerate 2-PSA. An analysis of the information provided by subproblems

solved in previous iterations of `2-PSA` is also interesting since these subproblems differ only in the objective function.

In our approach to solve FCNP, we can identify several points that can still be improved. For example, the method used to find Lagrangian multipliers with good dual bounds can be improved. Other methods, such as bundle, space expansion, and subgradient projection, can be tried in addition to the subgradient method. Moreover, for the Lagrangian heuristic, there may be better parameter settings that generate better solutions. Since the number of configuration options is enormous, we do not guarantee that we used the best possible configuration for the Lagrangian heuristic, despite satisfactory results and extensive experiments to choose our parameter values.

Acknowledgements. Partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and the Brazilian agencies CNPq (Proc. 312417/2022-5, Proc. 422912/2021-2) and FUNCAP (PNE – 0112-00061.01.00/16, PS1 – 0186-00155.01.00/21).

References

[1] E. Balas and J. Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica* **15** (1996) 397–412.
[2] D. Brélaz, New methods to color the vertices of a graph. *Commun. ACM* **22** (1979) 251–256.
[3] M. Campêlo and R.C. Corrêa, A combined parallel lagrangian decomposition and cutting-plane generation for maximum stable set problems. *Elec. Notes Discrete Math.* **36** (2010) 503–510.
[4] M. Campêlo, V.A. Campos and R.C. Corrêa, On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Appl. Math.* **156** (2008) 1097–1111.
[5] M. Campêlo, V.A. Campos and R.C. Corrêa, Um algoritmo de planos-de-corte para o número cromático fracionário de um grafo. *Pesqui. Operacional* **29** (2009) 179–193.
[6] A. Caprara, M. Fischetti and P. Toth, A heuristic method for the set covering problem. *Oper. Res.* **47** (1995) 730–743.
[7] F.C. Chow and J.L. Hennessy, The priority-based coloring approach to register allocation. *ACM Trans. Program. Lang. Syst.* **12** (1990) 501–536.
[8] F. Clarke and R. Jamison, Multicolorings, measures and games on graphs. *Discrete Math.* **14** (1976) 241–245.
[9] D. de Werra, An introduction to timetabling. *Eur. J. Oper. Res.* **19** (1985) 151–162.
[10] Z. Dvořák and X. Hu, Fractional coloring of planar graphs of girth five. *SIAM J. Discrete Math.* **34** (2020) 538–555.
[11] Z. Dvořák, J.-S. Sereni and J. Volec, Subcubic triangle-free graphs have fractional chromatic number at most 14/5. *J. London Math. Soc.* **89** (2014) 641–662.
[12] A. Gamst, Some lower bounds for a class of frequency assignment problems. *IEEE Trans. Veh. Technol.* **35** (1986) 8–14.
[13] M.R. Garey and D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness. W.H. Freeman & Co., New York, NY, USA (1990).
[14] J. Gimbel, A. Kündgen, B. Li and C. Thomassen, Fractional coloring methods with applications to degenerate graphs and graphs on surfaces. *SIAM J. Discrete Math.* **33** (2019) 1415–1430.
[15] A.M. Gleixner, D.E. Steffy and K. Wolter, Iterative refinement for linear programming. *INFORMS J. Comput.* **28** (2016) 449–464.
[16] N. Gvozdenovic, *Approximating the stability number and the chromatic number of a graph via semidefinite programming*. Ph.D. thesis, Faculty of Science (2008).
[17] P. Hell and F. Roberts, Analogues of the shannon capacity of a graph. *Ann. Discrete Math.* **12** (1982) 155–168.
[18] A. Hilton, R. Rado and S. Scott, A (≤ 5)-color theorem for planar graphs. *Bull. London Math. Soc.* **5** (1973) 302–306.
[19] R.v.d. Hulst, *A branch-price-and-cut algorithm for graph coloring*. Master's thesis, University of Twente, The Netherlands (2021).
[20] R. Klasing, N. Morales and S. Pérennes, On the complexity of bandwidth allocation in radio networks. *Theor. Comput. Sci.* **406** (2008) 225–239.
[21] M. Larsen, J. Propp and D. Ullman, The fractional chromatic number of mycielski's graphs. *J. Graph Theory* **19** (1995) 411–416.
[22] A.N. Letchford and P. Ventura, Strengthened clique-family inequalities for the stable set polytope. *Oper. Res. Lett.* **49** (2021) 586–589.
[23] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems. *J. ACM* **41** (1994) 960–981.
[24] A. Mehrotra and M.A. Trick, A column generation approach for graph coloring. *Informs J. Comput.* **8** (1996) 344–354.
[25] F. Pirot and J.-S. Sereni, Fractional chromatic number, maximum degree, and girth. *SIAM J. Discrete Math.* **35** (2021) 2815–2843.
[26] S. Rebennack, M. Oswald, D.O. Theis, H. Seitz, G. Reinelt and P.M. Pardalos, A branch and cut solver for the maximum stable set problem. *J. Comb. Optim.* **21** (2011) 434–457.

[27] C.R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press, New York (1993).
[28] F. Rossi and S. Smriglio, A branch-and-cut algorithm for the maximum cardinality stable set problem. *Oper. Res. Lett.* **28** (2001) 63–74.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Second edition. Society for Industrial and Applied Mathematics (2003).