CARPATHIAN J. MATH.
Volume **41** (2025), No. 2,
Pages 465-477

Online version at https://www.carpathian.cunbm.utcluj.ro/
ISSN 1584-2851 (Print edition); ISSN 1843-4401 (Electronic)
DOI: https://doi.org/10.37193/CJM.2025.02.13

A set packing model for the Partition Coloring Problem

EMANUEL FLORENTIN OLARIU AND CRISTIAN FRĂSINARU

ABSTRACT. In this paper we propose a set packing model for the Partition Coloring Problem (PCP) a generalization of the Vertex Coloring Problem (VCP). Given a graph whose vertex set is partitioned in clusters, PCP aims to select one vertex from each cluster such that the chromatic number of the resulted induced subgraph is minimal.

A set packing integer linear programming formulation is presented and, based on this, we implement a branch-and-price algorithm. The resulted formulation offers a very good quality root linear relaxation problem. Computational experiments led on instances from literature and on newly generated instances for the problem of routing and wavelength assignment in all-optical networks show that our algorithm performs at least as well and sometimes better than the state-of-the-art algorithms for large instances. We introduce a version of this algorithm based on strengthening the root linear relaxation with cuts using two families of valid inequalities which proved to be effective for small and medium instances.

1. INTRODUCTION

One of the most studied optimization problem in graph theory is vertex coloring problem (VCP): for a given graph we want to assign colors to its vertices in such a way that any two adjacent vertices receive different colors. This problem has a wide range of applications: scheduling, frequency allocation, register allocation, wavelength routing etc; for these applications and some approaches see [11, 12, 18, 21, 23, 29].

Motivated in the first place by the routing and wavelength assignment (RWA) in optical networks ([19, 24]), one of the many generalizations of VCP is the *partition coloring problem* (PCP), also known as the *selective coloring problem*. The decision variant of PCP is a NP-hard problem since it generalizes VCP decision counterpart which is known to be NP-complete. Besides RWA there are some other real life applications of PCP: dichotomy-based constraint encoding, (antenna positioning and) frequency assignment problem, quality test scheduling, berth allocation problem etc ([5]).

VCP and its variants are computationally challenging and different linear programming approaches were developed to solve these problems ([21]). Some of these models use compact formulations, like the assignment formulation, the representative formulation ([2]) or the partial-ordering based formulations ([17]); other models use an exponential number of variables, like the set covering formulation from [22], which requires the use of the column generation method (see [6]).

2. BACKGROUND

Let $G = (V, E)$ be a simple undirected graph with n vertices and $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ be a partition of its vertices, i. e., $V_i \cap V_j = \emptyset$, $\forall 1 \leq i < j \leq k$, $\bigcup_{i=1}^{k} V_i = V$, and $V_i \neq \emptyset$,

Received: 12.04.2024. In revised form: 06.11.2024. Accepted: 13.11.2024
2010 *Mathematics Subject Classification.* 90C10, 90C57.
Key words and phrases. *partition coloring, integer linear programming, set packing model, branch-and-price, cutting planes.*
Corresponding author: Emanuel F. Olariu; emanuel.olariu@info.uaic.ro

$\forall i \in [k]$, where, for a given $q \in \mathbf{N}^*$, $[q] = \{1, 2, \ldots, q\}$. The classes of this partition will be called clusters.

Definition 2.1. *A selective p-coloring for the pair $(G, \mathcal{P})$ is a p-coloring of the subgraph induced by a subset of vertices $V' \subseteq V$ that contains exactly one vertex from each cluster of $\mathcal{P}$.*

Formally, a selective p-coloring is a family of p pairwise disjoint stable (vertex independent) sets $S_1, S_2, \ldots, S_p$ of G such that $|V_i \cap S_j| \leq 1$, $\forall i \in [k]$, $\forall j \in [p]$, and each cluster intersects at least a stable set.

The aim of the *Partition Coloring Problem* (*PCP*, also called the *Selective Coloring Problem*) is to find the minimum p for which G has a selective p-coloring, that is the *partition* (or *selective*) *chromatic number*.

The first approaches to PCP proposed RWA related heuristics ([19]); a tabu search heuristic was also proposed in [24], a memetic algorithm in [27] and an ant-local search algorithm in [8]. Another approach based on reducing the scale of the underlying graph with good results for large graphs can be found in [30]. Very good results were obtained by using Integer Linear Programming (ILP) models: a representative formulation in [9], a combined model of the asymmetric representative formulation and the independent set model in [16], a set covering model in [10]. These last ILP approaches describe exact algorithms for solving PCP.

The complexity status of the PCP was investigated in [3] and [4] for some particular classes of graphs and in [13] for different fixed parameters (the number of colors, the number of clusters and the maximum cardinality of a cluster).

2.1. **Contributions and paper structure.** By taking advantage on the work already done we introduce a simple set packing ILP model based on a similar model for the VCP ([14]) which needs a pricing phase built around a pricing problem corresponding to the Maximum Weight Stable Set (MWSS) problem in a certain graph. In the initialization step we use the slack variables and the heuristic described in [16]; we later solve a sequence of MWSSs problems in order to tackle the restricted master problem. By employing a smart way of modifying and restoring the variables already introduced to the parent node in the branch-and-bound tree, we significantly speed up the computation.

We introduce also a variant of our branch-and-price algorithm by adding to the linear relaxation of the root problem two families of cuts based on maximal cliques in the corresponding intersection graph. This variant of the algorithm was tested on some small and medium sized instances and it is aimed at improving the lower bound of the objective function of the root problem.

The benchmark test instances are partly those from `www.ic.uff.br/celso/grupo/pcp.htm` and partly were generated from the RWA context (the so called ring network topologies).

The numerical results prove that our algorithms perform at least as well and sometimes better than the state-of-the-art algorithms from the literature (see [16, 10]) and are effective for very large instances (in terms of edge density and number of vertices and/or edges) and small and medium instances, respectively.

The remaining of the paper is organized as follows: in section 3 we discuss the ILP set packing model, in section 4 we describe the branch-and-price-algorithm and the derived variant which adds cuts to the root linear relaxation, the numerical results and the implementation details are discussed in section 5. Finally, section 6 is dedicated to conclusions.

3. ILP MODELS

Let $G = (V, E)$ be a with n vertices and $\mathcal{P} = \{V_1, V_2, \ldots, V_k\}$ be a partition of its vertices. The classical integer linear programming assignment formulation for the graph coloring problem ([28]) can be modified for modeling the Partition Coloring Problem ([9, 16]): we need two sets of binary variables: x_{vi}, with $v \in V$ and $i \in [k]$, $x_{vi} = 1$ if and only if vertex v receives the color i, and w_i with $i \in [k]$, where $w_i = 1$ if and only if color i is used.

$$(Assign) \quad min \sum_{j=1}^{k} w_j$$

$$\sum_{j=1}^{k} \sum_{v \in V_i} x_{vj} = 1, \forall i \in [k] \tag{3.1}$$

$$x_{uj} + x_{vj} \leq w_j, \forall uv \in E, \forall j \in [k] \tag{3.2}$$

$$x_{vj} \in \{0, 1\} \ \forall v \in V, \forall j \in [k], w_j \in \{0, 1\}, \forall j \in [k]$$

Let $\mathcal{S}$ be the family of all stable sets of G intersecting each cluster in at most one vertex:

$$\mathcal{S} = \{S \subseteq V \ : \ xy \notin E, \forall x, y \in S, |S \cap V_i| \leq 1, \forall i \in [k]\}$$

The following set partitioning formulation can be obtained by using the Dantzig-Wolfe decomposition (by convexification of (3.1) - [7, 10]).

$$(SetPart) \quad min \sum_{S \in \mathcal{S}} x_S$$

$$\sum_{S \in \mathcal{S}: S \cap V_i \neq \emptyset} x_S = 1, \forall i \in [k], \tag{3.3}$$

$$x_S \in \{0, 1\} \ \forall S \in \mathcal{S}$$

Now we will separate the variables corresponding to stable sets of cardinality 1 ([14]):

$$\sum_{v \in V_i} x_{\{v\}} = 1 - \sum_{S \in \mathcal{S}_2^i} x_S \geq 0, \forall i \in [k], \tag{3.4}$$

where $\mathcal{S}_2^i = \{S \in \mathcal{S} \ : \ S \cap V_i \neq \emptyset, |S| \geq 2\}$. After transforming the objective function we get the following set packing model of the problem:

$$(SetPack) \quad max \sum_{S \in \mathcal{S}_2} (|S| - 1) x_S$$

$$\sum_{S \in \mathcal{S}_2^i} x_S \leq 1, \forall i \in [k], \tag{3.5}$$

$$x_S \in \{0, 1\}, \ \forall S \in \mathcal{S}_2$$

where $\mathcal{S}_2 = \bigcup_{i=1}^{k} \mathcal{S}_2^i = \{S \in \mathcal{S} \ : \ |S| \geq 2\}$.

Consider the linear relaxations *(AssignRL)*, *(SetPartLR)*, and *(SetPackLR)* of the above ILP problems and z^0, z^1, and z^2 their corresponding optimal values. The following result is straightforward and we give it just for the sake of completeness.

Proposition 3.1. $z^0 \leq z^1 = k - z^2$.

Proof. The first relation easily follows since *(SetPart)* is obtained by Dantzig-Wolfe decomposition from *(Assign)*. For the second relation let $\mathrm{x} = (x_S)_{S\in\mathcal{S}_2}$ be an optimal solutions for *(SetPackLR)*.

Define first $\tilde{x}_S = x_S, \forall S \in \mathcal{S}_2$; second, for any cluster V_i for which $\sum_{S\in\mathcal{S}_2^i} x_S = 0$, we choose a vertex $v_i \in V_i$ and define $\tilde{x}_{\{v_i\}} = 1$, $\tilde{x}_{\{v\}} = 0$, $\forall v \in V_i \setminus \{v_i\}$; third, for any remaining cluster V_h, define $\tilde{x}_{\{v\}} = 0, \forall v \in V_h$. We get

$$\sum_{v\in V_i} \tilde{x}_{\{v\}} = 1 - \sum_{S\in\mathcal{S}_2^i} \tilde{x}_S, \forall 1 \le i \le k, \tag{3.6}$$

hence, $\tilde{\mathrm{x}} = (\tilde{x}_S)_{S\in\mathcal{S}}$ is a feasible solution for *(SetPartLR)*. On the other hand, by adding all relations (3.6)

$$z^1 \le \sum_{S\in\mathcal{S}} \tilde{x}_S = \sum_{v\in V} \tilde{x}_{\{v\}} + \sum_{S\in\mathcal{S}_2} \tilde{x}_S = k - \sum_{S\in\mathcal{S}_2} |S|\cdot\tilde{x}_S + \sum_{S\in\mathcal{S}_2} \tilde{x}_S =$$

$$= k - \sum_{S\in\mathcal{S}_2} (|S|-1)\cdot\tilde{x}_S = k - \sum_{S\in\mathcal{S}_2} (|S|-1)\cdot x_S = z^2.$$

The second inequality (i. e., $z^1 \ge z^2$) can be obtained by observing that the development of *(SetPackLR)* from *(SetPartLR)* allows to define a feasible solution to the former problem by restricting to $\mathcal{S}_2$ any solution for the latter. □

The above result says that the lower bounds for the partition chromatic number obtained by *(SetPartLR)* and *(SetPackLR)* could be better than that obtained using *(AssignLR)*. But both set partitioning and set packing models could have a huge number of variables even for medium-sized instances, hence we need to employ the *column generation* (*CG*) method ([6, 20]) for solving *(SetPackLR)*. The corresponding branch-and-price algorithm will be presented in the next section.

The set packing model has the advantage that the polyhedral theory developed around this formulation gives some easy to implement facet inducing inequalities of the set packing polyhedron, that is the convex envelope of the solutions to *(SetPack)*. Using such cuts we implemented a variant of the branch-and-price algorithm by strengthening the root linear relaxation.

4. Branch & Price algorithm

4.1. **The column generation.** The dual of *(SetPackLR)* (which is also in canonical form) is

$$\begin{aligned} (DSetPackLR)\ \ min\ & \left(\sum_{i=1}^{k}\sigma_i\right) \\ & \sum_{i:S\in\mathcal{S}_2^i} \sigma_i \ge |S|-1, \forall S\in\mathcal{S}_2, \\ & \sigma_i \ge 0, \forall i \in [k]. \end{aligned} \tag{4.7}$$

The corresponding Restricted Master Problem (*RMP*) is the problem *(SetPackLR)* over a subset of variables containing a feasible solution. The initial basic feasible solution is the

set of slack variables of the constraints (3.5); hence RMP is

$$
\begin{aligned}
(SetPackRMP)\quad max\ & \sum_{S\in\mathcal{S}_2'}(|S|-1)x_S \\
& \sum_{S\in\mathcal{S}_2'^{i}} x_S + s_i = 1, \forall i\in[k], \qquad (4.8)\\
& x_S \geq 0,\ \forall S\in\mathcal{S}_2', s_i\geq 0, \forall i\in[k].
\end{aligned}
$$

where $\mathcal{S}_2'^{i}\subseteq\mathcal{S}_2^i, \forall i\in[k]$ and $\mathcal{S}_2'=\bigcup_{i=1}^{k}\mathcal{S}_2'^{i}$.

The associated subproblem asks to find a stable set $S\in\mathcal{S}_2$ for which the corresponding dual constraint is violated:

$$(|S|-1)-\sum_{i:S\in\mathcal{S}_2^i}\sigma_i>0.$$

An integer linear program for finding such a stable set (provided that the optimal value is strictly less than -1) is:

$$
\begin{aligned}
(SubPSetPack)\quad min\ & \left[\sum_{v\in V}(\sigma_{i(v)}-1)x_v\right] \\
& x_u+x_v\leq 1, \forall uv\in E, \qquad (4.9)\\
& \sum_{v\in V_i} x_v\leq 1, \forall i\in[k], \qquad (4.10)\\
& \sum_{v\in V} x_v\geq 2, \qquad (4.11)\\
& x_v\in\{0,1\}, \forall v\in V,
\end{aligned}
$$

where $i(v)$ is the index i such that $v\in V_i$. This problem is that of finding a maximum weight stable set (MWSS) in the graph obtained from G by adding all possible edges between vertices in each $[V_i]_G$; $(x_v)_{v\in V}$ is the characteristic vector of such a stable set $S\in\mathcal{S}_2$. We add such variables x_S to *(SetPackRMP)* until the optimum value in *(SubPSetPack)* becomes at least -1.

Note that by choosing the set packing model and transforming the objective function (from min to max) the variables from the dual problems are non-negative but the weights could be negative, therefore we cannot employ the usual way of addressing MWSS problem (see [15]), instead we choose to solve the subproblem by the using the MILP available solver.

4.2. **The branching rule.** The details about the branch-and-price algorithm will be completed with the branching rule. For each $i,j\in[k]$, $i\neq j$ we define

$$\alpha_{ij}=\sum_{S\in\mathcal{S}_2'^{i}\cap\mathcal{S}_2'^{j}} x_S.$$

Proposition 4.2. *Let* $\mathrm{x}=(x_S)_{S\in\mathcal{S}_2'}$ *be a basic feasible optimal solution to RMP which is fractional, then there exist two distinct clusters* V_i *and* V_j *such that* $\alpha_{ij}\in(0,1)$.

Proof. We will show first that there exists a cluster V_i and two different stable sets $S_1,S_2\in\mathcal{S}_2'^{i}$ such that $x_{S_1},x_{S_2}\in(0,1)$. Suppose on the contrary that for each $i\in[k]$ there exists at

most one stable set $S \in S_2^{'i}$ such that $x_S \neq 0$, then the family $\{S \in \mathcal{S}_2^{'} \ : \ x_S \neq 0\}$ is formed with pairwise disjoint stable sets, hence the following solution is better than x:

$$\tilde{\mathrm{x}} = (\tilde{x}_S)_{S \in \mathcal{S}_2'}, \text{ where } \tilde{x}_S = \begin{cases} 0, & \text{if } x_S = 0 \\ 1, & \text{if } x_S \neq 0 \end{cases}, \forall S \in \mathcal{S}_2^{'},$$

which is a contradiction since x is already optimal. Let now V_i be a cluster such that there exist two stable sets $S_1, S_2 \in \mathcal{S}_2^{'i}$ with $x_{S_1}, x_{S_2} \in (0,1)$. Since the basis corresponding to x cannot contain two identical columns, there exists another cluster V_j, $j \neq i$, such that the corresponding constraint from 4.8 contains x_{S_1} or x_{S_2} but not both ([1]). Therefore

$$1 \geq \sum_{S \in \mathcal{S}_2^{'i}} x_S > \sum_{S \in \mathcal{S}_2^{'i} \cap \mathcal{S}_2^{'j}} x_S = \alpha_{ij} > 0.$$

□

Using the above result we can choose $i_0, j_0 \in [k]$, $i_0 \neq j_0$, such that

$$\alpha_{i_0 j_0} = \max\{\alpha_{ij} \ : \ \alpha_{ij} \in (0,1), i, j \in [k], i \neq j\}.$$

Then, we choose $S_0 \in S_2^{'i_0} \cap \mathcal{S}_2^{'j_0}$ such that $x_{S_0} = \max_{S \in S_2^{'i_0} \cap \mathcal{S}_2^{'j_0}} x_S$. Let us denote $\{v_{i_0}\} = S_0 \cap V_{i_0}$, $\{v_{j_0}\} = S_0 \cap V_{j_0}$; we then branch on this pair of vertices (v_{i_0}, v_{j_0}). In the first child node we require that the two vertices have the same color, while in the second we force the two vertices to have different colors. For the first branch we replace $V_{i_0} \cup V_{j_0}$ with a new cluster $V_{i_0 j_0}$ containing a single new vertex $v_{i_0 j_0}$ with the neighborhood $(N_G(v_{i_0}) \cup N_G(v_{j_0})) \setminus (V_{i_0} \cup V_{j_0})$. For the second branch we simply add a new edge between v_{i_0} and v_{j_0}.

4.3. **A variant of the main Branch & Price algorithm.** The lower bound obtained by solving *(setPackRMP)* can be improved by adding cuts like those described in [25, 26] or [14] for VCP. Denote by C the set of feasible solutions to the ILP counterpart of *(setPackRMP)* obtained by adding integral constrainst to all of its variables) and by $P^I = conv(C)$; then $P^I \subseteq P$, where P is the polyhedron (in fact is a polytope) of feasible solutions to *(setPackRMP)*.

Optimizing a linear objective function over C means in fact to optimize over P^I whose extreme points are among the extreme points of P. But finding perfect formulations for P^I was proved to be a very difficult task; thus, instead we can search for a sandwiched polyhedron $P^I \subseteq P' \subseteq P$, obtained by adding to the formulation of P inequalities which are valid to P^I.

The associated *intersection graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ([25]) is defined like this: $\mathcal{V} = \mathcal{S}_2'$, and two such variables S' and S'' are adjacent in $\mathcal{G}$ if their corresponding columns are not orthogonal; in other words $S'S'' \in \mathcal{E}$ if and only if S' and S'' intersect the same cluster of $\mathcal{P}$. By keeping the same objective function and replacing the constraints matrix in *(setPackRMP)* with the edge-vertex incidence matrix of G we get an equivalent problem *(setPackRMPG)*. The convex envelope of the integer feasible solutions for this problem, denoted by $P_{\mathcal{G}}^I$, equals P^I.

Theorem 4.1 ([25]). *Let $\pi \in \{0,1\}^{|\mathcal{V}|}$; an inequality $\pi \mathrm{x} \leq 1$ is facet defining for P^I if and only if the set $\{S \in \mathcal{V} \ : \ \pi_S \neq 0\}$ is a maximal clique of $\mathcal{G}$.*

There are similar results involving odd cycles and their complements or web graphs - but the maximal cliques of $\mathcal{G}$ are the more convenient way for producing facet defining inequalities because of the computational effort associated with the former subgraphs ([26]).

We add to the root problem of the branch-and-price tree two types of clique-based facet defining inequalities. The first type is generated starting with three stable sets which don't intersect the same cluster but induce a clique in $\mathcal{G}$ and, then, extending this clique to a maximal one. The second type is related to the so called *majority cliques* from [14] and adapted here for the PCP context.

Definition 4.2. *Let $I \subseteq [k]$, $|I| = p$ and define $I_S = \{i \in I \ : \ S \in \mathcal{S}_2^{'i}\}$, for all $S \in \mathcal{S}_2'$. A stability witness for I is a $S_0 \in \mathcal{S}_2'$ such that $I_{S_0} = I$ and each $S_0' \subseteq S_0$ with $|S_0'| \geq (p+1)/2$ belongs to $\mathcal{S}_2'$.*

Proposition 4.3. *Let $I \subseteq [k]$, $|I| = p \geq 3$. If p is odd and I has a stability witness, then $\mathcal{Q}_I = \{S \in \mathcal{S}_2' \ : \ |I_S \cap I| \geq (p+1)/2\}$ is a maximal clique in $\mathcal{G}$.*

Proof. Let S_0 be a stability witness for I. Obviously $\mathcal{Q}_I$ is a clique since if $S_1, S_2 \in \mathcal{Q}_I$, then there exists a cluster V_i with $i \in I_{S_1} \cap I_{S_2}$.

Now, if $S \notin \mathcal{Q}_I$, then $|I_S \cap I| \leq (p-1)/2$; define $S' = S_0 \setminus \{u \in S_0 \ : \ u \in V_i \text{ and } i \in I_S\}$. Thus, $I_{S'} = I_{S_0} \setminus I_S$ and $|I_{S'}| = |I_{S_0}| - |I_S \cap I| \geq p - (p-1)/2 = (p+1)/2$. Hence $S' \in \mathcal{Q}_I$ and $SS' \notin \mathcal{E}$; therefore Q_I is maximal. □

Obviously, if we replace $\mathcal{S}_2'$ with $\mathcal{S}_2$ the property of the witness that each $S_0' \subseteq S_0$ with $|S_0'| \geq (p+1)/2$ belongs to $\mathcal{S}_2'$ is not longer necessary in the above proof.

Note that these two classes of cuts described above are added to the linear relaxation problems in order to get a better lower bound for the optimum of the corresponding ILP problems, and afterwards they are withdrawn to allow the branch-and-price-algorithm to work the usual way.

By adding the constraints that follow to *(SetPack)* we get *(SetPackCuts)* problem

$$\sum_{S \in \mathcal{S}_2^f} x_S \leq 1, \forall f \in [F], \tag{4.12}$$

where (4.12) is the constraint corresponding to the f-th cut from the set of F employed cuts and $\mathcal{S}_2^f$ is the corresponding maximal clique.

These cuts are added only to the root problem linear relaxation of the branch-and-bound tree; we make this choice because of the computational burden associated with reusing/restoring (see subsection 5.2 below) the variables for all related constraints.

5. Implementation details and numerical experiments

Our computational experiments were carried on a MacBook Pro M1 with 8 GB of memory on macOS Monterey and using a Gurobi Academic License. Some of the benchmark instances are available at: `http://www2.ic.uff.br/celso/grupo/pcp.htm` - from which we considered only a subset - those which occurred to be the most difficult for the state-of-the-art algorithms ([16, 10]). We deliberately omitted the results for the smaller instances because they are usually very quickly solved by our algorithm. Instead we created other larger random ring type instances (arising from the RWA problem) following the procedure described in [9]. A public repository containing our Java project is here: `https://github.com/fe-olariu/PCP` and the instances are here: `https://github.com/fe-olariu/data/tree/main/PartitionCol`.

5.1. **Initialization.** The initial feasible basis for the RMP is formed with the slack variables, but in order to accelerate the resolution we add a subset of variables corresponding to stable sets in $\mathcal{S}_2$ obtained by perturbing (see [16]) an initial solution found using an heuristic developed in [24]. After adding all these variables we start to sequentially solve the subproblem until the reduced costs of all not added variables are non-negative.

Each subproblem is solved using the ILP model described by $(subPSetPack)$. To this end we employed the Gurobi MILP Solver with a parameterized number of solutions to store; thus, after each solve of a sub-problem we add a number of best solutions (stable sets) to RMP, this parameter was tuned to be 15, in Gurobi: *PoolSearchMode* $= 2$, *PoolSolutions* $= 15$. Extensive numerical experiments shown that larger values for *PoolSolutions* drastically increase the solving time, while very small values of this parameter have the same effect by increasing the number of needed subproblems.

5.2. **Reusing and restoring variables.** After calling the branching rule two new problems are created and added to the stack. Instead of using the same process from above for solving the corresponding new RMPs, we choose first to reuse the variables belonging to the parent problem; the result is a speed-up of RMP solve mainly because of the number of sub-problems which sharply decreases.

The way in which we reuse the variables depends on the type of the problem. If the problem is obtained by contracting the pair $v_{i_0}v_{j_0}$, then we keep or modify the following stable sets:

- containing both concerned vertices which are replaced by the new vertex $v_{i_0j_0}$;
- containing none of them but after deleting the vertices from their clusters - if any;
- containing only one of the two concerned vertices (replaced by $v_{i_0j_0}$) but after deleting the vertices which are adjacent with the other vertex or belong to the same cluster.

If the problem is obtained by adding the edge $v_{i_0}v_{j_0}$, then from each stable set (of the parent node) containing both v_{i_0} and v_{j_0} we get the following stable sets:

- a stable set containing only one of the two vertices after removing the other one;
- a stable set containing none of them after removing both concerned vertices.

During this process we ensure that for each stable set there exists only one variable in the corresponding problem.

5.3. **Branch & Price algorithm results.** Tables 1 and 2 contain the results of the branch-and-price algorithm. For both series of experiments the time limit was set to 1800 seconds; only for one instance (the largest in terms of number of edges) we allow a larger amount of time.

TABLE 1. Numerical results for known ring instances.

instance	n	m	lb	ub	lb^r	$\overline{ub}$	N	$rVar$	$nVar$	rSb	nSb	rT	nT	T
ring_n20p0.6s1	458	42,316	36	36	36	*36*	165	3,209	21.12	11	1.10	9.12	0.08	98.09
ring_n20p0.6s2	464	41,107	36	36	36	*36*	153	2,250	17.76	16	1.29	11.78	0.66	113.54
ring_n20p0.6s3	456	41,906	32	32	32	*32*	168	2,364	17.65	28	1.23	21.31	0.57	117.46
ring_n20p0.6s4	452	41,530	32	32	32	*32*	152	3,089	29.67	34	1.61	23.29	0.68	127.27
ring_n20p0.6s5	440	39,445	34	34	34	*34*	158	2.047	15.97	12	1.27	8.16	0.52	90.79
ring_n20p0.7s1	536	58,142	39	39	39	*39*	194	3,136	18.50	20	1.15	20.87	0.83	182.17
ring_n20p0.7s2	580	68,658	43	43	43	*43*	204	3,354	19.18	16	1.17	18.40	1.00	222.37
ring_n20p0.7s3	534	57,579	37	37	37	*37*	199	2,629	19.26	29	1.39	28.15	0.81	189.23
ring_n20p0.7s4	536	58,373	38	38	38	*38*	204	2,809	18.68	29	1.35	28.57	0.83	197.63
ring_n20p0.7s5	518	54,784	38	38	38	*38*	187	2,863	20.31	41	1.32	40.98	0.86	201.42
ring_n20p0.8s1	614	76,568	44	44	44	*44*	223	4,398	23.72	25	1.26	35.31	1.20	303.21
ring_n20p0.8s2	624	79,501	46	46	46	*46*	237	4,049	19.11	14	1.13	12.80	0.71	181.58
ring_n20p0.8s3	614	76,290	43	43	43	*43*	234	3,020	16.30	22	1.22	18.34	0.64	167.78
ring_n20p0.8s4	610	75,741	43	43	43	*43*	215	3,674	22.97	34	1.38	46.50	0.86	231.63
ring_n20p0.8s5	602	73,890	43	43	43	*43*	217	3,009	18.15	17	1.28	14.63	0.68	163.65
ring_n20p0.9s1	672	91,739	48	48	48	*48*	268	4,501	19.58	28	1.18	27.66	0.82	248.47
ring_n20p0.9s2	696	98,814	49	49	49	*49*	267	3,866	21.53	37	1.46	38.78	0.98	299.82
ring_n20p0.9s3	686	95,572	47	47	47	*47*	224	4,683	28.19	49	1.47	53.28	1.05	288.49
ring_n20p0.9s4	686	95,789	47	47	47	*47*	245	4,489	25.82	51	1.47	55.21	0.97	293.18
ring_n20p0.9s5	706	101,600	49	49	49	*49*	262	4,654	23.15	43	1.35	47.46	1.01	312.35
ring_n20p1.0s1	760	117,420	42	50	50	**50**	243	6,681	53.88	84	2.74	113.91	2.48	715.60

First column contains the name for instance. Table 1 presents the experimental results for the known medium to large ring instances from the literature. Its second and third columns contain the number of vertices and edges, respectively; the fourth and fifth column display the best known lower and upper bounds, respectively, for the partition chromatic number (see [16] and [10]). The next columns list the results of running our branch-and-price algorithm. The column lb^r reports the lower bound provided by the linear relaxation of the root problem; column $\overline{ub}$ contains the upper bound obtained by our algorithm - italic is a mark that the optimum was found also in the literature and bold marks a new found optimum; column N lists the number of explored nodes of the branch-and-price tree. Columns $rVar$ and $nVar$ report the number of variables in the root node problem and the average number of newly added variables in nodes other than the root, respectively. Columns rSb and nSb list the number of subproblems solved in the root and the average number of subproblems solved in nodes other than the root, respectively. Columns rT, n, and T, respectively, report the root node, the average in nodes other than root, and the overall solution time (in seconds).

TABLE 2. Numerical results for new larger ring instances.

instance	n	m	lb^r	$\overline{ub}$	N	$rVar$	$nVar$	rSb	nSb	rT	nT	T
ring_n25p0.7s1	822	138,333	59	**59**	311	7,515	27.48	28	1.21	53.74	2.37	506.85
ring_n25p0.7s2	776	129,635	71	**71**	264	9,582	37.23	15	1.05	28.85	2.43	406.45
ring_n25p0.7s3	806	140,648	74	**74**	298	11,112	40.39	28	1.19	59.71	1.56	523.29
ring_n25p0.7s4	816	144,986	79	**79**	292	13,672	49.65	15	1.17	32.55	1.68	523.63
ring_n25p0.7s5	800	138,013	71	**71**	300	11,432	40.82	44	1.17	94.19	1.58	568.95
ring_n25p0.8s1	960	188,382	67	**67**	392	9,289	27.36	30	1.24	122.19	3.32	1,422.29
ring_n25p0.8s2	942	186,659	67	**67**	383	8,533	25.89	27	1.23	107.43	3.07	1,283.19
ring_n25p0.8s3	950	194,327	84	**84**	353	14,089	41.50	9	1.09	50.48	3.52	1,292.78
ring_n25p0.8s4	940	190,070	81	**81**	345	14,354	45.16	27	1.23	121.00	3.73	836.32
ring_n25p0.8s5	940	191,347	85	**85**	336	14,219	45.54	32	1.20	159.58	2.80	1,097.97
ring_n25p0.9s1	1,068	233,391	72	**72**	413	11,794	35.71	67	1.47	215.32	3.01	1,457.12
ring_n25p0.9s2	1,100	261,524	98	**98**	417	21,476	53.50	13	1.12	58.80	3.04	1,326.53
ring_n25p0.9s3	1,064	231,187	73	**73**	415	10,527	29.44	48	1.26	140.60	3.57	1,208.71
ring_n25p0.9s4	1,090	242,754	74	**74**	426	15,350	41.75	56	1.37	213.55	3.24	1,591.45
ring_n25p0.9s5	1,090	257,229	95	**95**	425	18,806	48.28	36	1.26	138.41	2.85	1,349.88
ring_n25p1.0s1	1,200	294,400	78	**78**	403	14,968	70.66	156	3.22	569.55	7.72	$3,675.00^{tl}$
ring_n30p0.5s1	844	146,507	63	**63**	325	6,300	21.05	9	1.10	18.72	1.47	497.79
ring_n30p0.5s2	908	178,701	83	**83**	339	13,650	42.46	53	1.13	140.75	1.95	800.79
ring_n30p0.5s3	812	141,883	71	**71**	299	9,668	34.60	43	1.14	84.66	1.57	554.69
ring_n30p0.5s4	886	169,016	76	**76**	336	14,354	46.80	74	1.26	202.58	2.15	923.38
ring_n30p0.5s5	868	165,340	83	**83**	318	16,095	56.52	113	1.38	283.50	2.37	1,037.65
ring_n30p0.6s1	1,068	233,508	77	**77**	435	12,203	31.73	35	1.21	105.52	2.90	1,356.37
ring_n30p0.6s2	1,062	245,080	96	**96**	408	19,849	50.92	56	1.14	206.30	3.31	1,559.21
ring_n30p0.6s3	1,044	235,906	93	**93**	388	17,299	49.23	45	1.30	173.51	2.72	1,228.31
ring_n30p0.6s4	1,080	252,092	99	**99**	405	21,126	54.88	52	1.17	191.97	3.25	1,505.83
ring_n30p0.6s5	1,066	247,333	83	**83**	318	16,095	56.52	113	1.38	283.50	2.37	1,037.65

In Table 1 we used twenty one instances from literature - mostly solved to optimality in [16] or in [10]. The optima for *ring_n20p0.9s3* and *ring_n20p0.9s4* was found in the second article but not in the first while the optimum for *ring_n20p0.9s5* was found only in the former article. We found also for the first time the partition chromatic number for the largest instance from the literature, namely *ring_n20p1.0s1*. For the remaining instances the optima was found in both articles. We can observe from Table 1 that all these instances are solved to optimality by our algorithm which makes it more consistent- the running time being more predictable and quite small, but of course this could be due to the different computing power we used.

Table 2 contains the results for the newly generated larger *ring* type instances (with up to $250,000$ edges); all these instances were solved to optimality. The time limit was

intentionally exceeded only for one instance - the largest of all - which suggests that the computing power along the good quality root lower bounds play a decisive role.

An interesting feature of our algorithm is the fact that the root linear relaxation with the initial set of variables obtained by perturbing an heuristic solution has a very good quality - its optimum is the same as for the ILP problem. Although it is difficult to compare the computing power of different equipped computers, we can observe from the Table 1 that the running time of our algorithm is quite small and more predictable than in [10] for the same group of the five instances for a given type of ring topology). Besides, from both Tables 1 and 2, it seems that all instances for which the root relaxation is solvable in a reasonable amount of time can be solved to optimality by our branch-and-price algorithm.

For both types of instances most of the time is dedicated to solving the root relaxation. Figure 1 shows (in average across the five instances for each ring) that the time for solving the root relaxation represents roughly $10 - 15\%$ of the overall time and the number of variables used in the root relaxation is around $40 - 50\%$ of the total number of generated variables (excluding those restored).

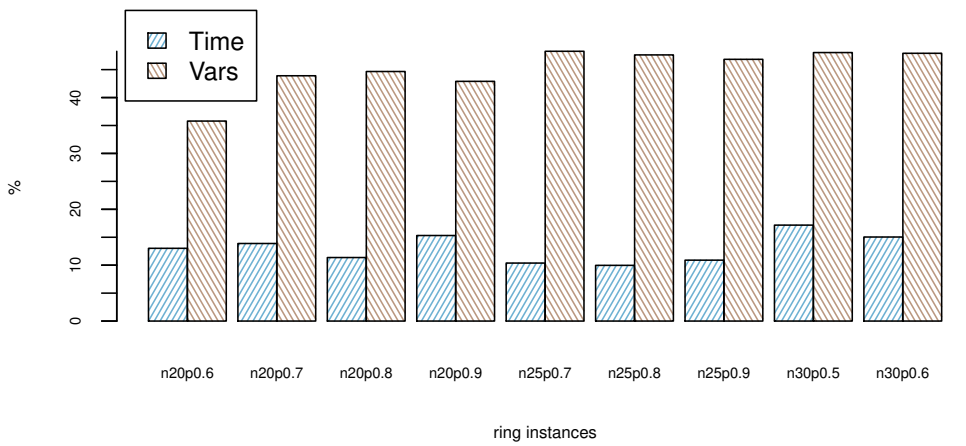

FIGURE 1. Percentage of solution time and number of generated variables in the root relaxation.

5.4. **Strengthening the root linear relaxation.** We next present the numerical results of a variant of our branch-and-price algorithm. This time we include an enforcement of the root linear relaxation with the families of valid inequalities (cuts) described in section 4, the goal being to get better lower bounds for the partition chromatic number. These cuts are added for small and medium instances and only to the root problem since building them are time consuming. The two families have a different effect on the quality of the optimum solution of the root LP relaxation, but in the same time they have a sort of complementary behaviour.

A valid inequality of first type (a maximal clique generic inequality) is generated like this: we first choose two stable sets intersecting the same cluster and second (if possible) a third stable set which doesn't intersect that cluster but intersects both first stable sets, then we extend this clique to a maximal one. Let f_1 be the number of times we repeat this process with the same cluster.

TABLE 3. Tuning the number of valid inequalities added to the root relaxation (for $n90p2t2s1$ instance).

$f_1 \backslash f_2$	500	1000	1500	3000	5000	7500	10000	15000
0	*0.97*	*1.62*	*1.82*	*2.70*	*3.41*	*4.11*	*4.21*	*4.86*
	5.99	8.79	14.54	13.15	19.17	26.77	34.52	50.28
1	*6.99**	8.56	9.46	10.85**	12.28**	13.73**	15.05**	16.40**
	7.97	9.16	10.08	13.31	18.35	24.15	30.92	43.18
5	8.45	10.10	11.03**	13.10**	14.01**	16.96**	17.91**	18.81**
	13.35	14.30	15.15	18.34	23.81	29.65	35.46	38.45
10	9.09	11.02	11.41	14.63**	15.38**	18.87**	20.86**	20.22**
	20.05	21.11	22.21	25.25	30.45	35.01	41.73	53.14

TABLE 4. Numerical results for the second versus the first branch-and-price algorithm.

instance	n	m	lb	ub	lb^r	$\overline{ub}$	N	$rVar$	$nVar$	rT	T	nT
n90p2t2s1	90	786	4	4	4	**4**	26	586	64.32	44.65	56.99	0.49
					3	4	300	586	232.46	5.96	23.70	0.06
n90p2t2s2	90	801	3	3	4	**4**	300	693	162.29	58.48	82.07	0.07
					3	4	300	693	50.71	6.14	17.83	0.06
n90p2t2s3	90	838	4	4	4	**4**	24	573	65.73	36.84	51.32	0.62
					3	4	300	573	241.93	5.76	26.96	0.07
n90p2t2s4	90	777	3	4	4	**4**	25	625	69.33	49.99	66.14	0.67
					3	4	300	625	207.42	6.33	23.03	0.05
n90p2t2s5	90	827	4	4	4	**4**	30	603	47.89	38.25	52.63	0.49
					3	4	300	603	258.70	5.75	26.17	0.06

An inequality of the second type (based on a majority clique) is generated in the following way: we first choose a candidate for a stability witness (see section 3, definition 4.2), S_0, of an appropriate size and then build the corresponding clique, $\mathcal{Q}_{I_{S_0}}$. Let f_2 be the number of times we perform this process for each possible size of the witness. Note that the associated inequality remains valid even if S_0 contains subsets of cardinality $\geq (p+1)/2$ whose corresponding variable, $x_{S_0'}$, doesn't belong to the problem - see the proof of Proposition 4.3. In other words the inequality remains valid because corresponds to a clique (which is not necessarily maximal); transforming all these cliques into maximal ones (thereby bringing facets defining inequalities) means to add a huge number of variables to the problem which should be avoided.

The way in which we tuned these two parameters, f_1 and f_2, is presented in the Table 3. We choose an instance for which the root problem has a non-integer optimum, namely *n90p2t2s1*. For this instance the solution to the root relaxation gives an optimum of 2.76 (obtained in 5.89 seconds - the average for ten runs), hence a lower bound for the partition chromatic number is 3. The addition of the above valid inequalities may increase this lower bound to the true value of the partition chromatic number which is 4 - the needed relative increase of optimum is 8.34%.

For each combination (f_1, f_2) we run the algorithm ten times and list two values in two corresponding rows: the average relative increase in optimum (as percentage) and the average solution time (in seconds). The italic values for the average relative increase in optimum value are those smaller than the above threshold. A star means that, while this threshold was not reached, some of the ten values are above it, a double star means that all ten values are at least 8.34%. We choose for our main experiments with this variant of the branch-and-price algorithm $f_1 = 10$ and $f_2 = 15,000$.

The experimental results are shown in Table 4. Column lb and ub list the best known bound for the partition chromatic number like above; column lb^r reports the lower bound provided by the linear relaxation of the root problem; column $\overline{ub}$ contains the upper bound obtained by performing the algorithm. Column N reports the number of explored nodes of the branch-and-price tree. Columns $rVar$ and $nVar$ report the number of variables in the root node problem and the average number of newly added variables in nodes other than the root, respectively. Columns rT, nT, and T, respectively, report the root node, the average in nodes other than root, and the overall solution time (in seconds).

For each instance we compared the results obtained with (the first row) and without (the second row) the cuts. For the simple branch-and-price algorithm the limit of 300 nodes of the tree was always reached without finding the optimum solution; by adding the cuts for four out of five instances the optimum was found, including one for which this optimum was previously unknown ($n90p2t2s4$).

6. Conclusions

In this paper we proposed a set packing integer programming formulation for the Partition Coloring Problem. Based on it we introduced a branch-and-price algorithm used for solving large and very large instances and a variant of it based on adding cutting planes to the root linear relaxation which used for solving small and medium instances.

By exploiting the resources of the parent nodes and restoring many of the stables sets (variables) and by using a simple branching rule our branch-and-price algorithm proved to be effective for large and very large instances performing at least as good and some times better than the state-of-the-art algorithms. We generate new large instances from RWA in all-optical networks instances with more than $250,000$ edges - all solved to optimality. Our approach seems to show that solving such a problem (on a ring instance) depends mostly on the computing resources since our model offers ILP problems with lower bounds of the root relaxation of very good quality.

The fact that all already known and new larger ring instances were solved to optimality and the running times are predictable (similar running times within the same family of instances) underlines the consistency of our set packing based algorithm.

The second variant algorithm is based on two families of valid inequalities and could be subject to improvements in terms of performance in order to be used on larger instances. This new algorithm proved to be very useful for small and medium problems with non-integral root relaxation optimum - for which the solving is quite difficult.

Future researches concern the improvement of the second algorithm: on one hand by using new classes of cuts and on the other hand by a better integration of these cutting planes towards a branch-price-and-cut algorithm.

Acknowledgements

This work was supported by FCSU grant, 1004.05 zone.

References

[1] Barnhart, C., Johnson, E. L. , Nemhauser, G. L., Savelsbergh, M. W. P. and P. H. Vance., Column generation for solving huge integer programs in *Op. Res.* **46** (1998), no. 3, 316–329.

[2] Campêlo, M., Campos, V. A. and Corrêa, R. C., On the asymmetric representatives formulation for the vertex coloring problem, *Discr. App. Maths.* **156** (2008), no. 7, 1097–1111.

[3] Demange, M., Monnot, J., Pop, P. and Ries, B., Selective graph coloring problem in some special classes of graphs, in *Combinatorial Optimization*, LNCS **7422** (2012), 320–331.

[4] Demange, M., Monnot, J., Pop, P. and Ries, B., On the complexity of the selective graph coloring problem in some special classes of graphs, *Theor. Comp. Sc.* **540-541** (2014), 89–102.

[5] Demange, M., Ekim, T., Ries, B. and Tanasescu, C., On some applications of the selective graph coloring problem, *Eur. J. of Op. Res.* **240** (2015), no. 2, 307–314.
[6] Desaulniers, G., Desrosiers, J., Solomon, M. M., Column Generation, *Springer, US*, 2005.
[7] Desaulniers, G. and Lubbecke, M. E.J., Branch-price-and-cut, in J. J. Cochran, L. A. Cox Jr., P. Keskinocak, J. P. Kharoufeh, J. C. Smith, (eds.) Wiley Encyclopedia of Operations Research and Management Science, *John Wiley and Sons, Inc., US*, 2010.
[8] Fidanova S. and P. Pop, P., An improved hybrid ant-local search algorithm for the partition graph coloring problem, *J. of Comp. and App. Maths.* **293** (2016), 55–61.
[9] Frota, Y., Maculan, N., Noronha, T. F. and Ribeiro, C. C., A branch-and-cut algorithm for partition coloring, *Special Issue: Network in Optimization* (INOC 2007) **55** (2009), no. 3, 194–204.
[10] Furini, F., Malaguti, E. and Santini, A., An exact algorithm for the partition coloring problem, *Comp. and Op. Res.* **92** (2018), 170–181.
[11] Gamache, M., Hertz, A. and Ouellet, J. O., A graph coloring model for a feasibility problem in monthly crew scheduling with preferential bidding, *Comp. and Op. Res.* **34** (2007), no. 8, 2384–2395.
[12] Giaro, K., Kubale, M. and Obszarski, P., A graph coloring approach to scheduling of multiprocessor tasks on dedicated machines with availability constraints, *Discr. App. Maths.* **157** (2009), no. 17, 3625–3630.
[13] Guo, Z. , Xiao, M. and Zhou, Y. , The complexity of the partition coloring problem, in J. Chen, Q. Feng, and J. Xu, (eds.) Theory and Applications of Models of Computation, *Springer International Publishing*, 2020, 390–401.
[14] Hansen, P., Labbé, M. and Schindl, D., Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results, *Discr. Opt.* **6** (2009), 135–147.
[15] Held, S., Cook, W. and Sewell, E. C., Maximum-weight stable sets and safe lower bounds for graph coloring, *Math. Progr. Comp.* **4** (2012), 363–381.
[16] Hoshino, E. A., Frota, Y. A. and de Souza, C. C., A branch-and-price approach for the partition coloring problem, *Op. Res. Lett.* **39** (2011), 132–137.
[17] Jabrayilov, A. and Mutzel, P., New integer linear programming models for the vertex coloring problem, in *Latin American Symposium on Theoretical Informatics*, LNCS (2018), 640–652.
[18] Lewis, R. M. R., A Guide to Graph Colouring - Algorithms and Applications, *Springer, Switzerland*, 2016.
[19] Li, G. and Simha, R., The partition coloring problem and its application to wavelength routing and assignment, in *Proceedings of the First Workshop on Optical Networks* (2000).
[20] Lubbecke, M. E. and Desrosiers, J., Selected topics in column generation, *Op. Res.* **53** (2005), no. 6, 1007–1023.
[21] Malaguti, E. and Toth, P., A survey on vertex coloring problems, *Int. Trans. in Op. Res.* **17** (2010), 1–34.
[22] Mehrotra, A. and Trick, M., A column generation approach for graph coloring, *INFORMS J. on Comp.* **8** (1996), no. 4, 344–354.
[23] Mostafaie, T. and Khiyabani, F. M. and Navimipour, N. J., A systematic study on meta-heuristic approaches for solving the graph coloring problem, *Comp. and Op. Res.* **120** (2020).
[24] Noronha, T. H. and Ribeiro, C. C., Routing and wavelength assignment by partition colouring, *Eur. J. of Op. Res.* **171** (2006), 797–810.
[25] Padberg, M. W., On the facial structure of set packing polyhedra, *Math. Prog.* **5** (1973), 199–215.
[26] Padberg, M. W., On the complexity of set packing polyhedra, *Ann. of Discr. Maths.* **1** (1977), 421–434.
[27] Pop, P., Hu, B. and Raidl, G. R., A memetic algorithm for the partition graph coloring problem, in *14th International Conference on Computer Aided Systems Theory*, Gran Canaria, Spain, LNCS (2013), 167–169.
[28] Schrijver, A., Combinatorial optimization - Polyhedra and efficiency, *Springer-Verlag, Berlin, Heidelberg*, 2003.
[29] Zhou, Y., Hao, J.-K. and Duval, B., Reinforcement learning based local search for grouping problems: A case study on graph coloring, *Exp. Sys. with Apps.* **64** (2016), 412–422.
[30] Zhu, E., Jiang, F., Liu, C. and Xu, J., Partition independent set and reduction-based approach for partition coloring problem, *IEEE Trans. on Cybern.* **52** (2020), no. 6, 4960–4969.

ALEXANDRU IOAN CUZA" UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
GENERAL BERTHELOT STREET, NO 16, 700483, IAŞI, ROMANIA
Email address: emanuel.olariu@info.uaic.ro

ALEXANDRU IOAN CUZA" UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
GENERAL BERTHELOT STREET, NO 16, 700483, IAŞI, ROMANIA
Email address: cristian.frasinaru@uaic.ro