A Branch-and-Cut Algorithm for Partition Coloring

Yuri Frota, Nelson Maculan
Universidade Federal do Rio de Janeiro, COPPE
Programa de Engenharia de Sistemas e Otimização
Rio de Janeiro, RJ 21945-970, Brazil
{abitbol, maculan}@cos.ufrj.br

Thiago F. Noronha
Catholic University of Rio de Rio de Janeiro
Department of Computer Science
Rio de Janeiro, RJ 22453-900, Brazil
tfn@inf.puc-rio.br

Celso C. Ribeiro
Universidade Federal Fluminense
Department of Computer Science
Niterói, RJ 22410-240, Brazil
celso@inf.puc-rio.br

Abstract

Let $G = (V, E, Q)$ be a undirected graph, where V is the set of vertices, E is the set of edges, and $Q = \{Q_1, \dots, Q_q\}$ is a partition of V into q subsets. We refer to $Q_1, \dots, Q_q$ as the components of the partition. The Partition Coloring Problem (PCP) consists of finding a subset V' of V with exactly one vertex from each component $Q_1, \dots, Q_q$ and such that the chromatic number of the graph induced in G by V' is minimum. This problem is a generalization of the graph coloring problem. This work presents a branch-and-cut algorithm proposed for PCP. An integer programming formulation and valid inequalities are proposed. A tabu search heuristic is used for providing primal bounds. Computational experiments are reported for random graphs and for PCP instances originating from the problem of routing and wavelength assignment in all-optical WDM networks.

Keywords: partition coloring, branch-and-cut, formulation by representatives.

1 Introduction

Let $G = (V, E, Q)$ be an undirected graph, where V is the set of vertices, E is the set of edges, and $Q = \{Q_1, \ldots, Q_q\}$ is a partition of V into q subsets, i.e., $Q_1 \cup \cdots \cup Q_q = V$ and $Q_i \cap Q_j = \emptyset$, for every $i, j = 1, \ldots, q$ with $i \neq j$. We refer to $Q_1, \ldots, Q_q$ as the *components* of the partition (or, more simply, as the components). We denote by $P[v]$ the index of the component of vertex $v \in V$: i.e., $v \in Q_{P[v]}$. The *Partition Coloring Problem* (PCP) consists of finding a subset of vertices $V' \subseteq V$ such that $|V' \cap Q_i| = 1$, for every $i = 1, \ldots, q$ (i.e., V' contains one vertex from each component Q_i), and the chromatic number of the graph induced in G by V' is minimum. This problem is clearly a generalization of the graph coloring problem. Li and Simha [18] have shown that the decision version of PCP is NP-complete.

We illustrate an instance of PCP in Figure 1. The associated graph has seven vertices and three components. An optimal solution makes use of two colors: the first color is used to color vertices 2 and 5, while the second is used to color vertex 3.

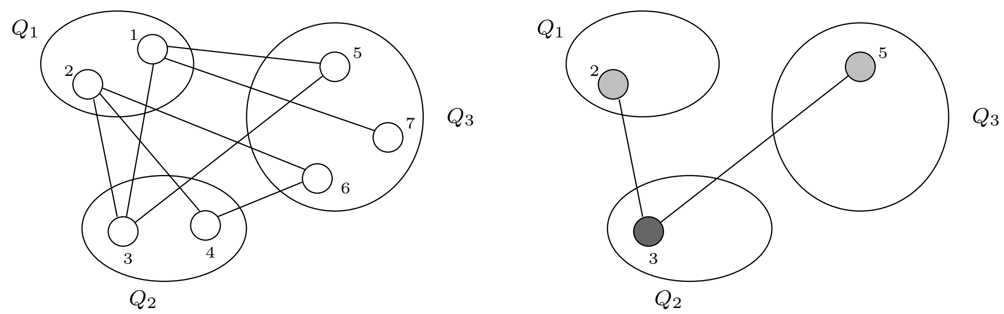


Figure 1: (a) Instance of PCP and (b) optimal solution with two colors.

Algorithms for solving PCP have been used in the literature as building blocks for algorithms for the *Routing and Wavelength Assignment* (RWA) problem in optical networks [15, 22]. In such networks, each signal is converted to the optical domain and reaches the receiver without conversion to the electrical domain. *Wavelength Division Multiplexing* (WDM) allows the more efficient use of the huge capacity of optical fibers, as far as it permits the simultaneous transmission of different channels along the same fiber, each of them using a different wavelength. An all-optical point-to-point connection between two vertices is called a lightpath. Two lightpaths may use the same wavelength, provided they do not share any common link. The routing and wavelength assignment problem consists of routing a set of lightpaths and assigning a wavelength to each of them. Variants of RWA are characterized by different optimization criteria and traffic patterns; e.g., see [8].

Li and Simha [18] proposed a two phase decomposition strategy to solve the min-RWA off-line variant, in which the objective function consists of minimizing the total number of wavelengths used to route all traffic demands. In the first phase, one or more possible routes are computed for each lightpath. In the second phase, one precomputed route and one wavelength are assigned to each lightpath by solving a PCP instance. In this transformed instance, the vertices correspond to routes, there is an edge between each pair of vertices whose associated routes share a common link, and all alternative routes associated with the same connection are placed in the same component of the partition.

This paper presents an exact branch-and-cut algorithm for the partition coloring problem. Related works are discussed in Section 2. Section 3 presents the integer programming formulation proposed for PCP. Section 4 describes the proposed branch-and-cut algorithm. Computational experiments are reported in Section 5. Concluding remarks are drawn in the last section.

2 Related work

Li and Simha [18] proposed two groups of construction heuristics for PCP, referred to as one-step algorithms and two-step algorithms. The heuristics in the first group are extensions of the three well-known graph coloring heuristics: largest-first [26], smallest-last [19], and color-degree [2]. The corresponding PCP heuristics are called `onestepLF`, `onestepSL`, and `onestepCD`, respectively. At each iteration, one vertex from a yet uncolored component of the partition is selected, according to the same greedy criterion used in the respective heuristic for graph coloring. The selected vertex is colored with the smallest color available (refer to colors as integer numbers), and the other vertices in the same component are discarded (i.e., they remain uncolored). Two-step algorithms have two different phases. First, one vertex is selected from each component. Next, a classical graph coloring heuristic is applied to color the graph induced by the selected vertices. The heuristics are called `twostepLF`, `twostepSL`, and `twostepCD`, respectively. The best results were obtained with algorithm `onestepCD` (One Step Color Degree).

Noronha and Ribeiro [22] proposed an improvement heuristic to PCP, based on tabu search [12]. Algorithm `onestepCD` is applied to create a feasible initial solution S to PCP with C colors. Then, a new (possibly infeasible) solution S' using $C-1$ colors is built. Next, the tabu search procedure `TS-PCP` attempts to restore the feasibility of S'. A *coloring conflict* is defined by a pair of adjacent vertices in different components which are colored with the same color. `TS-PCP` aims to minimize the number of coloring conflicts until there are no coloring conflicts in S' and feasibility is restored. It is based on a first-improving local search strategy using a 1-opt neighborhood. Each neighbor is obtained by (a) recoloring with a different color exactly one vertex involved in a coloring conflict or (b) changing (and coloring) the vertex that is colored in a component involved in a coloring conflict. If at any point of the algorithm there are no coloring conflicts in S', then a new solution with $C-1$ colors is obtained and the procedure is restarted from S' attempting to reduce one further color. If a stopping criterion is met and solution S' is still infeasible, the procedure halts and the best feasible solution is returned.

There is no exact algorithm for PCP in the literature. However, there are many algorithms for the classical graph coloring problem. The first enumeration techniques [2, 3, 16, 17] were inefficient for medium and large size instances. Better results were obtained by integer programming approaches. Mehrotra and Trick [20] proposed a column generation algorithm for the maximum independent set that was able to solve medium size instances. Figueiredo et al. [11] and Méndez-Díaz and Zabala [21] developed branch-and-cut algorithms. Campêlo et al. [5] proposed a 0-1 integer formulation for the graph coloring problem that can be seen as a set packing formulation with additional constraints [1]. An asymmetric formulation and valid inequalities for the same problem was proposed in [4]. The formulations in [4, 5] are extended in the next section to the partition coloring problem.

3 Integer programming formulation

The formulation proposed in this section is based on choosing one vertex to be the representative of all vertices with the same color, instead of directly coloring all vertices. Therefore, each vertex is in exactly one of the following three states: (i) colored and representing all vertices with this same color, (ii) colored and represented by another vertex with the same color, or (iii) uncolored.

We define $A(u) = \{w \in V : (u, w) \notin E, w \neq u\}$ as the *anti-neighborhood* of vertex u (i.e., the subset of vertices that are not adjacent to u) and $A_P(u) = \{v \in A(u) : P[u] \neq P[v]\}$ as the *component anti-neighborhood* of a vertex $u \in V$ (i.e., the vertices in the anti-neighborhood of u that are in another component of the partition). We also define $A'_P(u) = A_P(u) \cup \{u\}$. Given a subset of vertices $V' \subseteq V$, we denote by $E[V']$ the subset of edges induced in $G = (V, E)$ by V'. A vertex $v \in A_P(u)$ is said to be *isolated* in $A_P(u)$ if $E[A_P(u)] = E[A_P(u) \setminus \{v\}]$ (i.e., vertex v has no adjacent vertex in $A_P(u)$). We define the binary variables x_{uv} for all $u \in V$ and for all $v \in A'_P(u)$, such that $x_{uv} = 1$ if and only if vertex u represents the color of vertex v; otherwise $x_{uv} = 0$. The number of x variables is $\mid V \mid + \overline{m}_c$, where $\overline{m}_c$ is the number of edges in the complementary graph of G whose endvertices are in different components. The PCP can be formulated as the following integer programming problem:

$$\min \sum_{u \in V} x_{uu} \tag{1}$$

subject to:

$$\sum_{u \in Q_p} \sum_{v \in A'_P(u)} x_{vu} \geq 1 \quad \forall p = 1, \ldots, q \tag{2}$$

$$x_{uv} + x_{uw} \leq x_{uu} \quad \forall u \in V, \quad \forall (v, w) \in E \text{ with } v, w \in A_P(u) \text{ and } P[v] \neq P[w] \tag{3}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A_P(u) \text{ such that } v \text{ is isolated in } A_P(u) \tag{4}$$

$$x_{uv} \in \{0, 1\} \quad \forall u \in V, \quad \forall v \in A'_P(u). \tag{5}$$

The above model is said to be the *formulation by representatives* of PCP. The objective function (1) counts the number of representative vertices, i.e., the number of colors. Constraints (2) enforce that each component $Q_p, p = 1, \ldots, q$, have at least one of its vertices $u \in Q_p$ represented either by itself ($x_{uu} = 1$) or by some other vertex v ($x_{vu} = 1$, with $v \neq u$) in its component anti-neighborhood (i.e., a vertex from another component that does not share an edge with u). Inequalities (3) enforce that adjacent vertices have distinct representatives. Inequalities (3) together with constraints (4) ensure that a vertex can only be represented by a representative vertex. Note that a feasible solution for inequalities (2) - (5) may assign multiple representatives to the same vertex. However, any of the representative vertices leads to a feasible solution.

To break symmetries in the above formulation, we generalized the asymmetric formulation by representatives in [4]. We establish that a vertex u can only represent the color of a vertex v if $P[u] < P[v]$. Therefore, the representative vertex of a color is the vertex with the smallest component index. We define $A_{P>}(u) = \{v \in A_P(u) : P[u] < P[v]\}$ as the *out-anti-neighborhood* of a vertex $u \in V$ (i.e., the vertices that cannot represent the vertex u) and $A_{P<}(u) = \{v \in A_P(u) : P[v] < P[u]\}$ as the *in-anti-neighborhood* of vertex u (i.e., the vertices that can represent the vertex u). We also define $A'_{P>}(u) = A_{P>}(u) \cup \{u\}$ and $A'_{P<}(u) = A_{P<}(u) \cup \{u\}$. Based on this property, inequalities (2) - (5) are rewritten as inequalities (7) - (10), respectively.

A vertex that stands alone in a component is called an *elementary vertex*. We define $V^e \subseteq V$ as the set of all elementary vertices in G (i.e., the set of vertices that are guaranteed of being colored), $V^0 = \{u \in V^e : A_{P<}(u) = \emptyset\}$ as the set of elementary vertices that have no in-anti-neighborhood in G (i.e., the set of vertices that are always representatives), and Q^0 as the set of components that contains the vertices in V^0. Since a vertex $v \in V^0$ is always a representative, $x_{vv} = 1$ in any feasible solution of PCP. Therefore, these variables may be removed from the asymmetric formulation and the number of x variables is reduced to $\overline{m}_c + |V \backslash V^0|$. The objective function (1) is rewritten as (6) in the new formulation:

$$\min \sum_{v \in V \backslash V^0} x_{vv} + \mid V^0 \mid \tag{6}$$

subject to:

$$\sum_{u \in Q_p} \sum_{v \in A'_{P<}(u)} x_{vu} \geq 1 \quad \forall Q_p \in Q \backslash Q^0 \tag{7}$$

$$x_{uv} + x_{uw} \leq \beta_u \quad \forall u \in V, \quad \forall (v, w) \in E \text{ with } v, w \in A_{P>}(u) \text{ and } P[v] \neq P[w] \tag{8}$$

$$x_{uv} \leq x_{uu} \quad \forall u \in V, \quad \forall v \in A_{P>}(u) \text{ such that } v \text{ is isolated in } A_{P>}(u) \tag{9}$$

$$x_{uv} \in \{0, 1\} \quad \forall u \in V, \quad \forall v \in A'_{P>}(u), \tag{10}$$

where $\beta_u = 1$ if $u \in V^0$; otherwise $\beta_u = x_{uu}$.

4 Branch-and-cut

In this section, we describe the branch-and-cut algorithm for partition coloring [6] based on the asymmetric formulation given by inequalities (6) - (10). Valid inequalities are progressively added to each subproblem of the search tree, which in most cases improves the linear relaxation bound. We first explain how to reduce the size of a PCP instance by preprocessing. Next, we describe the branching strategy. Following, we show how to build good feasible solutions for each subproblem in the branch-and-cut tree. Finally, we present valid inequalities that are used in a cutting plane procedure developed for improving the linear relaxation.

4.1 Preprocessing

Preprocessing is used to reduce the size of the graph before applying the branch-and-cut algorithm. It is divided into three steps.

First, all edges $(u, v) \in E$ with $P[u] = P[v]$ are removed from the graph, because at most one of u and v is colored in the same solution.

Next, every elementary vertex (i.e., a vertex which stands alone in its component) connected to all other vertices in the graph is eliminated, because they will require an additional color whatever the colors of the other vertices are. The number of vertices removed is added to the chromatic number obtained at the end of the branch-and-cut algorithm.

Finally, all components with at least one vertex v with degree smaller than the linear relaxation lower bound at the root node of the branch-and-cut tree are removed. This can be done because the number of colors in the optimal solution will be larger than the number of neighbors of v. Therefore, v can be colored with any of the colors in the solution that are not being used by any of its neighbors.

4.2 Branching rule

The classical branching rule of Mehrotra and Trick [20] for the graph coloring problem branches on two non-adjacent vertices. We adopted a modified rule which takes into account the specific characteristics of PCP. We define the following operations for PCP, for any $i, j = 1, \ldots, q$ such that $i \neq j$:

- SAME(i, j) enforces that the same color is used to color components Q_i and Q_j; and
- DIFFER(i, j) requires that different colors are used to color components Q_i and Q_j.

The first operation can be implemented by merging the two components Q_i and Q_j into a single one: the new component will be formed by merging every pair of vertices $w \in Q_i$ and $z \in A_P(w) \cap Q_j$ such that $(w, z) \notin E$ into a new vertex y, with the neighbors of y being those of w or z. DIFFER(i, j) can be implemented by inserting edges between every pair of non-adjacent vertices in components Q_i and Q_j.

When these two operations are simultaneously applied in the branching step, they create two new PCP subproblems such that an optimal feasible coloring to the original problem exists in exactly one of them. In addition, this branching rule has the advantage of leading to denser graphs without increasing the complexity of the subproblems. Figure 2 illustrates the decomposition of a problem defined by a graph G into two disjoint subproblems defined by graphs G_1 and G_2, where G_1 was generated by SAME$(1, 2)$ and G_2 by DIFFER$(1, 2)$.

4.3 Upper bounds

The tabu search procedure `TS-PCP` [22] described in Section 2 provides an upper bound on each node of the branch-and-cut tree. However, we introduce another procedure to provide the initial solution to `TS-PCP`.

Since the branching strategy described in the previous subsection implies that the graph associated with the parent node slightly differs from those associated with its children, we can construct an initial feasible solution to `TS-PCP` at any node of the branch-and-cut tree by using the solution associated with its parent node. The constructive procedure starts from a partial solution that is equal to the solution at the parent node in the branch-and-cut tree, except for the two components involved in the branching operation. The partial solution has at most two uncolored components, which are colored with the same greedy strategy of `onestepCD` [18]. Therefore, `TS-PCP` starts from an initial solution with at most two more colors than the solution at the parent node.

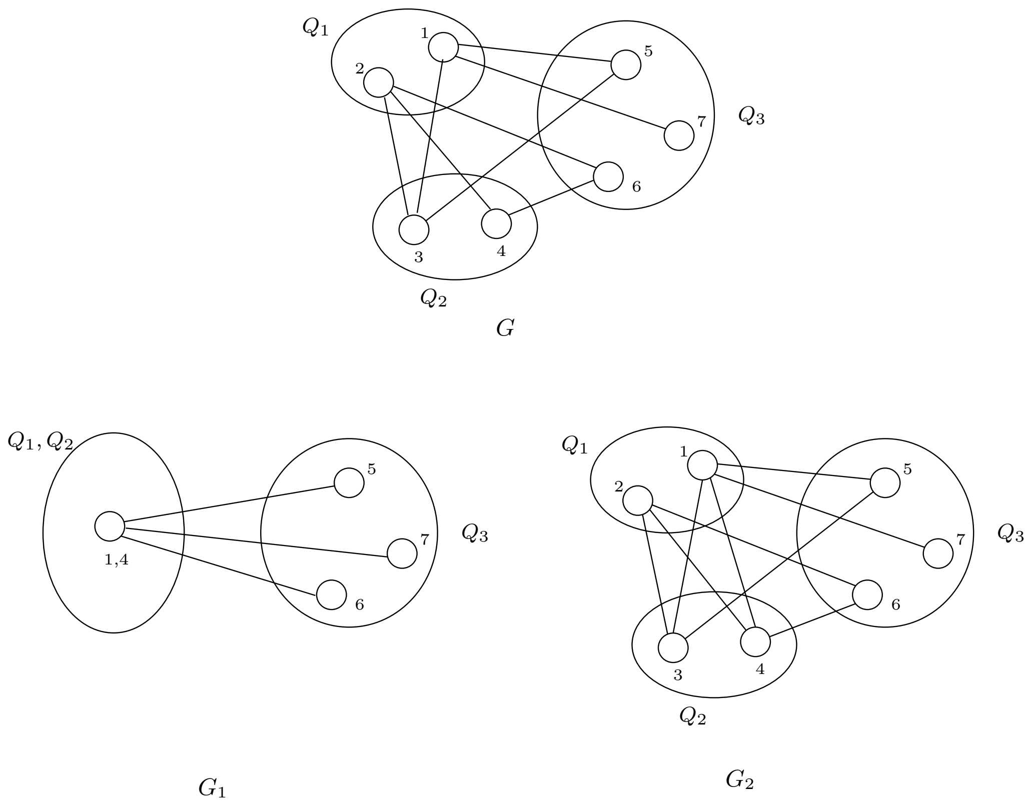


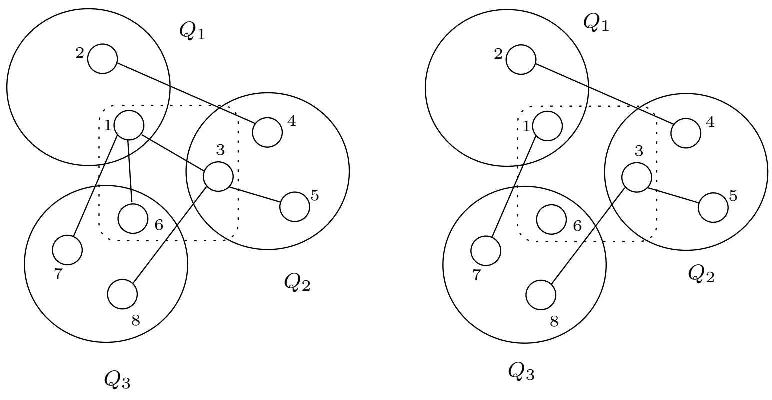
Figure 2: Decomposition of a problem graph G into two disjoint subproblem graphs G_1 and G_2.

4.4 Valid inequalities

The lower bounds provided by the relaxation of the integrality constraints of the symmetric and asymmetric formulations by representatives may be poor. Therefore, we generalize the two families of valid inequalities described in [4, 5]. A *component set* is a subset of V whose vertices belong to different components, while an *independent component set* is a component set whose vertices are not connected by inner edges (see Figure 3).

Internal cuts are based on the idea of how many colors are necessary to color any odd hole or anti-hole of the graph. We consider odd holes or odd anti-holes composed only of elementary vertices, because they are the only vertices that are guaranteed of being colored in a feasible solution. The internal cuts are defined by (11), where $H \subseteq V^e$ induces an odd hole or an odd anti-hole in G and $\chi(G[H])$ is the chromatic number of the subgraph $G[H]$ induced in G by H:

$$\sum_{v \in H \setminus V^0} x_{vv} + \mid H \cap V^0 \mid + \sum_{v \in H \setminus V^0, w \in A_{P<}(v) \setminus H} x_{wv} \geq \chi(G[H]). \tag{11}$$



(a) **(b)**

Figure 3: (a) A component set and (b) an independent component set.

Theorem 1 *If $H \subseteq V^e$ induces an odd hole or an odd anti-hole in G, then (11) is a valid inequality for the asymmetric formulation.*

Proof: The first component of the left-hand side counts the number of vertices in H that are representatives. The second component of the left-hand side counts the number of vertices in H that are represented by a vertex not in H. In any feasible coloring of G, each color appearing in H adds one either to the first or to the second term of the left-hand side. Since at least $\chi(G[H])$ colors are necessary to color $G[H]$, then inequality (11) holds. $\square$

External cuts are based on the idea of how many of the vertices of a subset $K \subseteq A_{P>}(u)$ can be represented by vertex $u \in V$. They are strengthened versions of inequalities (8). For any vertex $v \in K$, we define α_v as the maximum size of an independent set of the graph $G[K]$ induced in G by K that contains v, and $\alpha_K = \max_{v \in K}\{\alpha_v\}$ as the size of the maximum independent set of $G[K]$. The external cuts are defined by (12):

$$\sum_{v \in K} \frac{x_{uv}}{\alpha_v} \leq \beta_u. \tag{12}$$

Theorem 2 *For every $u \in V$ and any non-empty component set $K \subseteq A_{P>}(u)$, (12) is a valid inequality for the asymmetric formulation.*

Proof: We consider a feasible solution to PCP. If u is not a representative vertex in this coloring, then $\beta_u = 0$ and $x_{uv} = 0$ for all $v \in A_{P>}(u)$. Therefore, inequality (12) holds when u is not a representative vertex. We now consider the case where u is a representative vertex. Then, $\beta_u = 1$ and $x_{uv} = 1$ for all $v \in W$, otherwise $x_{uv} = 0$, where $W \subseteq G[K]$ is the component independent set composed of the vertices represented by u in the coloring. Let $\delta = \min_{v \in W}\{\alpha_v\}$. Then, $|W| \leq \delta$, because the cardinality of W is not larger than the largest independent set of its vertices. It follows that

$$\sum_{v \in K} \frac{x_{uv}}{\alpha_v} = \sum_{v \in W} \frac{1}{\alpha_v} \leq \sum_{v \in W} \frac{1}{\delta} = \frac{|W|}{\delta} \leq 1.$$

Therefore, inequality (12) also holds when u is a representative vertex. □

A clique K of graph G is said to be a *component clique* if all vertices in K belong to different components. Corollary 1 below follows from Theorem 2 and the fact that $\alpha_v = 1$, for any $v \in K$.

Corollary 1 *For every $u \in V$ and any maximal component clique $K \subseteq A_{P>}(u)$,*

$$\sum_{v \in K} x_{uv} \leq \beta_u \tag{13}$$

is a valid inequality for the asymmetric formulation.

An odd hole (resp. odd anti-hole) H is said to be a *component odd hole* (resp. *component odd anti-hole*) if all its vertices belong to different components. Theorem 2 is valid for any non-empty component set. Therefore, it is also valid for a component odd hole and for a component odd anti-hole. In both cases, the value of α_v, for any $v \in H$, is equal to the size α_H of the maximum independent set of $G[H]$, where $\alpha_H = \lfloor |H|/2 \rfloor$ for odd holes and $\alpha_H = 2$ for odd anti-holes. Therefore, the left-hand side of (12) can be rewritten as

$$\sum_{v \in H} \frac{x_{uv}}{\alpha_v} = \sum_{v \in H} \frac{x_{uv}}{\alpha_H} = \frac{\sum_{v \in H} x_{uv}}{\alpha_H}$$

for any $u \in V$ and $H \subseteq A_{P>}(u)$, and Corollary 2 follows:

Corollary 2 *For every $u \in V$ and any component odd hole or any component anti-hole $H \subseteq A_{P>}(u)$,*

$$\sum_{v \in H} x_{uv} \leq \alpha_H \beta_u \tag{14}$$

is a valid inequality for the asymmetric formulation.

4.5 Cut identification

We developed a cutting plane procedure based on inequalities (12)-(14). The separation of these inequalities consists of finding cliques, odd holes, and odd anti-holes in the graph associated with each node of the branch-and-cut tree. We use a GRASP heuristic for finding clique cuts and a modification of the Hoffman and Padberg heuristic [13] for finding odd holes and anti-holes cuts. Both procedures are described in the next two subsections.

4.5.1 Separation of external clique cuts

Let $G^u = (V^u, E^u)$ be the subgraph induced in the graph $\overline{G}$ associated with some node of the branch-and-cut tree by the out-anti-neighborhood $V^u = A_{P>}(u)$ of a vertex $u \in V$. Furthermore, let $\bar{x}_{uv}$ be the optimal value of variable x_{uv} in the linear relaxation of the asymmetric formulation (6) - (10), for any $u \in V$ and any $v \in A'_P(u)$. For any $u \in V$ such that $\bar{x}_{uu} > 0$, the separation of an external clique cut consists of finding a component clique $K \subseteq V^u$ such that $\sum_{v \in K} \bar{x}_{uv} > \beta_u$.

We developed a GRASP [9, 10, 23, 24, 25] heuristic for finding cuts. The heuristic is an iterative procedure composed of two phases: a *construction phase* and a *local search phase*. The construction phase finds an initial solution that might be later improved by the local search phase.

The GRASP heuristic attempts to find a clique C with maximum weight $\sum_{v \in K} \bar{x}_{uv}$ for each vertex $u \in V$ such that $\bar{x}_{uu} > 0$. Its construction phase begins with an empty clique C and builds a component clique, one vertex at a time. Let $C \subseteq V^u$ be a component clique of $\overline{G}$ and $\delta(C) = \{w \in V^u \backslash C : C \cup \{w\}$ is also a component clique of $\overline{G}\}$. At each iteration, one vertex $w \in \delta(C)$ is inserted into C with probability $\bar{x}_{uw}/(\sum_{z \in \delta(C)} \bar{x}_{uz})$ and the set $\delta(C)$ is updated. The procedure is repeated until $\delta(C) = \emptyset$.

There is no guarantee that the construction phase returns a locally optimal solution with respect to some neighborhood. Therefore, the component clique C may be improved by a local search procedure. The neighborhood $\gamma(C)$ is defined as the set of all component cliques obtained by exchanging a vertex $v \in C$ with another vertex $u \in \delta(C \backslash \{v\})$. The method starts with the solution provided by the construction phase. It iteratively replaces the current solution by that with maximum weight within its neighborhood. The local search halts when no better solution is found in the neighborhood of the current solution.

The heuristic stops after $10 \cdot |V^u|$ iterations have been performed since the last time the best solution was updated. The $|V^u|$ heaviest cliques are selected and a cut is generated for each clique C such that $\sum_{v \in C} \bar{x}_{uv}$. The cuts are added to the asymmetric formulation defined by (6) - (10).

4.5.2 Separation of external odd hole and anti-hole cuts

For any $u \in V$ such that $\bar{x}_{uu} > 0$, the separation of external odd hole cuts consists of finding a component odd hole or anti-hole $H \in V^u$ such that $\sum_{v \in H} \bar{x}_{uv} > \alpha_H \beta_u$.

We developed a generalization of the Hoffman and Padberg [13] algorithm for finding violated odd hole inequalities in $G^u = (V^u, E^u)$. The same algorithm is applied to the complementary graph to find violated odd anti-hole inequalities.

First, the method performs a breadth-first search labeling of the vertices of graph $G^u = (V^u, E^u)$ from any root vertex randomly chosen. A label h_w is assigned to each vertex $w \in V^u$. Figure 4 illustrates a layered graph with its vertex labels. For any two adjacent vertices w and z with labels $h_w = h_z \geq 2$, if there exist two vertex-disjoint paths p_w (from w to r) and p_z (from z to r), then there exists an odd cycle that contains w, z, and r.

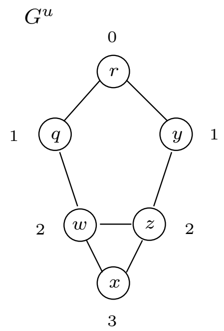


Figure 4: Layered graph with labels.

Next, the algorithm assigns weights $t_{wz} = 2 - \bar{x}_{uw} - \bar{x}_{uz}$ to every edge $(w, z) \in E^u$. Then, for every edge $(w, z) \in E^u$ with $h_w = h_z \geq 2$ and $P[w] \neq P[z] \neq P[r]$, the algorithm computes two vertex-disjoint shortest component paths (i.e., shortest paths formed exclusively using vertices from different components), one from z to r and the other from w to r. If both component paths exist, a component odd hole that can be used to generate a violated external cut is found. Otherwise, the algorithm continues from the next edge. This algorithm is applied $0.4 \cdot |V^u|$ times, starting from different root vertices. Other algorithms for the separation of odd hole inequalities can be found in [7].

4.5.3 Separation of internal odd hole and odd anti-hole cuts

The separation of internal odd hole or anti-hole cuts consists of finding a component odd hole or anti-hole $H \in V$ such that

$$\sum_{v \in H \setminus V^0} \bar{x}_{vv} + |H \cap V^0| + \sum_{v \in H \setminus V^0, w \in A_{P<}(v) \setminus H} \bar{x}_{wv} < \chi(G[H]).$$

The algorithm is similar to that developed for finding external odd hole cuts. The method is applied to the complementary graph to find violated odd anti-hole inequalities. First, it builds a layered graph rooted at a randomly chosen vertex $r \in V$. Next, the algorithm assigns weights

$$t_{wz} = \sum_{u \in A'_{P>}(w) \setminus H} \bar{x}_{uw} + \sum_{u \in A'_{P>}(z) \setminus H} \bar{x}_{uz}$$

to every edge $(w, z) \in E$. Then, for any edge $(w, z) \in E$ with $h_w = h_z \geq 2$ and $P[w] \neq P[z] \neq P[r]$, the algorithm calculates two vertex-disjoint shortest component paths, one from z to r and the other from w to r. If both component paths exist, a component odd hole that can be used to generate a violated internal cut is found. Otherwise, the algorithm continues from the next edge. This algorithm is repeated $0.4 \cdot |V|$ times, starting from different root vertices.

5 Computational results

Algorithm `B&C-PCP` described in the previous section was implemented in C++ and compiled with version v3.41 of the Linux/GNU compiler. The linear relaxation of the integer programming formulation was solved by version 2005-a of XPRESS. All experiments were performed on an AMD-Atlon machine with a 1.8 GHz clock speed and one Gbyte of RAM memory. The algorithm stops if the optimal solution is not found within two hours of processing time.

We first investigated the behavior of `B&C-PCP` for randomly generated graphs with 20 to 120 vertices. Each component has exactly two vertices and the edge density is 0.5. Five graph instances were generated for each number of vertices. Algorithm `B&C-PCP` was run three times for each instance, with different seeds for the pseudo-random number generator. The average lower and upper bounds for the number of colors over the 15 runs for each graph size are plotted in Figure 5 (a). Algorithm `B&C-PCP` solved to optimality all instances with up to 80 vertices. Regarding the larger instances, the gap between the lower and upper bounds was always exactly equal to one color.

In the second experiment, we randomly generated graphs with 90 vertices distributed over 45 components with two vertices each. The edge densities ranged from 0.1 to 0.9. Five graph instances were generated for each density. Algorithm `B&C-PCP` was run three times with different seeds for each instance. The average lower and upper bounds for the number of colors over the 15 runs for each graph size are plotted in Figure 5 (b). The number of instances solved to optimality for each edge density is given in Table 1. The most difficult instances are those with edge densities between 0.3 and 0.5. The number of instances solved to optimality increases with the edge density, because graphs with higher edge densities have more larger cliques that lead to better lower bounds. We also observe that the gap between the lower and upper bounds was never larger than one color in the instances not solved to optimality.

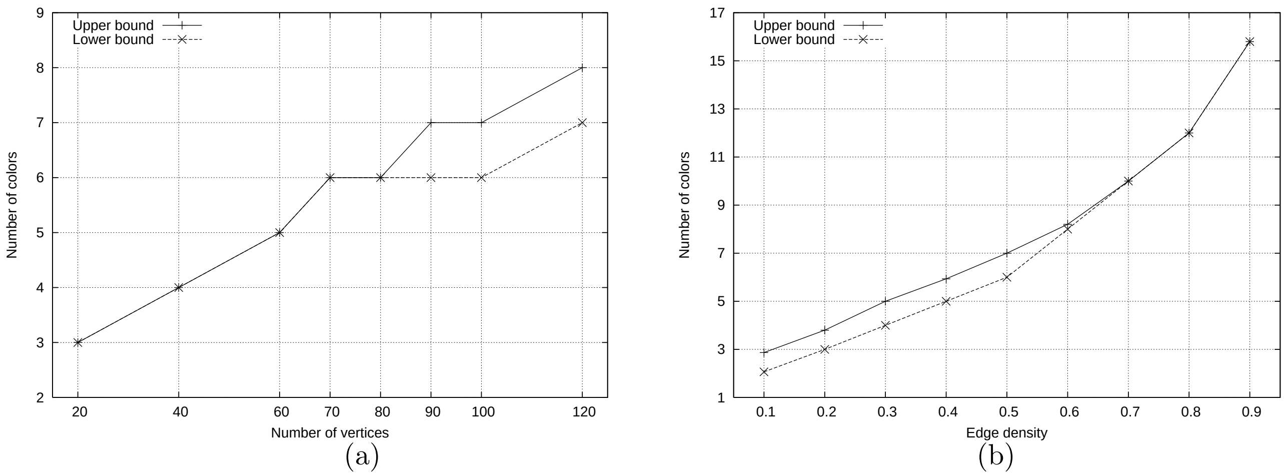


Figure 5: Average lower and upper bounds for the number of colors: (a) varying the number of vertices; (b) varying the edge densities.

Edge density	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Instances solved to optimality	5	3	—	1	—	12	15	15	15

Table 1: Instances solved to optimality (out of 15) for different edge densities.

Table 2 reports the contribution of external and internal cuts to the linear relaxation of the asymmetric formulation for the instances used in the two experiments above. The first three columns give the number of vertices, the number of edges, and the number of components for each group of five instances randomly generated in both experiments. The average value of the linear relaxation is displayed in the fourth column. The last two columns report the average relative improvement in the linear relaxation obtained by adding (i) external cuts and (ii) both external and internal cuts together. We observe that the external cuts improved the linear relaxation by 32.92% on average, while the internal cuts only slightly improved the results obtained with the external cuts.

Vertices	Edges	Components	Linear relaxation	Relative improvement (%) External cuts	Ext. and int. cuts
20	0.5	2	2.33	1.87	1.87
40	0.5	2	2.83	18.62	18.40
60	0.5	2	3.21	31.97	32.37
70	0.5	2	3.41	35.47	35.50
80	0.5	2	3.56	40.94	40.96
90	0.5	2	3.68	44.51	44.54
100	0.5	2	3.76	46.80	46.80
120	0.5	2	3.94	52.96	52.97
90	0.1	2	1.28	27.95	27.97
90	0.2	2	1.67	35.81	35.89
90	0.3	2	2.18	43.08	43.08
90	0.4	2	2.80	45.28	45.32
90	0.5	2	3.68	44.51	44.54
90	0.6	2	4.92	40.83	40.84
90	0.7	2	6.66	32.78	32.79
90	0.8	2	9.65	15.68	15.72
90	0.9	2	14.77	0.78	0.78
			Average	32.93	32.96

Table 2: Contribution of external and internal cuts to the linear relaxation.

In the third experiment, we generated partition coloring instances from graph coloring instances. The partitions were formed using exactly one vertex in each component. In this case, PCP reduces to a classical graph coloring problem. The results obtained by `B&C-PCP` are compared with those provided by the branch-and-cut algorithm proposed in [21]. The computational results for the most difficult instances in [21] are presented in Table 3. For each instance, the first three columns display the name of the instance, the number of vertices, and the number of edges in the graph, respectively. The next three columns give the upper bound, the lower bound, and the relative gap provided by `B&C-PCP`, respectively. The last three columns give the same data for the algorithm in [21], after two hours of processing time on a Sun ULTRA workstation (see Table 7 in [21]). We note that `B&C-PCP` performs better than the algorithm in [21] on instances with larger cliques (such as the DSJC instances), while it performs worse on instances with smaller cliques (such as the queen and Mycielsky instances). This is due to the fact that the clique cuts of `B&C-PCP` are stronger than those proposed in [21]. For the other instances, the performance of both algorithm were very similar.

The fourth and fifth experiments report computational results for PCP instances arising from those of the routing and wavelength assignment problem. First, we consider ring topologies as those in [14]. Three ring network topologies with 10, 15, and 20 communication nodes were considered. The traffic matrices were randomly generated, with the probability P_l that there is a lightpath request between a pair of communication nodes ranging from 0.1 to 1.0 by steps of 0.1. For each ring topology and each value of P_l, five traffic matrices

			B&C-PCP			Branch-and-cut [21]		
Name	$\|V\|$	$\|E\|$	UB	LB	Gap (%)	UB	LB	Gap (%)
DSJC125_5	125	3891	17	14	21.4	20	13	53.8
DSJC125_9	125	6961	43	43	0.0	47	42	11.9
DSJC250_1	250	3218	9	5	80.0	9	5	80.0
DSJC250_5	250	15668	29	15	93.3	36	13	176.9
DSJC250_9	250	27897	72	71	1.4	88	47	87.2
DSJR500_1c	500	121275	85	85	0.0	88	47	87.2
queen.9_9	81	2112	10	9	11.1	10	9	11.1
myciel6	95	755	7	4	75.0	7	5	40.0
myciel7	191	2360	8	4	100.0	8	5	60.0
1-Insertions-5	202	1227	6	3	100.0	6	4	50.0
1-Insertions-6	607	6337	7	3	133.3	6	4	50.0
2-Insertions-4	149	541	5	3	66.7	5	4	25.0
2-Insertions-5	597	3936	6	3	100.0	6	3	100.0
3-Insertions-4	281	1046	5	3	66.7	5	3	66.7
4-Insertions-3	79	156	4	3	33.3	4	3	33.3
4-Insertions-4	475	1795	5	3	66.7	5	3	66.7
1-FullIns-5	282	3247	6	4	50.0	6	4	50.0
2-FullIns-4	212	1621	6	5	20.0	6	5	20.0
2-FullIns-5	852	12201	7	5	40.0	7	5	40.0
3-FullIns-4	405	3524	7	6	16.7	7	6	16.7
3-FullIns-5	2030	33751	8	6	33.3	8	6	33.3
4-FullIns-4	690	6650	8	7	14.3	8	7	14.3
		Average			51.06			53.37

Table 3: Graph coloring instances.

were generated. All links are bidirectional and the lightpath requests are not necessarily symmetric, i.e., the number of lightpath requests from communication node i to j may be different from that from communication node j to i. Each RWA instance is transformed into a PCP instance $G = (V, E)$ and a partition Q that is built as following. Each component in Q corresponds to a lightpath and has one vertex $v \in V$ associated with each possible route for this lightpath. In the case of ring networks, there are exactly two alternative routes (clockwise and anti-clockwise) for each lightpath. There is one edge $e \in E$ between each pair of vertices whose associated routes share a common link. This transformation guarantees that the optimal solution of a PCP instance corresponds to the optimal solution of the respective RWA instance.

Algorithm `B&C-PCP` was applied to each of the above 150 instances. The computational results are presented in Table 4. The first two columns display the number of communication nodes in the ring and the probability P_l of the existence of each lightpath. The three next columns give the average number of vertices, edges, and components of the five corresponding PCP instances. The next columns display the average and the maximum number of evaluated nodes in the `B&C-PCP` tree, the average and the maximum absolute gaps $\lceil UB - LB \rceil$ (where UB and LB denote, respectively, the best upper and lower bounds for each instance), the average and the maximum relative gaps $(UB - LB)/LB$, and the number of instances solved to optimality (over the five instances associated with the same lightpath probability). All PCP instances that arose from rings with 10 and 15 communication nodes and 40 out of the 50 instances from rings with 20 communication nodes were solved to optimality. The relative gaps for the instances not solved to optimality within two hours of computational time were never larger than 6%.

Next, we consider the topology of the real network NSFnet with 14 communication nodes and 21 links, widely used in the literature for computational experiments. We first generated RWA instances in which a lightpath connecting communication node i to j exists with probabilities P_l ranging from 0.1 to 1.0 by steps of 0.1. As before, all links are bidirectional and the lightpath requests are not necessarily symmetric. Five RWA instances were generated for each probability value. Every RWA instance was transformed into a PCP instance by the `2-EDR` procedure proposed in [22]. First, `2-EDR` computes up to two alternative routes for each lightpath. Then, a PCP instance is built with one vertex for each alternative route. There is one edge between each pair of vertices whose associated routes share a common link. All vertices associated with the same lightpath are placed in the same component of the partition. We point out that in this case the optimal solution for the PCP instance is not guaranteed to be optimal for the respective RWA instance, because `2-EDR` does not generate all possible routes for each lightpath.

Algorithm `B&C-PCP` was applied to each of the 50 above instances. The computational

Ring nodes	P_l	Vertices avg.	Edges avg.	Comps. avg.	B&C nodes avg. (max.)	Gap avg. (max.)	Gap (%) avg. (max.)	Solved
10	0.1	17.6	63.6	8.8	1.0 (1)	– (–)	– (–)	5
10	0.2	35.6	242.8	17.8	1.2 (2)	– (–)	– (–)	5
10	0.3	49.6	477.2	24.8	1.0 (1)	– (–)	– (–)	5
10	0.4	67.6	886.2	33.8	1.0 (1)	– (–)	– (–)	5
10	0.5	83.2	1342.6	41.6	1.0 (1)	– (–)	– (–)	5
10	0.6	102.4	2024.0	51.2	1.0 (1)	– (–)	– (–)	5
10	0.7	127.2	3154.0	63.6	1.0 (1)	– (–)	– (–)	5
10	0.8	145.2	4119.2	72.6	1.0 (1)	– (–)	– (–)	5
10	0.9	164.4	5297.4	82.2	1.0 (1)	– (–)	– (–)	5
10	1.0	180.0	6360.0	90.0	1.0 (1)	– (–)	– (–)	5
15	0.1	47.2	457.8	23.6	1.0 (1)	– (–)	– (–)	5
15	0.2	92.8	1723.4	46.4	1.0 (1)	– (–)	– (–)	5
15	0.3	136.4	3729.4	68.2	1.0 (1)	– (–)	–(–)	5
15	0.4	178.0	6358.2	89.0	1.2 (2)	– (–)	– (–)	5
15	0.5	215.6	9316.8	107.8	1.0 (1)	– (–)	– (–)	5
15	0.6	255.6	13076.8	127.8	1.4 (3)	– (–)	– (–)	5
15	0.7	303.2	18492.6	151.6	1.0 (1)	– (–)	– (–)	5
15	0.8	339.6	23151.4	169.8	1.0 (1)	– (–)	– (–)	5
15	0.9	381.6	29285.2	190.8	1.2 (2)	– (–)	– (–)	5
15	1.0	420.0	35490.0	210.0	1.0 (1)	– (–)	– (–)	5
20	0.1	78.4	1251.0	39.2	2.2 (7)	– (–)	– (–)	5
20	0.2	155.2	4893.2	77.6	1.4 (3)	– (–)	– (–)	5
20	0.3	232.0	10925.6	116.0	8.0 (20)	0.4 (1)	2 (6)	3
20	0.4	308.4	19319.4	154.2	2.4 (8)	0.2 (1)	1 (4)	4
20	0.5	383.2	29848.8	191.6	1.0 (1)	– (–)	– (–)	5
20	0.6	454.0	41860.8	227.0	1.2 (2)	– (–)	– (–)	5
20	0.7	540.8	59507.2	270.4	1.0 (1)	– (–)	– (–)	5
20	0.8	612.8	76398.0	306.4	1.0 (1)	– (–)	– (–)	5
20	0.9	689.2	96702.8	344.6	1.0 (1)	0.4 (1)	1 (2)	3
20	1.0	760.0	117420.0	380.0	1.0 (1)	1.4 (2)	3 (4)	0

Table 4: Computational results for RWA instances associated with three ring networks.

Probability	Vertices avg.	Edges avg.	Components avg.	B&C nodes avg. (max.)	Gap avg. (max.)	Gap (%) avg. (max.)	Solved
0.1	26.4	36.0	22.2	1 (1)	– (–)	– (–)	5
0.2	52.6	149.0	39.0	1 (1)	– (–)	– (–)	5
0.3	79.0	359.6	58.6	1 (1)	– (–)	– (–)	5
0.4	103.0	634.4	75.4	1 (1)	– (–)	– (–)	5
0.5	125.2	947.6	92.0	1 (1)	– (–)	– (–)	5
0.6	151.2	1410.4	109.6	24.2 (75)	0.4 (1)	4 (11)	3
0.7	181.0	2093.8	130.4	4.0 (16)	0.2 (1)	2 (10)	4
0.8	204.6	2638.2	146.6	8.6 (18)	0.6 (1)	5 (9)	2
0.9	224.8	3118.2	165.6	4.6 (8)	0.6 (1)	5 (8)	2
1.0	248.8	3885.6	182.0	2.2 (4)	0.4 (1)	3 (8)	3

Table 5: Computational results for RWA instances associated with the NSFnet.

results are presented in Table 5. The first column displays the probability of the existence of each lightpath. The three next columns give the average number of vertices, edges, and components of the five PCP instances. The next columns display the average and the maximum number of evaluated nodes in the `B&C-PCP` tree, the average and the maximum absolute gaps, the average and the maximum relative gaps, and the number of instances solved to optimality (over the five instances associated with the same lightpath probability). All instances in which the probability of the existence of a lightpath is less than or equal to 0.5 were solved to optimality at the root node of the branch-and-cut tree. Furthermore, the lower bound provided by rounding up the objective value of the linear programming solution of each of these instances at the root node of the `B&C-PCP` is already the optimal value. Although denser instances are considerably harder, approximately half of them could be solved and the average gaps were never larger than 5%.

The largest instance solved to optimality within the 2-hour time limit is one of the five associated with the problem of routing and wavelength assignment in a ring network with 20 communication nodes and the probability of a lightpath existing between any two of them being equal to 90% (next to last line in Table 4). The corresponding PCP instance has 706 vertices, 101,600 edges, and 343 components.

6 Concluding remarks

In this work, we proposed an integer programming formulation for the partition coloring problem based on the model of representatives, together with valid inequalities and cutting plane heuristics, a primal constructive heuristic, a branching strategy, and a branch-and-cut algorithm for solving the partition coloring problem. The computational experiments were carried out not only on randomly generated instances, but also on test problems arising either from classical graph coloring instances or from instances of the problem of routing and wavelength assignment in all-optical WDM networks. The cutting plane procedures improved the value of linear relaxation by approximately 32% on average on the random instances. In addition, RWA instances with up to 706 vertices and 101,600 edges were solved to optimality within the 2-hour time limit. Furthermore, we also notice that the results obtained by B&C-PCP are competitive with those provided by the branch-and-cut algorithm in [21] for pure graph coloring instances.

Acknowledgments: We are grateful to two anonymous referees whose remarks on the first version of this paper contributed to improve the final text.

References

[1] E. Balas and M.W. Padberg, Set partitioning: A survey, *SIAM Review* 18 (1976), 710-760.

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

[3] R.J. Brown, Chromatic scheduling and the chromatic number problem, *Management Science* 19 (1972), 451-463.

[4] M. Campêlo, V. Campos, and R. Corrêa, On the asymmetric representatives formulation for the vertex coloring problem, *Electronic Notes in Discrete Mathematics* 19 (2005), 337-343.

[5] M. Campêlo, R.C. Corrêa, and Y.A.M. Frota, Cliques, holes and the vertex coloring polytope, *Information Processing Letters* 89 (2004), 159-164.

[6] A. Caprara and M. Fischetti, "Branch-and-cut algorithms: An annotated bibliography", *Annotated Bibliographies in Combinatorial Optimization*, M. Dell'Amico, F. Maffioli and S. Martello (editors), Wiley, 1997, pp. 45-63.

[7] A. Caprara and J.J.S. Gonzáles, Separating lifted odd-hole inequalities to solve the index selection problem, *Discrete Applied Mathematics* 92 (1999), 111-134.

[8] J.S. Choi, N. Golmie, F. Lapeyrere, F. Mouveaux, and D. Su, A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case, *Proceedings of the 7th International Conference on Optical Communication and Networks*, Paris, 2000, pp. 1109-1115.

[9] T.A. Feo and M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, *Operations Research Letters* 8 (1989), 67-71.

[10] T.A. Feo, M.G.C. Resende, and S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set, *Operations Research* 42 (1994), 860-878.

[11] R. Figueiredo, V. Barbosa, N. Maculan, and C. Souza, Acyclic orientations with path constraints, *RAIRO Operations Research* 42 (2008), 455-467.

[12] F. Glover and M. Laguna, *Tabu search*, Kluwer, 1997.

[13] K.L. Hoffman and M. Padberg, Solving airline crew scheduling problems, *Management Science* 39 (1993), 657-682.

[14] B. Jaumard, Network and traffic data sets for optical network optimization, Online publication available at http://users.encs.concordia.ca/~bjaumard, last visited on January 3th, 2008.

[15] B. Jaumard, C. Meyer, and X. Yu, Of how much wavelength conversion allows to reduce the blocking rate?, Online publication available at http://users.encs.concordia.ca/~bjaumard/Publications/2005/Multi_Hop_RWA_2005.pdf, last visited on September 24th, 2008.

[16] S.M. Korman, "The graph-coloring problem", *Combinatorial Optimization*, N. Christophides, P. Toth and C. Sandi (editors), Wiley, New York, 1979, pp. 211-235.

[17] M. Kubale and B. Jackowski, A generalized implicit enumeration algorithm for graph coloring, *Communications of the ACM* 28 (1985), 412-418.

[18] G. Li and R. Simha, The partition coloring problem and its application to wavelength routing and assignment, *Proceedings of the 1st Workshop on Optical Networks* (CDROM), Dallas, 2000.

[19] D. Matula, G. Marble, and J. Isaacson, "Graph coloring algorithms", *Graph Theory and Computing*, R.C. Read (editor), Academic Press, New York, 1972, pp. 109-122.

[20] A. Mehrotra and M.A. Trick, A column generation approach for graph coloring, *Operations Research* 42 (1996), 344-354.

[21] I. Méndez-Díaz and P. Zabala, A branch-and-cut algorithm for graph coloring, *Discrete Applied Mathematics* 154 (2006), 826-847.

[22] T.F. Noronha and C.C. Ribeiro, Routing and wavelength assignment by partition coloring, *European Journal of Operational Research* 171 (2006), 797-810.

[23] M.G.C. Resende and C.C. Ribeiro, A GRASP with path-relinking for private virtual circuit routing, *Networks* 41 (2003), 104-114.

[24] M.G.C. Resende and C.C. Ribeiro, "Greedy randomized adaptive search procedures", *Handbook of Metaheuristics*, F. Glover and G. Kochenberger (editors), Kluwer, 2003, pp. 219-249.

[25] M.G.C. Resende and C.C. Ribeiro, "GRASP with path-relinking: Recent advances and applications", *Metaheuristics: Progress as real problem solvers*, T. Ibaraki, K. Nonobe and M. Yagiura (editors), Springer, 2005, pp. 29-63.

[26] D. Welsh and M. Powell, An upper bound to the chromatic number of a graph and its application to time-table problems, *Computer Journal* 10 (1967), 85-86.