UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Victor Deluca Almirante Gomes

A Line Graph Reformulation for the Rainbow Spanning Forest Problem

Belo Horizonte
2024

Victor Deluca Almirante Gomes

A Line Graph Reformulation for the Rainbow Spanning Forest Problem

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Alexandre Salles da Cunha

Belo Horizonte
2024

Gomes, Victor Deluca Almirante.

G633l A Line graph reformulation for the rainbow spanning forest problem [recurso eletrônico] / Victor Deluca Almirante Gomes – 2024.
1 recurso online (79 f. il., color.) : pdf.

Orientador: Alexandre Salles da Cunha.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 53-55

1. Computação - Teses. 2. Otimização combinatória – Teses. 3. Programação inteira - Teses. 4. Floresta Aleatória – Teses. 4. Árvores (Teoria dos grafos) - Teses. I. Cunha, Alexandre Salles da. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*61(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A Line Graph Reformulation for the Rainbow Spanning Forest Problem

VICTOR DELUCA ALMIRANTE GOMES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Documento assinado digitalmente
gov.br
ALEXANDRE SALLES DA CUNHA
Data: 28/03/2024 15:12:02-0300
Verifique em https://validar.iti.gov.br

PROF. ALEXANDRE SALLES DA CUNHA - Orientador
Departamento de Ciência da Computação - UFMG

Documento assinado digitalmente
gov.br
ABILIO PEREIRA DE LUCENA FILHO
Data: 29/03/2024 09:18:09-0300
Verifique em https://validar.iti.gov.br

PROF. ABÍLIO PEREIRA DE LUCENA FILHO
Programa de Engenharia de Sistemas e Computação - COPPE - UFRJ

Documento assinado digitalmente
gov.br
GERALDO ROBSON MATEUS
Data: 03/04/2024 10:08:58-0300
Verifique em https://validar.iti.gov.br

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

Documento assinado digitalmente
gov.br
YURI ABITBOL DE MENEZES FROTA
Data: 03/04/2024 13:13:47-0300
Verifique em https://validar.iti.gov.br

PROF. YURI ABITBOL DE MENEZES FROTA
Departamento de Ciência da Computação - UFF

Belo Horizonte, 20 de março de 2024.

Acknowledgments

I shall thank FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais) for supporting the Operations Research Laboratory (LAPO) via project RED-00119-21 - Modelos e algoritmos de otimização para localização, dimensionamento e operação eficientes de usinas de energia eólicas e solares integradas.

"If you hold someone's hand, that someone will hold another hand
That way, you can grasp more than just two things
However, don't forget this one thing:
You can't choose what to hold onto, except for the two hands you possess"
(Unknown Author)

Resumo

Seja $G(V, E)$ um grafo conexo e não direcionado, composto pelo conjunto de vértices V e de arestas E, e seja $L = \{L_1, L_2, ..., L_h\}$ um conjunto finito de cores. A cada aresta $e \in E$ é associada uma cor $l \in L$. Uma floresta geradora $F(V, E_F) \subset G$ é dita iridescente se cada componente conexa em F é composta por arestas de cores diferentes. O problema da Floresta Geradora Iridescente consiste então em encontrar uma floresta geradora iridescente para G com o mínimo número de componentes conexas.

Neste trabalho, propomos uma formulação de Programação Inteira baseada em grafos linha para o problema, além de desigualdades válidas com o propósito de gerar limites duais mais fortes. Implementamos então um algoritmo *Branch-and-cut* para resolver a formulação proposta. Experimentos mostram que a formulação proposta apresenta limites duais significativamente melhores que as formulações encontradas na literatura.

Palavras-chave: otimização combinatória; problema da floresta geradora iridescente; grafos linha; branch-and-cut.

Abstract

Let $G(V, E)$ be a connected and undirected graph, with sets of vertices V and edges E, and let $L = \{L_1, L_2, ..., L_h\}$ be a finite set of colors. Each edge $e \in E$ is colored with a color $l \in L$. A spanning forest $F(V, E_F) \subset G$ is a rainbow spanning forest if no connected component in F contains two edges of the same color. The Rainbow Spanning Forest Problem then consists of finding a rainbow spanning forest of G with the minimum number of connected components.

In this work, we propose an Integer Programming formulation based on line graphs for the problem, as well as valid inequalities in order to achieve better bounds. We also present a Branch-and-cut algorithm to solve the proposed formulation. Our experiments show that this new formulation yields tighter bounds than other formulations in the literature.

Keywords: combinatorial optimization, rainbow spanning forest problem, line graphs, branch-and-cut.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Edge-colored graphs are an important tool to model complex problems in network optimization. Depending on the application, colors may represent different service providers, connection types or transmission frequencies. Hence, problems defined over edge-colored graphs have garnered attention from researchers in recent years.

The Rainbow Spanning Forest Problem is one of those problems. Let $G(V, E, L)$ be a connected and undirected edge-colored graph, such that every edge $e \in E$ is colored with a color $l(e) \in L$. A rainbow spanning forest of G is any acyclic subgraph, i.e. a forest, $F(V, E_f, L), E_f \subseteq E$, such that no connected component of F contains two edges of the same color. The Rainbow Spanning Forest Problem (RSFP) then consists of finding a rainbow spanning forest with the minimum number of connected components [1].

The RFSP is a NP-hard problem, being a generalization of the Minimum Labeling Spanning Tree [20]. Solution approaches found in the literature include techniques such as approximation algorithms [1], heuristics [14, 15, 16] and exact algorithms [2, 14]. In this work, we provide a novel Integer Programming formulation based on line graphs and show that, for all instances proposed in the literature, the new dual bounds are tight.

1.1 Contributions

The main contributions of this dissertation are:

- A line graph reformulation for the Rainbow Spanning Forest Problem;
- Valid inequalities for the convex hull of integer feasible solutions to the problem, as well as lifting strategies for certain inequalities;
- Strong lower bounds for small and medium-sized instances of the literature, which suggest strong lower bounds for larger instances as well;
- A Branch-and-cut algorithm for the proposed formulation;

1.2 Organization of this Dissertation

The remainder of the text is structured as follows. Chapter 2 contains a literature review on the RSFP and related problems. In Chapter 3, we present a novel Integer Programming Formulation (IFP) for the RSFP based on line graphs, as well as valid inequalities and lifting strategies. In Chapter 4, we describe a cutting plane algorithm that approximates the dual bounds provided by the proposed formulation. We then embed the cutting plane algorithm in a Branch-and-cut scheme and analyse the results. Finally, Chapter 5 contains our concluding remarks and future research suggestions.

Chapter 2

Literature Overview

Literature on the Rainbow Spanning Forest Problem (RSFP) is relatively scarce [15]. The RSFP was introduced in [20], where it was shown to be NP-hard even for instances with only two colors. The RSFP is studied in greater detail in [1], which shows polynomially solvable cases. The authors show that the RSFP remains NP-hard in trees, but can be solved in polynomial time in paths, cycles and in cases where the optimal solution is 1, i.e., when the problem is reduced to a Rainbow Spanning Tree Problem [17].

An Integer Programming (IP) Formulation for the problem is presented by Carrabs et al. in [2]. In their formulation, binary variables $\mathbf{y}$ are associated to every vertex and binary variables $\mathbf{x}$ are associated to every edge of the input graph. In addition, both sets of variables are indexed by the number of possible connected components in a viable solution. That is, a variable y_v^k takes the value of 1 if vertex v belongs to component k. Likewise, a variable x_e^k will take the value of 1 if edge e belongs to component k. In order for such formulation to work, a previously known upper bound on the minimum number of components of a rainbow forest is required. The authors experiment with the number of vertices n as a trivial upper bound, as well as a tighter upper bound obtained by a greedy heuristic.

An improvement over the formulation in [2] is introduced by Frota et al. in [14]. The authors add separate variables to represent trivial trees, that is, connected components consisting of a single vertex. In doing so, the authors are able to reduce the number of variables by half, as a trivial upper bound of $\frac{n}{2}$ can be used for the number of connected components. Though the modifications are simple, the improvements are significant, with speedups as high as 87%. The authors also improve on the heuristic primal side, using GRASP [12] to find better upper bounds for larger instances.

As far as RSFP heuristics are concerned, researchers have employed GRASP [14], bee colony algorithms [15] and genetic algorithms [16]. A 2-factor approximation algorithm for cases where the input graph is a tree was described in [1].

The RSFP generalizes the Rainbow Spanning Tree Problem [5, 17], a decision problem where the objective is to determine whether or not the input graph has a spanning tree where no pair of edges share the same color. The Rainbow Spanning Tree problem is solvable in polynomial time using a greedy augmentation algorithm, described in detail

in [17]. Naturally, this is equivalent to verifying whether or not the optimal solution to the Rainbow Spanning Forest problem is 1.

The same greedy algorithm described in [17] can be used to solve in polynomial time the more general Maximum Label Spanning Tree Problem, an optimization problem that aims to find a spanning tree in an edge-colored graph with the largest possible number of colors. On the other hand, the closely related Minimum Label Spanning Tree Problem, which aims to find a spanning tree with the smallest possible number of colors, is shown to be NP-hard in the same paper.

Other problems that focus on rainbow structures in edge-colored graphs are the Rainbow Cycle Cover Problem [24], the Rainbow Connectivity Problem [4, 6] and the Rainbow Steiner Tree Problem [13]. In the Rainbow Cycle Cover Problem (RCCP), the goal is to cover the graph with the minimum number of cycles, rather than trees. The RCCP is closely tied to the RSFP, as the current best formulation for the RSFP [14] is closely related to a RCCP formulation provided in [23]. Studies of the RCCP primarily focus on Integer Programming formulations [23, 24], including a Branch-and-Price algorithm being described in [27].

The Rainbow Connectivity Problem (RCP) is the decision problem of determining whether or not a rainbow path, i.e. a path where all edge colors are different, exists between any two vertices of an edge-colored graph. It is a NP-complete problem, with an exponential algorithm on the number of colors described in [25]. Originally, that algorithm was also exponential in space. However, improvements introduced in [28] made it polynomial in space. Finally, the Rainbow Steiner Tree Problem is the problem of finding a minimum weight Steiner Tree that uses at most one edge for each color. An Integer Programming formulation and heuristics for the Rainbow Steiner Tree Problem are proposed in [13].

Other problems defined over edge-colored graphs include the Maximum Labeled Clique Problem [3], the Minimum Labeling Steiner Tree Problem [5] and the Colored Traveling Salesman Problem [19]. In the Maximum Labeled Clique Problem, the goal is to find a maximum clique, not necessarily rainbow, containing at most k colors. Approaches to solve this problem focus in Integer Programming formulations and techniques [3, 22].

As for the Minimum Labeling Steiner Tree Problem, the goal is to find a Steiner Tree that uses the lowest possible number of colors. Heuristics for the Minimum Labeling Steiner Tree Problem are described in [7, 10, 8]. Finally, for the Colored Traveling Salesman Problem, the goal is to find a minimum cost cycle cover for a weighted edge-colored graph, where each cycle is assigned to a salesman and each vertex may only be visited by certain salesmen. Two Genetic Algorithms for the problem are proposed in [19].

2.1 Preliminary Definitions and Notation

Before proceeding, we outline the notation employed throughout the remainder of this work. An edge-colored graph $G(V, E, L)$ is a graph represented by a set of vertices V, a set of edges E and a set of colors/labels $L = \{L_1, L_2, ..., L_h\}$, $h = |L|$. For simplicity, we define $n = |V|$ and $m = |E|$. Each edge $e \in E$ is colored with one color $k \in L$. We equivalently say that $l(e) = k$.

In an undirected graph, we say that an edge $e = \{u, v\}$ is *incident* to vertices u and v. The set of edges incident to a vertex u is denoted by $\delta(u)$. Sometimes, we wish to single out the set of all edges incident to u with a specific color k. This set is denoted by $\delta_k(u) = \{e \in \delta(u) : l(e) = k\}$.

This concept can be extended to sets of vertices, $S \subseteq V$. The set of edges with exactly one endpoint incident to S is denoted by $\delta(S)$, and the set of edges incident to S with a specific color k is $\delta_k(S)$. Finally, we denote by $E(S)$ the set of edges with both endpoints in S, that is, $E(S) = \{e = \{u, v\} \in E : u, v \in S\}$ and $E_k(S) = \{e = \{u, v\} \in E : u, v \in S, l(e) = k\}$.

A tree is any connected graph containing no cycles. A trivial tree, or a singleton tree, is a tree consisting of a single vertex and no edges. A forest is a set of trees, that is, an acyclic, but not necessarily connected graph. We occasionally denote a forest by $F = \{T_1, T_2, \ldots, T_n\}$, indicating that the forest consists of n trees, $T_1, \ldots, T_n$.

2.2 Mathematical Formulations from the Literature

For the sake of completeness, we present the formulation proposed by Carrabs et al. [2] and the improvements introduced by Frota et al. in [14].

Let $\alpha = \{\alpha_c \in \mathbb{B} : c \in \{1, \ldots, \overline{c}\}\}$ be a set of binary variables, indicating whether or not the solution involves at least c components. With some abuse of notation, we use "component c" to refer to the graph $G(V^c, E^c, L^c)$ associated with a component c.

Let variables $\mathbf{y} = \{y_v^c \in \mathbb{B} : v \in V\}$ indicate whether or not vertex v belongs to component c, that is, $y_v^c = 1$ applies in case v belongs to c and $y_v^c = 0$ holds otherwise. Similarly, let $\mathbf{x} = \{x_e^c \in \mathbb{B} : e \in E\}$ indicate whether or not edge e belongs to component c. Then, the formulation for the Rainbow Spanning Forest Problem proposed in [2] is:

$$z^* = \min \left\{ \sum_{c=1}^{\overline{c}} \alpha_c : (\alpha, x, y) \in P_1 \cap (\mathbb{B}^{\overline{c}}, \mathbb{B}^{m\overline{c}}, \mathbb{B}^{n\overline{c}}) \right\} \tag{2.1}$$

where P_1 is the polytope defined by constraints (2.2)-(2.17):

$$\sum_{v \in V} y_v^c \leq (|L|+1)\alpha_c \qquad c = \{1, \ldots, \overline{c}\}, \tag{2.2}$$

$$\alpha_c \leq \sum_{v \in V} y_v^c \qquad c = \{1, \ldots, \overline{c}\}, \tag{2.3}$$

$$\sum_{c=1}^{\overline{c}} y_v^c = 1 \qquad v \in V, \tag{2.4}$$

$$x_e^c \leq y_v^c \qquad v \in V, e \in \delta(v), \tag{2.5}$$

$$\sum_{e \in E_k(V)} x_e^c \leq \alpha_c \qquad c = \{1, \ldots, \overline{c}\}, k \in L, \tag{2.6}$$

$$\sum_{c=1}^{\overline{c}} \sum_{e \in E(S)} x_e^c \leq |S| - 1 \qquad S \subseteq V, |S| \geq 2, \tag{2.7}$$

$$\sum_{e \in E} x_e^c = \sum_{v \in V} y_e^c - \alpha_c \qquad c = \{1, \ldots, \overline{c}\}, \tag{2.8}$$

$$\alpha_{c+1} \leq \alpha_c \qquad c = \{1, \ldots, \overline{c}-1\}, \tag{2.9}$$

$$y_1^1 = 1, \tag{2.10}$$

$$y_v^c \leq \sum_{w < v} y_w^{c-1} \qquad v \in V \backslash \{1\}, c = \{3, \ldots, \overline{c}\}, \tag{2.11}$$

$$y_v^c \leq \alpha_c \qquad v \in V, c = \{1, \ldots, \overline{c}\}, \tag{2.12}$$

$$\sum_{e \in \delta_k(v)} x_e^c \leq y_v^c \qquad v \in V, c = \{1, \ldots, \overline{c}\}, k \in L, \tag{2.13}$$

$$\sum_{c=1}^{\overline{u}} \left\{ x_e^c + \sum_{f \in \{\delta_k(u) \cup \delta_k(v)\}} x_f^c \right\} \leq 2 \qquad e = \{u, v\} \in E, k \in L \backslash \{l(e)\}, \tag{2.14}$$

$$\alpha_c \in [0, 1] \qquad c = \{1, \ldots, \overline{c}\}, \tag{2.15}$$

$$y_v^c \in [0, 1] \qquad v \in V, c = \{1, \ldots, \overline{c}\}, \tag{2.16}$$

$$x_e^c \in [0, 1] \qquad e \in E, c = \{1, \ldots, \overline{c}\}. \tag{2.17}$$

The objective function (2.1) minimizes the number of (rainbow) components. Constraints (2.2) and (2.3) couple variables α_c and y_v^c so that a component will be used in the solution iff it contains one or more vertices. Since each rainbow tree must not include more than $|L|$ edges, inequalities (2.2) also state that the number of vertices assigned to a given component cannot exceed $|L|+1$. Constraints (2.4) enforce that the solution spans the graph, by requiring that every vertex must belong to exactly one component.

Constraints (2.5) couple edge variables $x^c_{u,v}$ with vertex variables y^c_u and y^c_v, that is, a edge cannot be present in a component unless the vertices it connects are present too. Constraints (2.6) enforce that all edges of a component have different colors, ensuring the rainbow property. Constraints (2.7) are subtour elimination constraints [9]. Alongside cardinality constraints (2.8) and box constraints (2.15)-(2.17), they ensure that every component induces a tree. Note that (2.7) number exponentially many constraints.

In this formulation, the same set of connected components may be arranged in several ways. Therefore, symmetry-breaking constraints (2.9), (2.10) and (2.11) are also included. Constraints (2.9) ensure that a component $c+1$ will only be included in the solution if the component c is also included. Constraints (2.10)-(2.11) define a lexicographic ordering for the components, that is, a vertex v may only be used in component $c+1$ if at least one vertex $w < v$ is used in component c.

Constraints (2.12)-(2.14) are not necessary to generate a valid solution for the RSFP. However, they play an important role in improving the Linear Programming Relaxation (LPR) bounds provided by that formulation.

Constraints (2.12) couple vertex variables y^c_v with component variables α_c, so that a vertex v may only be included in component c if $\alpha_c = 1$. Constraints (2.13) forbid the selection of two similarly colored edges, incident to the same vertex. That is, for any vertex $v \in V^c_T$ in a tree $T(V^c_T, E^c_T)$, at most one edge incident to v with color k may belong to the component (for any color $k \in L$). Constraints (2.14) are analogous to (2.13). However, they forbid the selection of similarly colored edges whenever they are connected by another edge (rather than sharing a common vertex). Finally, constraints (2.15)-(2.17) are box constraints for variables α, $\mathbf{x}$ and $\mathbf{y}$, respectively.

Constraints (2.2)-(2.17), alongside the objective function (2.1), are the original formulation for the Rainbow Spanning Forest Problem, proposed in [2]. Note that the formulation depends on an upper bound for $\bar{c}$. In the absence of a better bound, the trivial upper bound $\bar{c} = n - 1$ is used.

Frota et al. [14] note that the number of variables and constraints is directly proportional to $\bar{c}$. They propose a more compact reformulation by distinguishing between singleton components and non-trivial ones. Variables $\phi = \{\phi_v \in \mathbb{B} : v \in V\}$ are used to indicate whether or not the v-th vertex is included in the solution as a singleton tree, while variables $\alpha = \{\alpha_c \in \mathbb{B} : c \in \{1, \ldots, \bar{c}\}\}$ indicate whether or not a c-th non-trivial component is used in the solution. The formulation then becomes:

$$z^* = \min \left\{ \sum_{c=1}^{\bar{c}} \alpha_c + \sum_{v \in V} \phi_v : (\alpha, \phi, x, y) \in P_2 \cap (\mathbb{B}^{\bar{c}}, \mathbb{B}^n, \mathbb{B}^{m\bar{c}}, \mathbb{B}^{n\bar{c}}) \right\}, \tag{2.18}$$

where P_2 is the polytope defined by constraints (2.2), (2.6) - (2.9), (2.11), (2.13), (2.15) - (2.17), as well as constraints (2.19)-(2.24), described next.

$$\sum_{v \in V} y_v^c \geq 2\alpha_c \qquad c = \{1, \ldots, \overline{c}\}, \tag{2.19}$$

$$\sum_{c=0}^{c-1} y_v^c = 1 - \phi_v \qquad v \in V, \tag{2.20}$$

$$2x_e^c \leq y_u^c + y_v^c \qquad e = \{u, v\} \in E, c = \{1, \ldots, \overline{c}\}, \tag{2.21}$$

$$\sum_{c=v+1}^{\overline{c}} y_v^c = 0 \qquad v \in V, v < \overline{c}, \tag{2.22}$$

$$\alpha_0 = 1, \tag{2.23}$$

$$\phi_v \in [0, 1] \qquad v \in V. \tag{2.24}$$

In this reformulation, constraints (2.19) enforce that variables α_c must be associated with non-trivial trees, replacing (2.15). As such, any nontrivial component must include at least two vertices. Constraints (2.20) replace (2.4), enforcing that the resulting forest must span the graph. Constraints (2.21) replace (2.5) for non-trivial components. Constraints (2.22) are symmetry-breaking constraints, ensuring that a vertex v cannot belong to a non-trivial tree c such that $c > v$. Constraint (2.23) is an optimality condition based on the simple observation that any graph with a non-empty set of edges has at least one non-trivial rainbow subtree. Constraints (2.24) are box constraints for the singleton trees ϕ_v.

By making a clear distinction between trivial and non trivial components, the authors are able to define a tighter trivial lower bound for the number of non-trivial components, $\overline{c} = \lfloor \frac{n}{2} \rfloor$, resulting in a more compact formulation. In doing so, they significantly reduce the time required by a branch-and-cut algorithm to find an optimal solution. Despite that, the resulting LPR bounds are not very tight. In the following chapters, we present a reformulation that yields strong LPR bounds, and present IP techniques to exploit these bounds.

Chapter 3

A Line Graph Reformulation for the RSFP

Rather than formulating the RSFP over the original input graph, the formulation proposed in this chapter is based on an extended line graph associated with G, in order to represent rainbow spanning forests in a more convenient way. We therefore devote the following sections to the discussion of line graphs, the extended line graph and how the properties of rainbow trees may be represented in such graphs. In the sequence, we present our IP formulation.

3.1 Line Graphs and Vertex-Colored Rainbow Trees

Let $G(V, E, L)$ be the input graph for the RSFP. The line graph of G is a graph $G'(E, E', L)$ whose vertices are in a one-to-one correspondence with the edges of G. Two vertices of G' are adjacent if and only if the corresponding edges in G share a vertex in common [26]. Since no tree in a rainbow forest contains two edges with the same color, edges connecting similarly colored vertices are not included in E'. For edge-colored input graphs, the corresponding line graph G' is a vertex-colored graph.

Figure 3.1 illustrates the concept of line graphs in the context of edge-colored graphs, using the example graph G_0 on the left side of the figure. A vertex in the line graph is represented by the notation "$\{u, v\}, k$", where $\{u, v\} \in E$ is the edge from the original graph represented by that vertex and $k \in L$ is its color.

A rainbow tree T' (not necessarily spanning) of G' is analogous to a rainbow tree in edge-colored graphs, that is, a subtree of G' where every vertex has a different color. Furthermore, T' is a rainbow tree if, and only if, the corresponding tree T in the original graph is rainbow too. Note that the reciprocal statement is true as long as T contains more than one vertex, that is, T can be represented in a line graph. Figure 3.2 illustrates the mapping between a rainbow tree in G and its representation under the line graph.

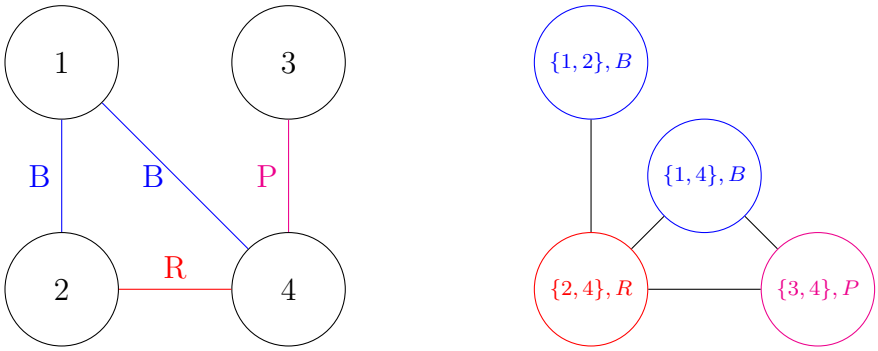

Figure 3.1: Example graph G_0 (left) and its corresponding line graph G'_0 (right). The set of colors is $L = \{B, P, R\}$.

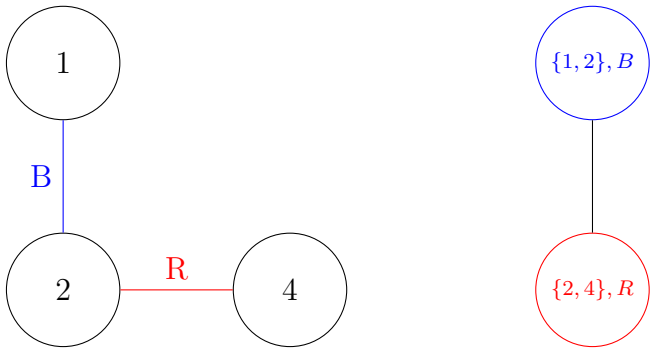

Figure 3.2: A non-trivial rainbow tree of the example graph G_0 (left) implies a rainbow tree on the line graph G'_0 (right), and vice-versa.

3.2 The Extended Line Graph

The line graph G' preserves all connectivity properties of G, but does not share the same RSFP solutions, as singleton components in G have no equivalent in G'. We therefore extend the line graph into $\hat{G}(\hat{V}, \hat{E}, L)$, so that:

- The set of vertices $\hat{V}$ includes:
  - An artificial vertex 0, the root of the tree of $\hat{G}$ we look for.
  - A vertex $v_e : e \in \{1, \ldots, m\}$ for each edge e of the original input graph. Let $\hat{V}_1$ denote these vertices.
  - An artificial vertex $v_{i+m} : i \in \{1, \ldots, n\}$ for each vertex i of the original input graph. Let $\hat{V}_2$ denote these vertices.
- And the set of edges $\hat{E}$ is partitioned as $\hat{E}_1 \cup \hat{E}_2 \cup \hat{E}_3$, where:
  - Edges in set $\hat{E}_1 = \{\{0, v_{i+m}\} : i = 1, \ldots, n\}$ represent connections between the artificial vertex 0 and $v_{i+m} : i = 1, \ldots, n$. These edges are included in the tree

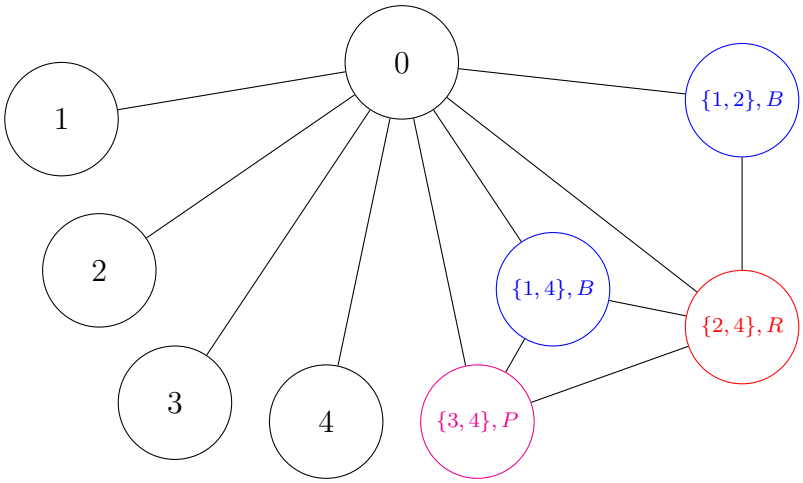

Figure 3.3: Extended line graph of the example graph G_0. See Figure 3.1 for the original graph.

if vertex i alone defines a trivial rainbow component, i.e., if i is the only vertex in a connected component of the rainbow forest.

- Edges $\hat{E}_2 = \{\{0, v_e\} : e = 1, \ldots, m\}$ represent connections between the artificial vertex 0 and each other vertex of $\hat{V}$.
- Edges $\hat{E}_3 = \bigcup_{i \in V} \{\{p, q\} : p, q \in \delta(i), l(p) \neq l(q)\}$ represent connections between pairs of adjacent edges of G, colored with diferent colors.

Note that the subgraph of $\hat{G}$ induced by vertices $\hat{V}_1$ and edges $\hat{E}_3$, that is, $G'(\hat{V}_1, \hat{E}_3, L)$ is the line graph of G. Figure 3.3 illustrates the extended line graph of our example graph, G_0. Vertices $\{1, 2\}, B$ and $\{1, 4\}, B$ are not connected by an edge, since they have the same color.

Now, consider a forest $\hat{F}(V_{\hat{F}}, E_{\hat{F}}, L)$, $V_{\hat{F}} \in \hat{V}$, $E_{\hat{F}} \in \hat{E}$ with the following properties:

- $\hat{F}$ spans G, that is, every vertex of G is either an isolated vertex in $\hat{V}_2 \cap V_{\hat{F}}$ or an endpoint of an edge in $\hat{V}_1 \cap V_{\hat{F}}$.
- Every connected component of $\hat{F}$ is either a vertex-colored rainbow tree or an isolated vertex, thus inducing a trivial rainbow tree.

The forest $\hat{F}$ corresponds to a rainbow spanning forest in G, as its vertices span the graph and every tree in $\hat{F}$ is rainbow. We say that $\hat{F}$ is an extended rainbow spanning forest of G. Likewise, every rainbow spanning forest in G has at least one equivalent in $\hat{G}$.

Note that, because the artificial vertex 0 connects to every other vertex of $\hat{G}$, any extended rainbow spanning forest can be transformed into a tree by adding vertex 0 and one edge between 0 and each connected component. In this case, the optimal solution to the Rainbow Spanning Forest problem is the number of edges incident to the artificial

Figure 3.4: A rainbow spanning forest of sample graph G_0 (left) and one equivalent subtree in $\hat{G}_0$ (right). This subtree is achieved by connecting one vertex of every component of an Extended Rainbow Spanning Forest to the vertex 0. In this case, the components are $\{\{\{1,2\},B\},\{\{2,4\},R\}\}$ and the singleton tree $\{3\}$. Refer to Figure 3.1 for the original graph G_0 and Figure 3.3 for the extended line graph $\hat{G}_0$.

vertex 0. Figure 3.4 illustrates a rainbow spanning forest and one equivalent subtree in the line graph.

3.3 The Extended Line Graph Reformulation

We now present an Integer Programming formulation for the Rainbow Spanning Forest Problem defined over the extended line graph. In this formulation, a solution will be an extended rainbow spanning forest augmented with the vertex 0, as defined in the previous section. Let $\hat{G}(\hat{V},\hat{E},L)$ be the extended line graph associated with the input graph G. We define $\hat{\delta}(i)$ as the set of edges of $\hat{E}$ incident to $i \in \hat{V}$ and $\hat{E}(S)$ as the subset of edges of $\hat{E}$ with both endpoints in $S \subseteq \hat{V}$. Also, if $S \subseteq \hat{V}_1$, let $K(S)$ denote the size of the largest set of similarly colored vertices in S, that is, $K(S) = \max_{k\in L} |\{v \in S : l(v) = k\}|$.

We define binary variables $\mathbf{y} = \{y_i : i \in \hat{V}\}$, and $\mathbf{x} = \{x_e : e \in \hat{E}\}$ for the model. Variables y_i indicate whether or not a vertex i belongs to the solution tree. A value of $y_i = 1$ indicates that the vertex i is used in the solution, while a value of $y_i = 0$ states otherwise. Likewise, variables x_e indicate whether or not an edge e belongs to the solution tree. Then, the extended line graph formulation for the RSFP is:

$$z^* = \min \left\{ \sum_{f\in\hat{E}_1\cup\hat{E}_2} x_f : (x,y) \in P_3 \cap (\mathbb{B}^{|\hat{V}|}, \mathbb{B}^{|\hat{E}|}) \right\}, \tag{3.1}$$

where P_3 is the polytope defined by constraints (3.2)-(3.12).

$$y_0 = 1, \tag{3.2}$$

$$x_f \leq y_i \qquad f \in \hat{\delta}(i), i \in \hat{V}, \tag{3.3}$$

$$x_{0,i+m} + \sum_{e \in \delta(i)} x_{0,e} + \sum_{\{p,q\} \in \hat{E}_3 : p \in \delta(i) \text{ or } q \in \delta(i)} x_{p,q} \geq 1 \qquad i \in V, \tag{3.4}$$

$$\sum_{f \in \hat{E}} x_f = \sum_{i \in \hat{V}} y_i - 1, \tag{3.5}$$

$$y_{i+m} + y_e \leq 1 \qquad i \in V, e \in \delta(i), \tag{3.6}$$

$$x_{0,i+m} + \sum_{e \in \delta(i)} x_{0,e} \leq 1 \qquad i \in V, \tag{3.7}$$

$$\sum_{f \in \hat{E}(S)} x_f \leq \sum_{i \in S \setminus \{j\}} y_i \qquad S \subseteq \hat{V}, j \in S, \tag{3.8}$$

$$\sum_{f \in \hat{E}(S)} x_f \leq |S| - K(S) \qquad S \subseteq \hat{V}_1, \tag{3.9}$$

$$x_{0,i+m} + \sum_{e \in \delta(i)} y_e - \sum_{\{p,q\} \in \hat{E}_3 : p \in \delta(i), q \in \delta(i)} x_{p,q} = 1 \qquad i \in V, \tag{3.10}$$

$$y_i \in [0,1] \qquad i \in \hat{V} \setminus \{0\}, \tag{3.11}$$

$$x_f \in [0,1] \qquad f \in \hat{E}. \tag{3.12}$$

A solution to the formulation above is a tree of $\hat{G}$ that necessarily spans artificial vertex 0, as enforced by Constraint (3.2). If the edges incident to 0 in that tree are dropped, each connected component of the resulting subgraph associates to a connected component of a rainbow forest in G. Therefore, the objective function minimizes the number of edges connected to the artificial vertex 0.

Connectivity requirements of a tree of $\hat{G}$ are enforced by Generalized Subtour Elimination Constraints (3.8) (GSECs) [21], alongside (3.5), which ensures that the number of edges in a tree is the number of vertices spanned by the tree minus one, and Box Constraints (3.11) -(3.12).

Logical Constraints (3.3) are a particular case of GSECS (3.8). They enforce that an edge incident to a vertex $i \in \hat{V}$ can only be selected if the corresponding vertex i is spanned by the tree. The rainbow forest of G we look for must be spanning, meaning that either each vertex $i \in V$ defines a connected component alone, or else vertex i is an endpoint of a forest edge. That condition is enforced by constraints (3.4). In case i defines a connected componnet alone, $x_{0,i+m} = 1$ must hold. Otherwise, there must be an edge $p \in \delta(i)$ in the forest. Accordingly, either $x_{0,p} = 1$ or else $x_{q,p} = 1$ for some edge $\{q,p\} \in \hat{E}_3$ of the extended line graph.

Note that Constraints (3.6) and (3.7) forbid two vertices v_{i+m} for $i \in V$ and v_e for $e \in \delta(i)$ to be spanned by the tree. That applies since, if $y_{i+m} = x_{0,i+m} = 1$, vertex i

alone defines a (trivial) rainbow tree. In that case, no edge incident to i must be in any other connected component of the rainbow forest, thus, $y_e = 0$ and $x_{0,e} = 0$ for all edges $e \in \delta(i)$.

Constraints (3.10), alongside GSECS (3.8), forbid any vertex $i \in V$ from appearing in two or more connected components at the same time, by forcing all edges in $\delta(i)$ to be connected. Note that it is not possible to fulfill such condition if i appears in two different components, as $\{0, e\} \notin \hat{E}_3, \forall e \in \delta(i)$. On the other hand, if i is only included in a single component, a solution where edges $e \in \delta(i)$ form a tree always exists. Single-vertex trees are accounted for as a special case, where $x_{0,i+m} = 1$ and $\sum_{e \in \delta(i)} y_e = \sum_{\{p,q\} \in \hat{E}_3 : p \in \delta(i), q \in \delta(i)} x_{p,q} = 0$.

Finally, inequalities (3.9) enforce the rainbow property. In a subset $S \subseteq \hat{V}_1$, the number of rainbow connected components must be at least $K(S)$, by the pigeonhole principle. Because a solution to the RSFP is a forest, this is equivalent to limiting the number of edges in S to be at most $E(S) - K(S)$. Note that, if $K(S) = 1$, then (3.9) is the Subtour Elimination Constraint associated with S.

Lastly, let $w(P_3)$ denote the linear programming relaxation (LPR) of z^*, that is:

$$w(P_3) = \min \left\{ \sum_{f \in \hat{E}_1 \cup \hat{E}_2} x_f : (x, y) \in P_3 \right\}. \tag{3.13}$$

3.4 Valid Inequalities

Our numerical tests have shown that the gap between z^* and $w(P_3)$, that is, $\frac{z^* - w(P_3)}{z^*}$ is often larger than 50%, resulting in high CPU times when that formulation is solved by a Branch-and-cut algorithm. In this section, we derive valid inequalities to strengthen these bounds.

3.4.1 Lifting Color Subtour Elimination Constraints

Inequalities (3.9) turn out to be rather weak when integrality constraints are dropped, as they are do not make use of variables $\mathbf{y}$. To enhance them, we introduce a lifting procedure based on bipartite matching.

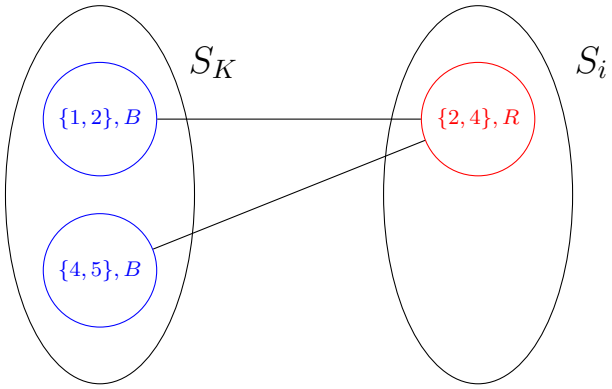

Figure 3.5: Two sets of vertices, S_K and S_i, form a bipartite graph in the Extended Rainbow Spanning Forest if all vertices in S_K have the same color and all vertices in S_i also have the same color.

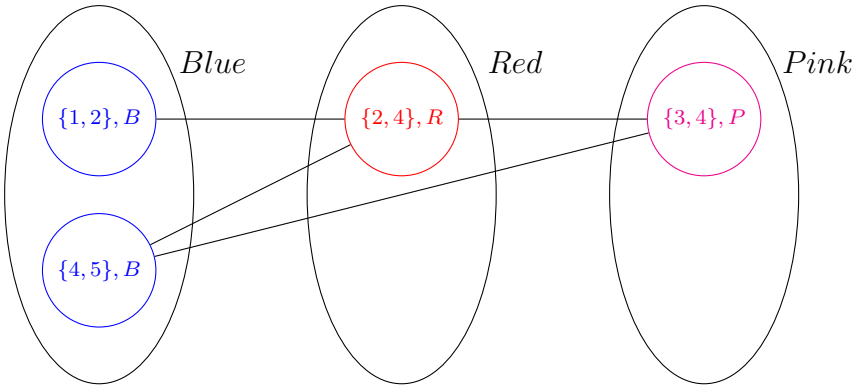

Figure 3.6: The union $S_i \cup S_K$ forms a k-partite graph for any subset of vertices $S_i \subseteq S$, where $k-1$ is the number of different colors in $S_K \cup S_i$. In this example, the subset S_i has two different colors, red and pink, therefore the union $S_i \cup S_K$ forms a 3-partite graph.

Consider a subset of vertices $S \subseteq \hat{V}_1$, and recall that $K(S)$ denotes the frequency of the most prevalent color in S, that is, $K(S) = \max_{k \in L} |\{v \in S : l(v) = k\}|$. Let $S_K \subseteq S$ be a set of vertices with the same color such that $|S_K| = K(S)$. For simplicity, we denote by $l(S_K)$ the color of all vertices in S_K. Let $S_i \subseteq S \setminus S_K$ be another set of similarly colored vertices, such that $l(S_i) \neq l(S_K)$.

Because S_K and S_i are sets of vertices with the same color, $\hat{E}(S_K) = \hat{E}(S_i) = \{\}$ and therefore $B(S_K \cup S_i, \hat{\delta}_{(S_K,S_i)}, \{l(S_K), l(S_i)\})$ defines a bipartite graph, where $\hat{\delta}_{(S_K,S_i)} = \{\{e,f\} : e \in S_K, f \in S_i\}$. Figure 3.5 illustrates such a graph. In a rainbow forest, every vertex in S_i can be connected to at most one vertex in S_K. Therefore, we can write:

$$\sum_{\{e,f\} \in \hat{E}(S_K \cup S_i)} x_{e,f} = \sum_{\{e,f\} \in \hat{\delta}(S_i, S_K)} x_{e,f} \leq \sum_{i \in S_i} y_i. \tag{3.14}$$

$$\sum_{\{e,f\} \in \hat{E}(S_K \cup S_i)} x_{e,f} \leq \sum_{i \in S_i} y_i. \tag{3.15}$$

Now, assume that S_i contains vertices with $k-1$ different colors. In that case,

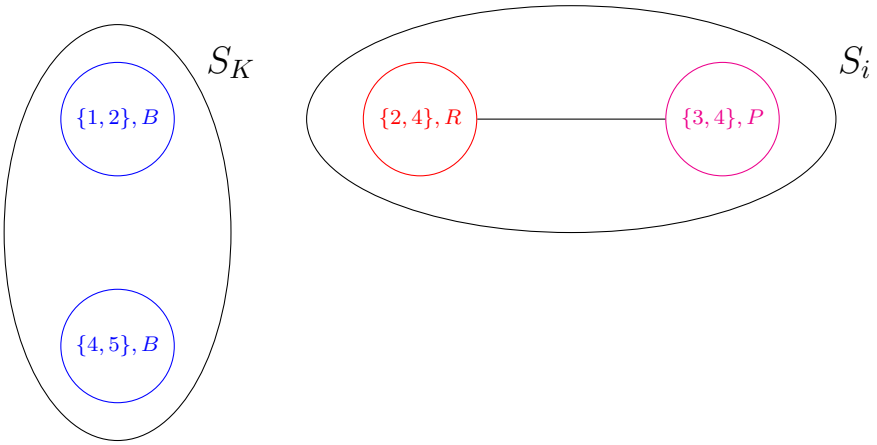

Figure 3.7: In the example below, there is exactly one connected component in S_i not connected to S_K. Accordingly, $\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i)} x_{e,f} = \sum_{i\in S_i} y_i - 1$ and $\alpha = 1$.

$S_i \cup S_K$ is a k-partite graph, that is, a graph that can be partitioned into k independent sets. Figure 3.6 illustrates a 3-partite graph in the example graph G_0.

Theorem 1. *Inequalities* (3.16) *are valid for any* $S_i \subseteq S \setminus S_K$*, where* S_i *is a set of vertices with* h *different colors.*

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i)} x_{e,f} \leq \sum_{i\in S_i} y_i. \tag{3.16}$$

Proof. Let $L = \{L_1, L_2, ..., L_h\}$ be the set of colors associated with the RSFP instance. Without loss of generality, we can assume that the color of the vertices in S_K is h. Our proof is based on induction in the color index j. The base case occurs when all the vertices in subset S_i have the same color, for which we have already demonstrated its correctness.

Let $V_i = \{v \in \hat{V} : l(v) = L_i\}$ denote the set of vertices in the line graph that have the color L_i. The induction hypothesis (IH) is that the inequality is valid for any $S_i \subseteq \bigcup_{i=1}^{j-1} V_i$. In other words, IH states that the inequality below is valid for RFP, if S_i contains only vertices with colors in the set $\{L_1, ..., L_{j-1}\}$.

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i)} x_{e,f} \leq \sum_{i\in S_i} y_i.$$

Let us introduce a slack variable $\alpha \geq 0$ to the inequality above, so that it reads as

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i)} x_{e,f} + \alpha = \sum_{i\in S_i} y_i. \tag{3.17}$$

Under the context of the k-bipartite graph representation, α denotes the number of connected components in S_i that are not connected to S_K. Figure 3.7 illustrates the idea.

We now show that the inequality must be valid for the union of $S_i = \bigcup_{i=1}^{j-1} V_i$ and any subset of V_j, i.e., we prove the induction step.

To that aim, pick any $S_j \subseteq V_j$ and note that the induction step consists in proving that $\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i\cup S_j)} x_{e,f} \leq \sum_{i\in(S_i\cup S_j)} y_i$ holds.

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i\cup S_j)} x_{e,f} = \sum_{\{e,f\}\in\hat{E}(S_K\cup S_i)} x_{e,f} + \sum_{\{e,f\}\in\hat{E}(S_j)} x_{e,f} + \sum_{\{e,f\}\in\hat{\delta}(S_j,(S_K\cup S_i))} x_{e,f}.$$

Since all vertices in S_j have the same color, $\sum_{\{e,f\}\in\hat{E}(S_j)} x_{e,f} = 0$, therefore:

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i\cup S_j)} x_{e,f} = \sum_{i\in S_i} y_i - \alpha + \sum_{\{e,f\}\in\hat{\delta}(S_j,(S_K\cup S_i))} x_{e,f}. \tag{3.18}$$

Let us express $\sum_{\{e,f\}\in\hat{\delta}(S_j,(S_K\cup S_i))} x_{e,f}$ in a more convenient way. We have established that α denotes the number of connected components in S_i not connected to S_K. Let S_{i1} denote the set of vertices that belong to these connected components, and S_{i2} denote the remaining vertices in S_i. Then, we can write:

$$\sum_{\{e,f\}\in\hat{\delta}(S_j,(S_K\cup S_i))} x_{e,f} = \sum_{\{e,f\}\in\hat{\delta}(S_j,S_K)} x_{e,f} + \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i1})} x_{e,f} + \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i2})} x_{e,f}. \tag{3.19}$$

Because all vertices in S_j have the same color, each component in S_{i1} can only have one edge incident to S_j, that is, $\sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i1})} x_{e,f} \leq \alpha$. On the other hand, note that S_{i2} denotes the set of vertices of S_i that belong to a connected component with one vertex in S_K. Let $u \in S_{i2}$ and $w \in S_K$ be vertices that belong to the same connected component. If an edge between u and a vertex $v \in S_j$ exists, then no edge between w and any vertex in S_j may exist in a rainbow forest. For if an edge connecting w to u exists, a cycle will be formed, and if an edge connecting w to $t \in S_j \setminus \{u\}$ exists, the rainbow property no longer holds. Conversely, if an edge between v and w exists, no edge connecting u to any vertex in S_j will exist in a rainbow forest. In other words, we can write $\sum_{\{e,f\}\in\hat{\delta}(S_j,S_K)} x_{e,f} \leq \sum_{j\in S_j} y_j - \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i2})} x_{e,f}$.

Substituting the inequalities above in (3.19), we find:

$$\begin{aligned}\sum_{\{e,f\}\in\hat{\delta}(S_j,(S_K\cup S_i))} x_{e,f} &= \sum_{\{e,f\}\in\hat{\delta}(S_j,S_K)} x_{e,f} + \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i1})} x_{e,f} + \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i2})} x_{e,f} \\ &\leq \sum_{j\in S_j} y_j - \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i2})} x_{e,f} + \alpha + \sum_{\{e,f\}\in\hat{\delta}(S_j,S_{i2})} x_{e,f} \\ &= \sum_{j\in S_j} y_j + \alpha\end{aligned}$$

We can substitute the expression above in (3.18) to find:

$$\sum_{\{e,f\}\in\hat{E}(S_K\cup S_i\cup S_j)} x_{e,f} \leq \sum_{i\in S_i} y_i - \alpha + \sum_{j\in S_j} y_j + \alpha = \sum_{i\in(S_i\cup S_j)} y_i.$$

concluding the proof.

□

By defining $S_i = S \setminus S_K$, we derive the following lifting of inequalities (3.9):

$$\sum_{\{e,f\}\in\hat{E}(S)} x_{e,f} \leq \sum_{i\in(S\setminus S_K)} y_i. \tag{3.20}$$

Note that we can derive additional valid inequalities (3.21) by replacing S_K with any set of equally colored vertices $S_l : |S_l| \leq K(S)$. The proof is analogous to the proof of correctness for inequalities (3.20).

$$\sum_{\{e,f\}\in\hat{E}(S)} x_{e,f} \leq \sum_{i\in(S\setminus S_l)} y_i. \tag{3.21}$$

3.4.2 Symmetry-breaking inequalities

Recall that inequalities (3.10) define a subtree structure for every vertex $i \in \hat{V}$. Such a structure suffers from symmetry issues, as several subtrees can be used to represent a single set of edges. Figure 3.8 illustrates this issue, by depicting three different ways of connecting the same set of vertices of $\hat{G}$.

In order to reduce symmetry, we define a set of inequalities that force a specific topology upon every subtree associated with a vertex $i \in V$. We have experimented with two topologies: path subtrees and star subtrees. Path subtrees can be enforced by inequalities (3.22), which dictate that every edge $e \in \delta(i)$ connects, at most, two other edges in $\delta(i)$.

$$\sum_{f\in\delta(i)} x_{e,f} \leq 2y_e, \forall i \in V, e \in \delta(i). \tag{3.22}$$

In a star topology, we define a root edge and several leaf edges. Every leaf edge $e \in \delta(i)$ is only allowed to be connected to one other edge $r \in \delta(i)$, that is, the root. To represent this structure, we define a new set of binary variables $\beta = \{\beta_e^i \in \mathbb{B} : i \in V, e \in \delta(i)\}$. The value of β_e^i indicates whether or not edge e is the root of the subtree associated with vertex i. With this new set of variables, we can force a star topology upon every subtree by introducing constraints (3.23)-(3.25) to our model.

$$\sum_{f\in\delta(i)} x_{e,f} \leq (|\delta(i)| - 1)\beta_e^i, \forall i \in V, e \in \delta(i). \tag{3.23}$$

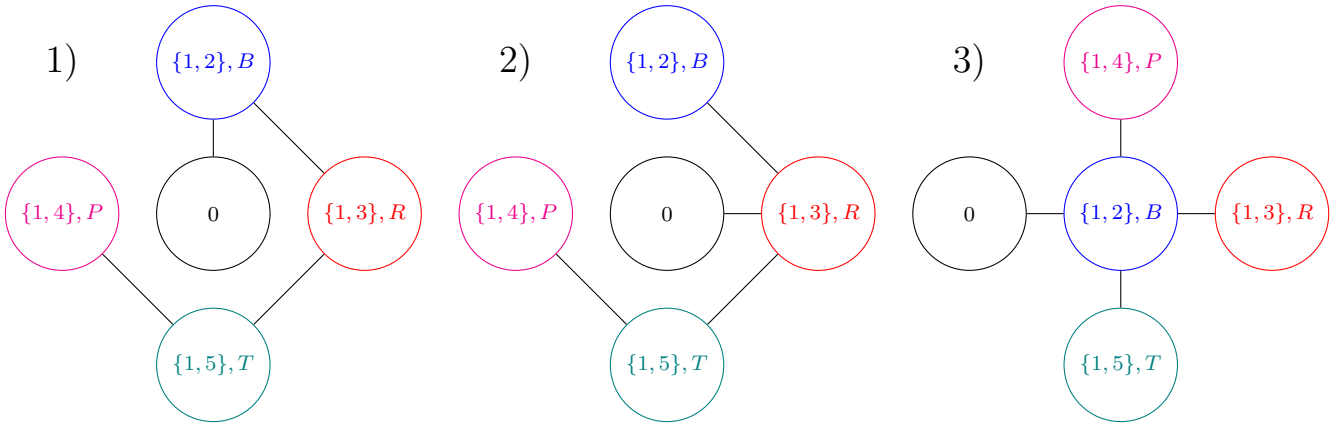

Figure 3.8: A set of edges can be represented in several ways within the line graph.

$$\sum_{e \in \delta(i)} \beta_e^i \leq 1 - y_{i+m}, \forall i \in V. \tag{3.24}$$

$$\beta_e^i \in [0, 1], \forall i \in V, e \in \delta(i). \tag{3.25}$$

In our experiments, star topologies yield much better results than path topologies. These results are expected, as the number of symmetrical path trees is far greater than the number of symmetrical stars for any given set of edges. The downside of such an approach is that the number of variables in our model grows by $2m$.

It is worth noting that these inequalities do not affect the LPR bounds. Their purpose is to simply reduce the runtime of our branch-and-cut algorithm.

3.4.3 Other valid inequalities

Additional valid RSFP inequalities can be derived by further exploring the properties of connected components in an extended rainbow spanning forest. For example, we note that no component can have more than $|L|$ edges and $|L|+1$ vertices. Therefore, we derive the following constraints:

$$\sum_{f \in \hat{E}_2} (|L| + 1) x_f + \sum_{f \in \hat{E}_1} x_f \geq |V|, \tag{3.26}$$

$$\sum_{\{e,f\} \in \hat{E}_3 : e \in \delta(i) \text{ or } f \in \delta(i)} x_{e,f} \leq |L| - 1, \forall i \in V. \tag{3.27}$$

We derive another set of inequalities by observing that, in a valid solution, all edges of $\delta(i)$ must have different colors, for any vertex $i \in V$. This observation follows from the fact that all edges connected to a vertex must belong in the same connected component.

$$\sum_{e \in \delta(i): l(e)=k} y_e \leq 1, \forall i \in V, k \in L. \tag{3.28}$$

Finally, we also include a particular case of inequalities (3.20), defined for the case where $K(S) = |S| - 1$.

$$\sum_{f:\{f \in \delta(u) \cup \delta(v) \setminus \{e\},\ \ l(f)=k\}} x_{e,f} \leq y_e, \quad \forall e = \{u, v\} \in E, k \in L \setminus l(e). \tag{3.29}$$

We thus define a stronger reformulation as follows:

$$z^* = \min \left\{ \sum_{f \in \hat{E}_1 \cup \hat{E}_2} x_f : (x, y, \beta) \in P_4 \cap (\mathbb{B}^{|\hat{V}|}, \mathbb{B}^{|\hat{E}|}, \mathbb{B}^{|V||E|}) \right\}, \tag{3.30}$$

where P_4 is the polytope defined by constraints (3.2)-(3.8), (3.10)-(3.12), (3.20), (3.23)-(3.29).

Chapter 4

A Branch-and-Cut Algorithm for the RSFP

4.1 An outline of the branch-and-cut algorithm

Our branch-and-cut algorithm starts solving the following relaxation for the RSFP.

$$w(P_5) = \min \left\{ \sum_{f \in \hat{E}_1 \cup \hat{E}_2} x_f : (x, y, \beta) \in P_5 \right\}, \tag{4.1}$$

where $P_5 \supset P_4$ is the polytope defined by constraints (3.2)-(3.7), (3.10)-(3.12), (3.23), (3.24) and (3.26)-(3.29). Note that the initial relaxation P_5 does not contemplate the exponentially many constraints (3.8) and (3.20), as those will be added dynamically to the relaxation, as cutting planes. From now on, let $\langle \overline{x}, \overline{y}, \overline{\beta} \rangle$ denote an optimal solution to (4.1).

To implement the branch-and-cut algorithm, we use the solver CPLEX 22.11. We disable all optimization parameters, such as parallel processing, cuts and heuristics native to the solver. The branching and node selection policies employed are the solver's default.

At every node of the branch-and-cut tree, we first generate a RSFP upper bound using the algorithm described in Section 4.5, hereby denoted by $\overline{z}$. We compare this upper bound to the best known global lower bound at the moment, $\underline{z}$. We terminate the algorithm if $\overline{z} - \underline{z} < 1$, as $\overline{z}$ is guaranteed to be optimal. This pruning is possible because every solution of (3.30) has integral weight, that is, $z^* \in \mathbb{Z}^+$. We say that $\underline{z}$ proves the optimality of z^* by bounds.

If $\overline{z} - \underline{z} > 1$, we attempt to separate CSEC inequalities (3.20) using the heuristic algorithm described in Section 4.4. If no violated inequalities are found by the heuristic algorithm, we attempt to separate GSECS (3.8) using the exact algorithm proposed by Lucena and Resende [21]. The same separation scheme is used to heuristically find violated CSECs (3.20) using the procedure described in Section 4.2.1.

If no GSECs or color inequalities are found by Lucena and Resende's algorithm, we employ the exact procedure described in Section 4.3 to separate CSECs. Finally, if no violated inequalities are found, the algorithm branches, following CPLEX's default branching policy.

In order to identify and prevent tailing off effects, the algorithm also branches if 20 iterations of the aforementioned sequence of steps are completed without any improvement to the LPR value (4.1).

4.2 Separation of GSECs

To separate GSECs (3.8), we use the $O(|\hat{V}|^4)$ separation algorithm proposed by Lucena and Resende [21]. The basic outline of the algorithm is to iterate over all possible $j \in \hat{V} : \overline{y}_j > 0$ and solve a conveniently defined maximum flow problem for each j.

In evaluating all cantidate $j \in \hat{V}$, one may find multiple violated inequalities. To cope with possibly too many violated inequalities, two intuitive solutions are either, adding all of them to the model, or only adding the most violated one.

A third possibility is suggested in [21]. Let $S \subseteq \hat{V}$ be the set of vertices that maximizes $\sum_{f \in \hat{E}(S)} \overline{x}_f - \sum_{i \in S \setminus \{j\}} \overline{y}_i$, that is, the set associated with the most violated inequality. Add the corresponding inequality to the model and remove vertices $v \in S$ and corresponding edges $e \in \hat{E}(S)$ from the network that defines the maximum flow problem. Find the most violated inequality in the new network. Repeat the procedure until the current network yields no violated inequality. The result is a set of inequalities that do not share common variables. These inequalities are said to be *orthogonal*, and their inclusion generally decreases overall CPU times by reducing the number of required cutting planes iterations [21]. For that reason, we employ this approach in our branch-and-cut algorithm.

4.2.1 Obtaining CSECs from the GSEC separation

Our experiments have found that sets S separated by the exact GSEC separation procedure often violate inequalities (3.20) as well. We therefore also check, for every $j \in \hat{V}$, whether or not the set S associated with the most violated inequality $\sum_{f \in \hat{E}(S)} x_f \leq \sum_{i \in S \setminus \{j\}} y_i$ also violates inequalities (3.20). This additional test is inexpensive and often reduces the number of color constraints that need to be separated by

other methods. If multiple violated inequalities (3.20) are found, we keep the one that maximizes $\sum_{\{e,f\}\in E(S)} \overline{x}_{e,f} - \sum_{i\in(S-S_K)} \overline{y}_i$ and add it to the relaxation.

It is worth highlighting that color inequalities found by the procedure above are added independently of their associated GSECs (3.8). That is, in a single iteration, we may add one inequality (3.8) and another inequality (3.20) at the same time, and those inequalities do not need to be associated with the same set S.

4.3 An exact separation algorithm for CSECs

Recall that the separation problem is defined over the line graph. Unlike GSECs, we know no polynomial separation algorithm for color constraints (3.9) or their strengthened version (3.20). Instead, we present an Mixed Integer Programming formulation to separate them.

We define binary variables $\gamma = \{\gamma_i \in \mathbb{B} : i \in \hat{V}_1\}$, $\omega = \{\omega_e \in \mathbb{B} : e \in \hat{E}_3\}$, $\sigma = \{\sigma_k \in \mathbb{R} : k \in L\}$ and $\kappa \in \mathbb{R}$ for the model. Variables γ_i indicate whether or not a vertex i belongs to the optimal subset S. Likewise, variables ω_e indicate whether or not an edge e belongs to the optimal subset. A value of $\sigma_k = 1$ indicates that color $k \in L$ is the most frequent color in the optimal set, with $\kappa = K(S)$ vertices. Then, inequalities (3.20) can be separated by solving the following Quadratic Mixed Integer Program.

$$z_{csec} = \max \left\{ \sum_{e\in\hat{E}_3} \overline{x}_e w_e - \sum_{i\in\hat{V}_1} \overline{y}_i v_i (1 - \sigma_{l(i)}) : (\gamma, \omega, \sigma, \kappa) \in P_{csec} \cap (\mathbb{B}^{|\hat{V}_1|}, \mathbb{B}^{|\hat{E}_3|}, \mathbb{B}^{|L|}, \mathbb{R}) \right\} \tag{4.2}$$

where P_{csec} is the polytope defined by constraints (4.3)-(4.11).

$$\kappa \geq \sum_{i \in \hat{V}_1 : l(i) = k} \gamma_i \qquad k \in L \tag{4.3}$$

$$\kappa \leq (1 - \sigma_k)|\hat{V}_1| + \sum_{i \in \hat{V}_1 : l(i) = k} \gamma_i \qquad k \in L \tag{4.4}$$

$$\sum_{k \in L} \sigma_k = 1 \tag{4.5}$$

$$\omega_{e,f} \leq v_e \qquad \{e, f\} \in \hat{E}_3 \tag{4.6}$$

$$\omega_{e,f} \leq v_f \qquad \{e, f\} \in \hat{E}_3 \tag{4.7}$$

$$\sigma_k \in [0, 1] \qquad k \in L \tag{4.8}$$

$$\gamma_i \in [0, 1] \qquad i \in \hat{V}_1 \tag{4.9}$$

$$\omega_e \in [0, 1] \qquad e \in \hat{E}_3 \tag{4.10}$$

$$\kappa \geq 0 \tag{4.11}$$

Inequalities (4.3)-(4.4) ensure that $\kappa = \max_{k \in L}(\sum_{i \in \hat{V}_1 : l(i) = k} \omega_i) = K(S)$ and inequalities (4.5)-(4.6) ensure that an edge $\omega_{\{e,f\}}$ can only be part of a solution if its adjacent vertices γ_e and γ_f are also part of the solution. If $z_{csec} > 0$, we have found an violated inequality that can be added to the model.

4.4 A heuristic separation algorithm for CSECs

Solving the quadratic problem (4.2) at every iteration is very expensive. Thus, we attempt to separate these inequalities heuristically at first, using the greedy algorithm below. We begin with an empty set of vertices S and define a weight for each vertex v as $\sum_{\{v,e\} \in \hat{E}(S \cup \{v\})} x_{v,e} - y_v$, that is, the amount vertex v would contribute to violate the inequality $\sum_{\{e,f\} \in \hat{E}(S)} x_{e,f} - \sum_{i \in (S-K)} y_i \leq 0$, were it added to S. At every iteration, we choose between the k heaviest vertices, where k is empirically defined. After we find a violated inequality, we continue adding vertices. However, we only introduce vertices that strictly increase the violation, stopping as soon as none is found. We randomize the vertice selection in order to allow a richer variety of solutions, while still prioritizing the most violating sets.

Our experiments show that the greedy algorithm finds violated CSECs very often, provided that they exist, heavily reducing the number of calls for the exact algorithm.

Algorithm 1 Greedy CSEC Separation Algorithm

Input: Incumbent solution $\langle \overline{x}, \overline{y}, \overline{\beta} \rangle$
Output: Locally most violated set S
1: $S \leftarrow \emptyset$
2: **while** $(\sum_{\{e,f\} \in E(S)} x_{e,f} - \sum_{i \in (S \setminus K)} y_i) \leq 0$ **do**
3: $v \leftarrow$ One of the k heaviest vertices in $\hat{V} \setminus S$
4: $S \leftarrow S \cup \{v\}$
5: **end while**
6: **while** max weight in $\hat{V} \setminus S > 0$ **do**
7: $v \leftarrow$ One of the k heaviest vertices in $\hat{V} \setminus S$ with weight greater than 0
8: $S \leftarrow S \cup \{v\}$
9: **end while**

4.5 Upper bounds for the RSFP

We now describe a heuristic procedure to generate upper bounds for the RSFP, based on the GRASP algorithm proposed in [14] and adapted to benefit from information coming from the current LPR.

The GRASP algorithm [12] consists of two steps: A constructive step and a local search step. We describe both steps in detail below.

4.5.1 Constructive step

For the constructive step, consider the original graph $G(V, E, L)$ and construct an auxiliary edge-colored weighted graph $\widetilde{G}(V, \widetilde{E}, L)$ such that every edge $e \in E$ corresponds to an edge $\widetilde{e} \in \widetilde{E}$, with color $l(\widetilde{e}) = l(e)$ and weight $w(\widetilde{e}) = \overline{y}_e$.

We now describe a modified Kruskal [18] algorithm that generates a locally minimum rainbow spanning forest from graph $\widetilde{G}$. The function $comp(i) : i \in V$ denotes the tree T_i that i currently belongs to. The function $join(T_i, T_j, e)$ receives two trees $T_i(V_i, E_i, L_i)$, $T_j(V_j, E_j, L_j)$ and an edge e, and returns a tree $T_{ij}(V_{ij}, E_{ij}, L_{ij})$ with vertices $V_{ij} = V_i \cup V_j$, edges $E_{ij} = E_i \cup E_j \cup \{e\}$ and colors $L_{ij} = L_i \cup L_j \cup l(e)$.

We initialize the algorithm with n connected components, one for every vertex $v \in V$. At every iteration, we choose among the k heaviest edges, where k is an empirically defined parameter, and attempt to add the chosen edge to the forest. Note that, when $k = 1$, the algorithm is the original Kruskal algorithm.

The idea behind this algorithm is to prioritize edges that have been used in the

Algorithm 2 Modified Kruskal Algorithm

Input: Modified input graph $\widetilde{G}$
Output: A Rainbow Spanning Forest F

```
F ← {T_i(i, ∅, ∅) : i ∈ V}
while Ẽ ≠ ∅ do
    {i, j} ← One of the k heaviest edges in Ẽ
    Ẽ ← Ẽ \ {{i, j}}
    T_i(V_i, E_i, L_i) ← comp(i), T_j(V_j, E_j, L_j) ← comp(j)
    if L_i ∩ L_j = ∅ then
        T_ij ← join(T_i, T_j, {i, j})
        F ← (F \ {T_i, T_j}) ∪ T_ij
    end if
end while
```

current linear relaxation solution, as the strong linear programming bounds may suggest that some of these edges are likely to be present in good feasible solutions.

4.5.2 Local search

Solutions generated by the constructive step can usually be improved by exploring their neighborhoods. For this step, we use a slightly modified version of the Variable Neighborhood Descent (VND) algorithm proposed in [14]. A neighborhood $N(F) \mapsto F'$ is a function that transforms a solution F into a neighbor solution F'. In the context of the RSFP, we say that two forests F and F' are neighbors if three or less subtrees of F are not present in F'.

The idea is to explore increasingly complex neighborhoods in search of an improving solution, until no improvement is found. Let $F = \{T_1, \ldots, T_z\}$ be our current solution, where each tree $T_i(V_i, E_i, L_i)$ contains vertices V_i, edges E_i and colors L_i. Frota et al. [14] define three neighborhoods for the RSFP, described in order of their size as follows.

- **Leaf_Merge**: Encompasses all solutions that can be reached by merging two trees T_i and T_j using a leaf vertex from a third tree, T_k.

  More specifically, we look for vertices $i \in V_i, j \in V_j, k \in V_k$ such that $\{i,k\}, \{k,j\} \in E; l(\{i,k\}) \neq l(\{k,j\}); l(\{i,k\}), l\{k,j\}) \notin L_i \cup L_j$, and create tree $T_{i \cup j}$ with vertices $V_i \cup V_j \cup \{k\}$ and edges $E_i \cup E_j \cup \{\{i,k\},\{k,j\}\}$. Figures 4.1 and 4.2 show, respectively, an example forest F and one of its neighbors in $Leaf\_Merge(F)$.

- **Edge_Merge**: Encompasses all solutions that can be reached by merging two trees T_i and T_j by directly connecting them with one edge and repairing inviabilities by

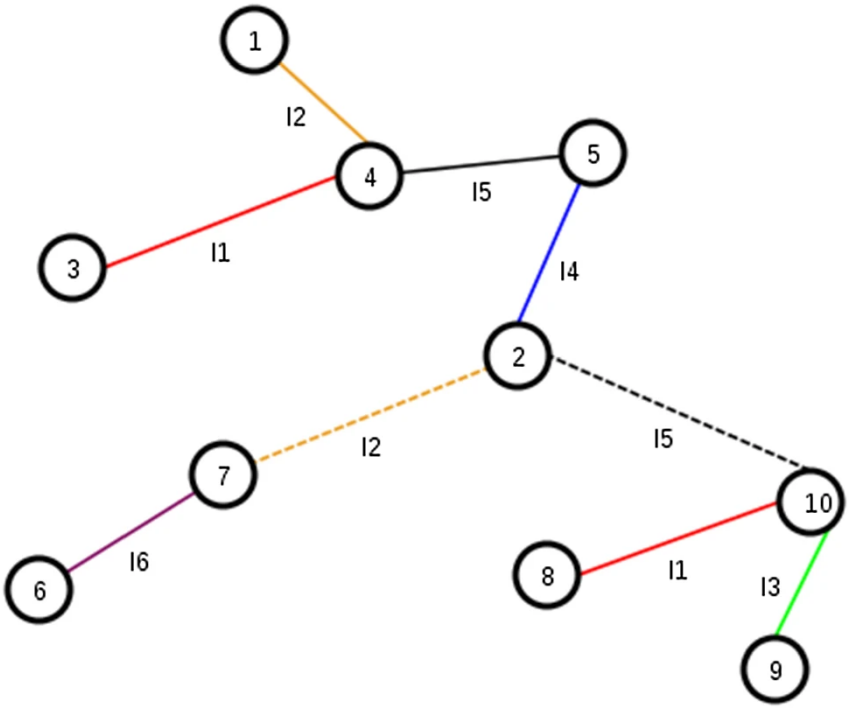

Figure 4.1: A forest F with three components. Dashed edges are edges present in the original graph G, but not in F.

Source: Frota et al., [14]

removing one edge from T_j and adding another in its place.

More specifically, we look for vertices $i \in V_i; j, u, v \in V_j; \{u, v\} \notin E_j$ and an edge $e \in E_j$ such that $l(\{u, v\}) \neq l(\{i, j\})$ and $l(\{u, v\}), l(\{i, j\}) \notin (L_i \cup L_j \setminus l(e))$. We then create tree $T_{i \cup j}$ with vertices $V_i \cup V_j$ and edges $E_i \cup (E_j \setminus \{e\}) \cup \{\{i, j\}, \{u, v\}\}$. Figures 4.3 and 4.4 show, respectively, an example forest F and one of its neighbors in $Edge\_Merge(F)$.

- **2_Edge_Merge**: Encompasses all solutions that can be reached by merging two trees T_i and T_j by directly connecting them with two edges and repairing inviabilities by removing one edge from T_j.

  More specifically, we look for vertices $i, u \in V_i; j, v \in V_j$ and an edge $e \in E_j$ such that $l(\{u, v\}) \neq l(\{i, j\})$ and $l(\{u, v\}), l(\{i, j\}) \notin (L_i \cup L_j \setminus l(e))$. We then create tree $T_{i \cup j}$ with vertices $V_i \cup V_j$ and edges $E_i \cup (E_j \setminus \{e\}) \cup \{\{i, j\}, \{u, v\}\}$. Figures 4.5 and 4.6 show, respectively, an example forest F and one of its neighbors in $2\_Edge\_Merge(F)$.

In our implementation, we modify neighborhood $Edge\_Merge$, to allow any edges $v \in \{T_1, \ldots, T_z\} - \{T_i\}$. That is, once we remove e, splitting T_j into two components T_{j1} and T_{j2}, we do not need to rejoin T_{j1} and T_{j2} back together with $\{u, v\}$. We can instead attempt to join T_{j1} with any component in the graph, including but not limited to T_{j2}.

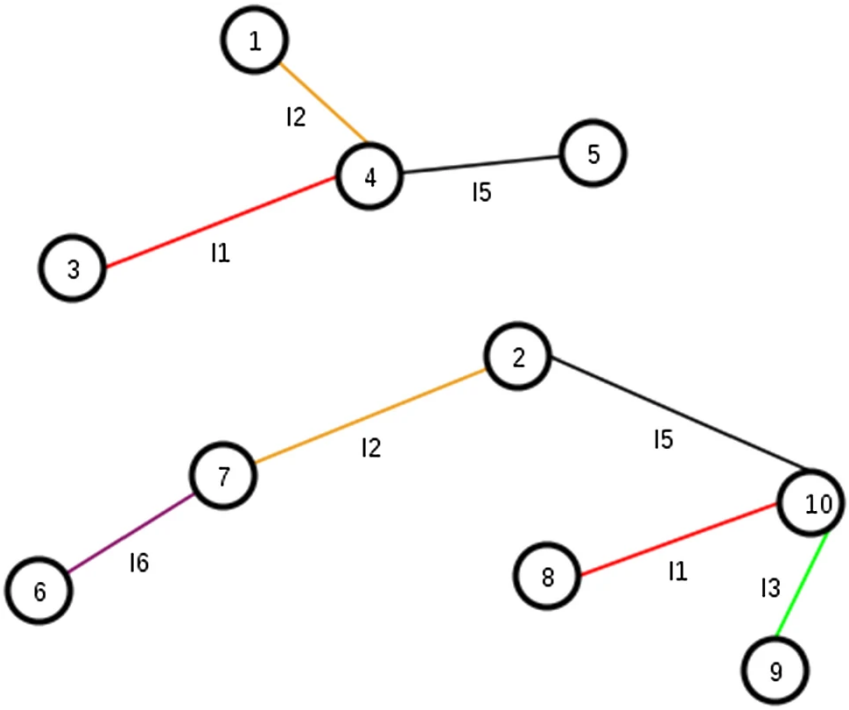

Figure 4.2: Neighbor in $Leaf\_Merge(F)$ with two components (Let $i = 7, j = 10, k = 2$).

Source: Frota et al., [14]

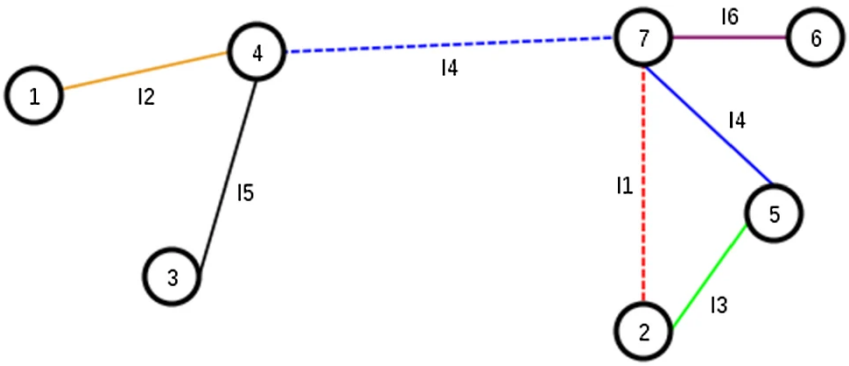

Figure 4.3: A forest F with two components. Dashed edges are edges present in the original graph G, but not in F.

Source: Frota et al., [14]

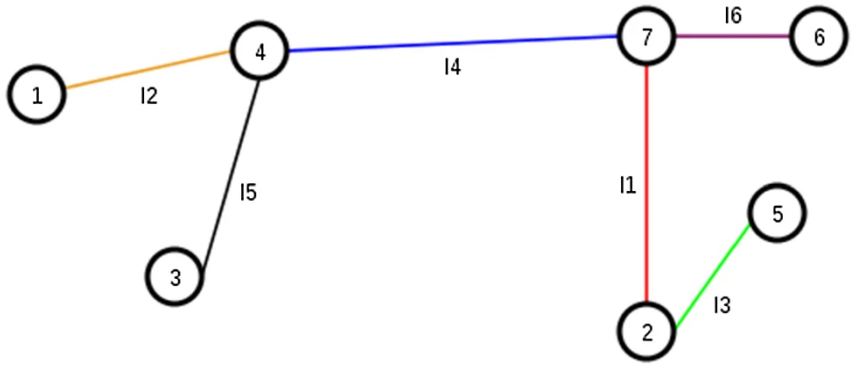

Figure 4.4: Neighbor in $Edge\_Merge(F)$ with one component (Let $i = 4, j = 7, u = 7, v = 2$, $e = \{5, 7\}$).

Source: Frota et al., [14]

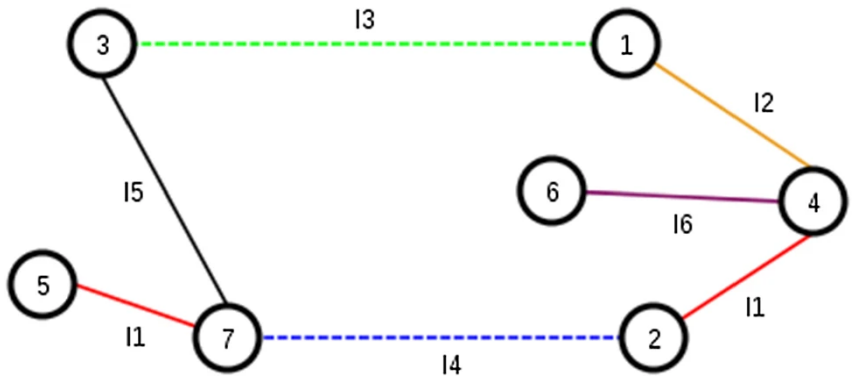

Figure 4.5: A forest F with two components. Dashed edges are edges present in the original graph G, but not in F.

Source: Frota et al., [14]

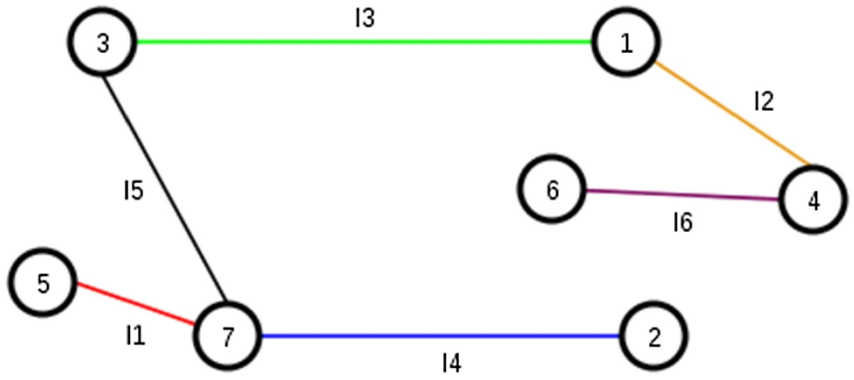

Figure 4.6: Neighbor in $2\_Edge\_Merge(F)$ with one component (Let $i = 3, j = 1, u = 7, v = 2$, $e = \{4, 2\}$).

Source: Frota et al., [14]

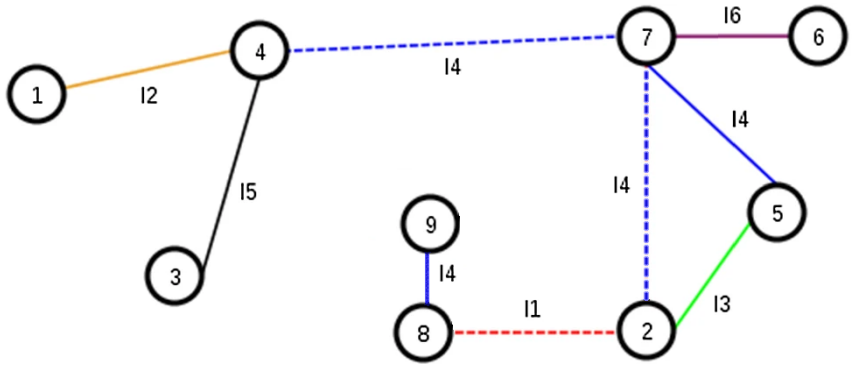

Figure 4.7: A forest F with three components. Dashed edges are edges present in the original graph G, but not in F.

Source: Adapted from Frota et al., [14]

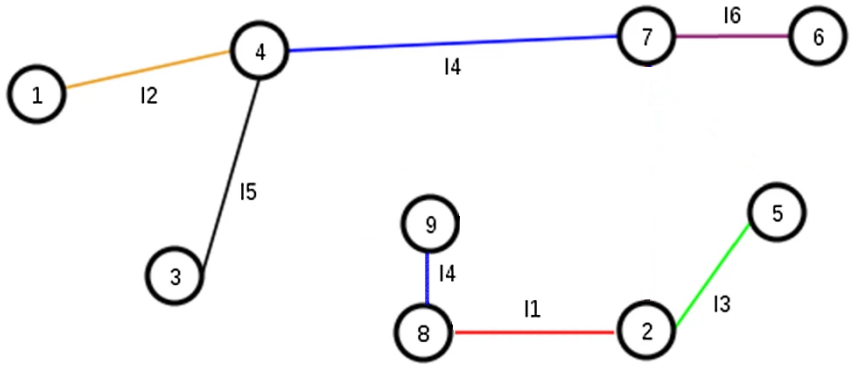

Figure 4.8: Neighbor in $Edge\_Merge\_Imp(F)$ with two components (Let $i = 4, j = 7, u = 2, v = 8$, $e = \{5, 7\}$).

Source: Adapted from Frota et al., [14]

Let $Edge\_Merge\_Imp$ denote this new neighborhood. Figure 4.5 shows an example forest F and Figure 4.6 shows one of its neighbors in $Edge\_Merge\_Imp(F)$.

4.6 Instances for Numerical Experiments

In the following sections, we present numerical experiments to evaluate the quality of our LPR bounds and of our branch-and-cut algorithm. Our numerical experiments make use of the RSFP instances introduced in [1]. This dataset consists of instances with 20, 30, 40, 50, 100, 200 and 400 vertices. The number of edges of an instance is given by $\lceil \frac{dn(n-1)+n}{2} \rceil, d \in \{0.1, 0.2, 0.3\}$ and the number of colors is set to $\lceil \frac{log(m)}{2} \rceil$, $\lceil log(m) \rceil$ and $\lceil 2log(m) \rceil$. Each edge is then randomly assigned a color, with equal probability. This combination of vertices, edges and colors defines a *scenario*. The dataset consists of five

different instances per scenario, totalling 315 instances.

4.7 Linear Programming Relaxation Bounds

In this section, we describe numerical experiments showing that our LPR bounds are strong compared to those found in the literature. More precisely, we compare our LPR bounds to those provided by the formulation in [14]. Let LP_L denote the Linear Programming relaxation associated with our formulation, P_4:

$$LP_L = \min \left\{ \sum_{f \in \hat{E}_1 \cup \hat{E}_2} x_f : (x, y, \beta) \in P_4 \right\}. \tag{4.12}$$

Likewise, let LP_C denote the Linear Programming relaxation of Frota et al.'s formulation, that is:

$$LP_C = \min \left\{ \sum_{c=1}^{\overline{c}} \alpha_c + \sum_{v \in V} \phi_v : (\alpha, \phi, x, y) \right\} \in P_2. \tag{4.13}$$

Finally, we denote by LB the trivial lower bound $LB = \frac{|V|}{|L|+1}$ described in [1]. We highlight that constraints (2.2) and (2.4) ensure that $LP_C \geq LB$, while (3.26) ensures that $LP_L \geq LB$.

In order to calculate LP_L, we employ the cutting plane algorithm described in section 4.1. To separate GSECs for LP_C, we use an adaptation of the algorithm described in Section 4.2, defining two arcs in the flow graph for every edge $e = \{u, v\} \in E$ of the original graph, with weight $w_{u,v} = w_{v,u} = \sum_{c=1}^{\overline{c}} \overline{x}_e^c$. We define a time limit of 10400 seconds and compare the best bounds obtained by both formulations within that time limit. Detailed results for every instance can be found in Appendix A.1. Detailed results of separation time costs can be found in Appendix A.2.

Both formulations timed out without finding any lower bound for all instances with 200 vertices or above. For all instances with 100 vertices, both formulations timed out with little to no gain over the trivial lower bound LB. Out of the 180 instances with 50 vertices or less, we were able to find the value of LP_L for 67 of them. We build performance profiles [11] for both formulations, shown in Figures 4.9 and 4.10. Let $g(LP)$ denote the gap between a LPR bound LP and z^*, that is, $\frac{z^* - LP}{z^*}$. Figure 4.9 then plots the number of instances for which $g(LP) \leq Gap$ holds.

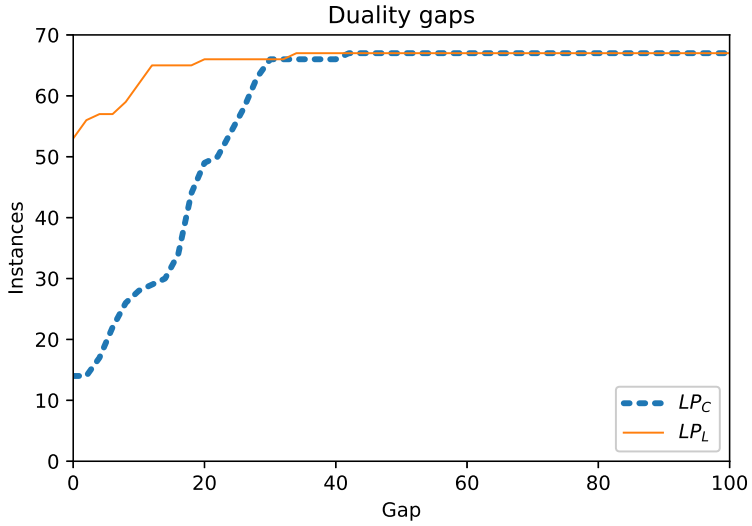

Figure 4.9: Performance profile for both formulations.

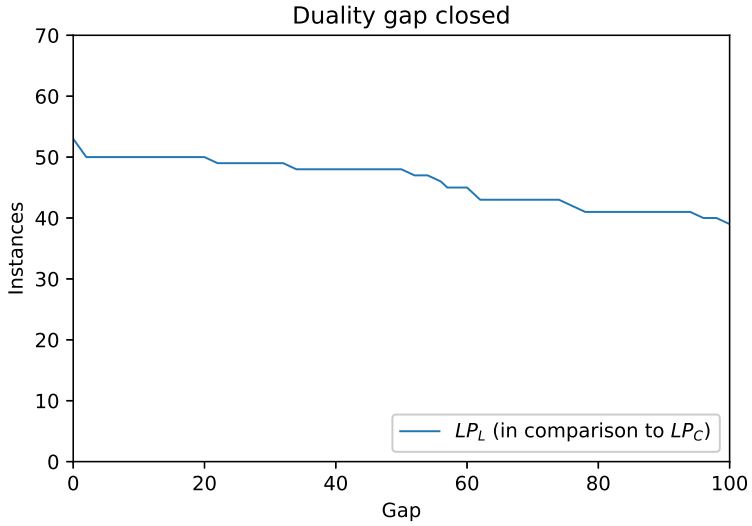

Figure 4.10: Performance profile of LP_L in comparison to LP_C.

Among the instances for which both relaxations could be evaluated within the time limit, we observe that $LP_L \geq LP_C$ always holds. We now analyze how much of the gap between LP_C and z^* is closed by LP_L. We denote this value by $h(LP_L) = \frac{LP_L - LP_C}{z^* - LP_C}$. Figure 4.10 then plots the number of instances for which $LP_L \geq Gap$ holds, ignoring instances for which $LP_C = LP_L = z^*$. We highlight that, for 39 instances, LP_L entirely closes the gap. These two figures give us an idea of how close to the optimal solution both LPR bounds are.

Among the remaining 113 instances with 50 vertices or less, we define LP_L and LP_C as the best lower bounds achieved for each formulation in the time limit of 10400 seconds. We observe that $LP_L > LP_C$ holds for 76 of them, while $LP_L = LP_C = LB$ holds for 23 instances and $LP_L = LP_C > LB$ holds for 7 instances. There are only 5

instances for which $LP_C > LP_L$ holds.

There are 117 instances with 50 vertices or less for which the trivial lower bound LB is enough to prove the optimality of z^*, by integrality. Among the remaining 63 instances, LP_L proves the optimality of z^* in 37 instances, with 24 of the remaining 26 instances timing out. For comparison, LP_C only proves the optimality of z^* in 23 instances, with no timeouts.

We now turn our attention to the CPU time needed to calculate LP_L, which is very high even for medium-sized instances. Indeed, we failed to calculate LP_L to optimality for most instances with 40 vertices or above. Table 4.1 shows detailed information about the average computational cost of LP_L for every scenario. The first three columns identify the number of vertices, edges and colors of a scenario. The fourth column shows its average time cost over the five instances that define the scenario, in seconds. The fifth column shows the average percentage of the total time consumed by the GSEC separation algorithm. The sixth and seventh columns show the same information for the exact and heuristic CSEC separation algorithms, respectively. Finally, the eighth column shows the percentage of the total time spent on reoptimization, that is, on solving linear programming subproblems.

The table shows that the separation costs are generally dwarved by the reoptimization cost as the instances grow larger, implying that, while the separation algorithms are cheap, the formulation tends to require too many CSECs and GSECs, resulting in a large linear program and too many re-optimizations.

Finally, in Table 4.2, we show that the line graph is less dense than the original graph. The first four columns show the average number of vertices, edges, colors and the density of the original graph in every scenario, while the last three columns show the average number of vertices, edges and the density of the line graph.

4.8 Computational Experiments

In this Section, we compare the quality of the solutions generated by Frota et al.'s formulation (BC^C) and our branch-and-cut algorithm (BC^L). The computational experiments were carried on an Intel(R) Xeon(R) CPU E5405 processor running at 2.00 GHZ with 16 GB of RAM. Like in Section 4.7, we use the same set of instances proposed in [2] and run both algorithms only once.

In order to evaluate BC^C and BC^L under the same conditions, we disable all optimization parameters, such as parallel processing and heuristic cuts. In our branch-and-cut, we also disable the native heuristic upper bounds provided by CPLEX. The

Vertices	Edges	Colors	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
20	39	3	3.93	34.63%	5.47%	0.51%	59.39%
20	39	6	124.61	19.91%	12.30%	0.15%	67.63%
20	39	11	387.72	10.60%	23.64%	0.12%	85.63%
20	58	3	26.78	16.06%	0.97%	0.35%	82.62%
20	58	6	599.85	5.25%	7.84%	0.24%	86.68%
20	58	12	32.40	9.70%	16.48%	0.38%	73.44%
20	77	4	55.13	3.98%	3.87%	0.39%	91.76%
20	77	7	17.32	6.19%	6.66%	0.68%	86.47%
20	77	13	15.39	8.13%	13.16%	0.73%	77.98%
30	74	4	1155.98	23.96%	1.26%	0.28%	74.50%
30	74	7	10400.07	3.22%	0.24%	0.03%	96.50%
30	74	13	1401.70	3.48%	2.75%	0.13%	93.64%
30	117	4	5760.48	4.00%	0.28%	0.13%	95.59%
30	117	7	9752.83	0.57%	0.30%	0.05%	99.08%
30	117	14	10400.17	0.90%	0.62%	0.04%	98.44%
30	161	4	4440.11	0.86%	0.30%	0.14%	98.69%
30	161	8	7463.99	0.98%	0.33%	0.16%	98.53%
30	161	15	8651.82	1.08%	0.78%	0.11%	98.03%
40	118	4	8733.21	7.78%	0.01%	0.12%	92.08%
40	118	7	10400.17	2.55%	0.03%	0.05%	97.37%
40	118	14	10400.13	0.86%	0.26%	0.04%	98.83%
40	196	4	10400.12	0.65%	0.04%	0.09%	99.22%
40	196	8	10400.21	0.53%	0.04%	0.12%	99.31%
40	196	16	10400.24	0.91%	0.14%	0.14%	98.81%
40	274	5	10400.12	0.44%	0.02%	0.22%	99.32%
40	274	9	10400.20	0.75%	0.03%	0.21%	99.01%
40	274	17	10400.73	1.63%	0.11%	0.26%	98.00%
50	173	4	10400.16	9.48%	0.01%	0.13%	90.38%
50	173	8	10400.40	15.19%	0.00%	0.08%	84.72%
50	173	15	10400.18	0.66%	0.01%	0.09%	99.23%
50	295	5	10400.19	0.38%	0.01%	0.22%	99.39%
50	295	9	10400.20	0.44%	0.00%	0.17%	99.39%
50	295	17	10400.39	1.22%	0.00%	0.20%	98.57%
50	418	5	10400.30	0.24%	0.00%	0.26%	99.49%
50	418	9	10400.41	0.40%	0.00%	0.23%	99.37%
50	418	18	10400.29	4.97%	0.01%	0.25%	94.77%

Table 4.1: Computational cost details for the line graph reformulation.

Vertices	Edges	Colors	Density	Line Graph Vertices	Line Graph Edges	Line Graph Density
20	39	3	0.1	60	174.80	0.05
20	39	6	0.1	60	215.40	0.06
20	39	11	0.1	60	234.00	0.06
20	58	3	0.2	79	346.60	0.06
20	58	6	0.2	79	421.80	0.07
20	58	13	0.2	79	475.00	0.08
20	77	4	0.3	98	615.40	0.06
20	77	7	0.3	98	704.00	0.07
20	77	13	0.3	98	762.20	0.08
30	74	4	0.1	105	530.00	0.05
30	74	7	0.1	105	626.40	0.06
30	74	13	0.1	105	678.40	0.06
30	117	4	0.2	147	1101.60	0.05
30	117	7	0.2	147	1290.60	0.06
30	117	14	0.2	147	1402.20	0.06
30	161	4	0.3	191	1754.60	0.05
30	161	8	0.3	191	2191.80	0.06
30	161	15	0.3	191	2357.20	0.06
40	118	4	0.1	158	1055.20	0.04
40	118	7	0.1	158	1290.00	0.05
40	118	14	0.1	158	1415.00	0.06
40	196	4	0.2	236	2412.00	0.04
40	196	8	0.2	236	2985.00	0.05
40	196	16	0.2	236	3200.00	0.06
40	274	5	0.3	314	4336.20	0.04
40	274	9	0.3	314	4915.40	0.05
40	274	17	0.3	314	5262.00	0.05
50	173	4	0.1	223	1970.60	0.04
50	173	8	0.1	223	2333.80	0.05
50	173	15	0.1	223	2561.20	0.05
50	295	5	0.2	345	4765.20	0.04
50	295	9	0.2	345	5550.80	0.05
50	295	17	0.2	345	5990.80	0.05
50	418	5	0.3	468	7977.40	0.04
50	418	9	0.3	468	9285.40	0.04
50	418	18	0.3	468	10140.20	0.05

Table 4.2: A brief comparison between the average dimensions of the original graph and the line graph in each scenario.

branching and node selection policies employed are the solver's default.

Table 4.3 presents the results for instances with up to 50 vertices. Because BC^C could not find feasible solutions within the time limit for some instances with 50 vertices, we do not include these groups of instances in the table. Every line presents average statistics over a scenario. The first three columns identify a scenario, reporting respectively the group's number of vertices, edges and colors. The fourth column, BC^C, indicates the average optimal solution found by Frota et al.'s formulation over each group of instances. The fifth column indicates the execution time, in seconds, required to find that solution. Likewise, the sixth and seventh columns indicate, respectively, the average optimal solution of our branch-and-cut algorithm and the execution time it requires.

A more detailed comparison of both formulations for all instances with up to 50 vertices can be found in Appendix A.3. For all instances with 100 vertices, Frota et al's formulation was unable to find any feasible solution within the time limit. This is most likely due to the disabling of CPLEX heuristic upper bounds for both models. Meanwhile, our formulation is able to find upper bounds, without proving their optimality. For instances larger than 200 vertices, neither formulation is able to find a feasible solution within the time limit.

The results show that the BC^L requires high running times even for some small instances and often performs worse than the BC^C. Table 4.4 shows detailed information of separation costs for the BC^L algorithm. The first three columns identify the number of vertices, edges and colors of an instance group. The fourth column shows its average time cost. The fifth column shows the average percentage of the total time consumed by the GSEC separation algorithm. The sixth and seventh columns show the same information for the exact and heuristic CSEC separation algorithms, respectively. Finally, the eighth column shows the percentage of the total time spent on reoptimization, that is, on solving LPR subproblems. We do not include the time spent on calculating upper bounds, as these always take up less than 0.01% of the total time. The results show that reoptimization dominates the time cost of the branch-and-cut algorithm as instances grow larger.

Another issue is that the upper bounds found by the GRASP heuristic do not always match the optimal solution, which nullifies the pruning strategy of stopping the algorithm when $\overline{z} - \underline{z} < 1$. Therefore, it may be worth looking into better upper bounding heuristics as well.

Finally, for large instances, the model grows too large to be solved at the root node, even when relaxing inequalities (3.8) and (3.20). For instances with 100 vertices and 1595 edges, for example, the algorithm is at most able to add 18 inequalities in total before timing out.

Vertices	Edges	Colors	BC^L	Time	BC^C	Time
20	39	3	6.60	0.06	6.60	0.63
20	39	6	4.20	3.20	4.20	0.85
20	39	11	2.20	55.68	2.20	0.62
20	58	3	5.60	1.88	5.60	0.88
20	58	6	3.20	3.92	3.20	0.85
20	58	12	2.00	0.40	2.00	1.80
20	77	4	4.00	3.64	4.00	0.18
20	77	7	3.00	1.23	3.00	0.80
20	77	13	2.00	1.57	2.00	1.88
30	74	4	8.40	84.20	8.40	45.40
30	74	7	5.00^{2}	5233.04	5.00	15.18
30	74	13	3.00	14.47	3.00	13.12
30	117	4	6.80^{1}	2241.10	6.80	122.20
30	117	7	4.20^{1}	3203.50	4.20	25.21
30	117	14	3.00^{5}	10400.04	2.60	11.33
30	161	4	6.40	1238.04	6.40	13.80
30	161	8	4.00	76.57	4.00	92.46
30	161	15	2.20^{1}	3699.03	2.00	85.11
40	118	4	10.20^{1}	3758.22	10.20	1659.85
40	118	7	6.40^{5}	10400.04	$6.40^{(1)}$	2420.88
40	118	14	4.00^{5}	10400.04	3.20	1597.95
40	196	4	8.80^{4}	8574.96	8.80	406.33
40	196	8	5.60^{3}	6373.35	5.00	469.85
40	196	16	3.20^{1}	3267.06	3.00	1186.14
40	274	5	7.40^{2}	4384.88	7.00	313.11
40	274	9	4.80^{4}	8931.32	4.00	58.81
40	274	17	3.40^{2}	4855.44	3.00	4882.92
50	173	4	12.40^{3}	6583.83	$12.60^{(2)}$	7164.37
50	173	8	8.40^{5}	10400.05	$9.20^{(4)}$	8555.60
50	295	5	9.60^{2}	5371.35	9.00	1625.25
50	295	9	6.40^{5}	10400.13	$5.20^{(1)}$	3045.47
50	295	17	4.00^{5}	10400.09	3.00	3358.94

Table 4.3: Branch-and-cut results for both formulations.

Vertices	Edges	Colors	Time	GSEC Cost	Exact Cost	GRASP Cost	Reoptimization Cost
20	39	3	0.06	9.73%	0.00%	0.85%	89.42%
20	39	6	3.20	10.77%	1.44%	0.72%	87.07%
20	39	11	55.68	14.90%	9.78%	0.53%	74.79%
20	58	3	1.88	6.58%	1.42%	0.62%	91.37%
20	58	6	3.92	4.31%	2.44%	0.78%	92.47%
20	58	12	0.40	11.30%	4.36%	0.76%	83.59%
20	77	4	3.64	1.50%	0.00%	0.75%	97.74%
20	77	7	1.23	1.94%	0.00%	1.30%	96.76%
20	77	13	1.57	11.13%	0.00%	1.00%	87.87%
30	74	4	84.20	21.35%	0.44%	0.72%	77.49%
30	74	7	5233.04	0.10%	0.04%	0.28%	99.58%
30	74	13	14.47	2.17%	0.49%	0.72%	96.63%
30	117	4	2241.10	0.39%	0.37%	0.71%	98.53%
30	117	7	3203.50	0.80%	0.00%	0.19%	99.01%
30	117	14	10400.04	0.00%	0.00%	0.04%	99.95%
30	161	4	1238.04	0.03%	0.01%	0.19%	99.77%
30	161	8	76.57	1.38%	0.00%	0.52%	98.10%
30	161	15	3699.03	0.02%	0.00%	0.07%	99.91%
40	118	4	3758.22	6.93%	0.01%	0.33%	92.73%
40	118	7	10400.04	0.01%	0.00%	0.09%	99.90%
40	118	14	10400.04	0.01%	0.00%	0.04%	99.95%
40	196	4	8574.96	0.01%	0.00%	0.16%	99.83%
40	196	8	6373.35	0.20%	0.00%	0.32%	99.47%
40	196	16	3267.06	0.47%	0.00%	0.52%	99.02%
40	274	5	4384.88	0.16%	0.00%	0.44%	99.41%
40	274	9	8931.32	0.01%	0.00%	0.13%	99.86%
40	274	17	4855.44	8.19%	0.00%	0.27%	91.54%
50	173	4	6583.83	9.12%	0.00%	0.38%	90.49%
50	173	8	10400.05	0.01%	0.00%	0.07%	99.92%
50	173	15	10400.06	0.00%	0.00%	0.04%	99.96%
50	295	5	5371.35	0.01%	0.00%	0.20%	99.79%
50	295	9	10400.13	0.01%	0.00%	0.04%	99.95%
50	295	17	10400.09	0.17%	0.00%	0.05%	99.78%
50	418	5	6492.46	0.16%	0.00%	0.11%	99.73%
50	418	9	10400.29	0.08%	0.00%	0.14%	99.78%
50	418	18	10400.15	1.02%	0.00%	0.14%	98.84%

Table 4.4: Branch-and-cut details for the line graph formulation.

Chapter 5

Conclusion

In this dissertation, we proposed a new line graph reformulation for the Rainbow Spanning Forest Problem (RSFP), as well as classes of valid inequalities for strengthening Linear Programming Relaxations for the problem. Our formulation finds RSFP solutions using an extended line graph, forgoing the need to index variables by the rainbow trees they belong to. We also propose a Branch-and-cut algorithm to solve the proposed formulation. We introduce exact and heuristic separation algorithms, as well as a GRASP heuristic to find upper bounds.

Our computational experiments show that the proposed formulation yields strong LPR lower bounds for the RSFP, dominating other formulations from the literature. However, these LPR bounds are expensive to be evaluated. As a result, our Branch-and-cut algorithm that relies on that formulation attains inferior results to the current best exact algorithm from the literature.

We found that most of the CPU time requirements for the evaluation of our new LPR bounds comes from Linear Programming reoptimization and not from separating the new classes of valid inequalities. In part, that can be associated to the fact that the line graph involves a considerably larger number of vertices and edges, and that reflects on the number of binary decision variables required by our reformulation.

Since our reformulation is strong and evaluating its dual bounds by directly solving Linear Programs is very time consuming, its seems that a promising direction for future research is the development of a Lagrangean Relaxation algorithm to approximate our dual bounds.

Our branch-and-cut algorithm can also be improved. For example, by better identifying and preventing tailing off effects, as well as by the design of better separation algorithms for the new classes of valid inequalities we introduced here.

Finally, there may be room for improvements in the formulation itself. Though they do not appear in our Branch-and-cut algorithm, lifted color inequalities (3.21) may be useful to improve LPR bounds more quickly. Investigating alternative ways to break symmetry may also prove worthwhile.

References

[1] Francesco Carrabs, Carmine Cerrone, Raffaele Cerulli, and Selene Silvestri. On the complexity of rainbow spanning forest problem. *Soft Computing*, 12:443–454, 2018.

[2] Francesco Carrabs, Carmine Cerrone, Raffaele Cerulli, and Selene Silvestri. The rainbow spanning forest problem. *Soft Computing*, 22:2765–2776, 2018.

[3] Francesco Carrabs, Raffaele Cerulli, and Paolo Dell'Olmo. A mathematical programming approach for the maximum labeled clique problem. *Procedia - Social and Behavioral Sciences*, 108:69–78, 2014. Operational Research for Development, Sustainability and Local Economies.

[4] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Raphael Yuster. Hardness and algorithms for rainbow connection. *Journal of Combinatorial Optimization*, 21(3):330–347, Apr 2011.

[5] Ruay-Shiung Chang and Leu Shing-Jiuan. The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277–282, 1997.

[6] Gary Chartrand, Garry L. Johns, Kathleen A. McKeon, and Ping Zhang. Rainbow connection in graphs. *Mathematica Bohemica*, 133(1):85–98, 2008.

[7] S. Consoli, J. A. Moreno Pérez, K. Darby-Dowman, and N. Mladenović. *Discrete Particle Swarm Optimization for the Minimum Labelling Steiner Tree Problem*, pages 313–322. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[8] Thiago Gouveia da Silva, Eduardo Queiroga, Luiz Satoru Ochi, Lucídio dos Anjos Formiga Cabral, Serigne Gueye, and Philippe Michelon. A hybrid metaheuristic for the minimum labeling spanning tree problem. *European Journal of Operational Research*, 274(1):22–34, 2019.

[9] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson. Solution of a large scale traveling salesman problem. *Operations Research*, 2(4):393–410, 1954.

[10] Nassim Dehouche. Devolutionary genetic algorithms with application to the minimum labeling steiner tree problem. *Evolving Systems*, 9(2):157–168, Jun 2018.

[11] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan 2002.

[12] Thomas Feo and Mauricio Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 03 1995.

[13] Daniele Ferone, Paola Festa, and Francesca Guerriero. The rainbow steiner tree problem. *Computers & Operations Research*, 139:105621, 2022.

[14] Yuri Frota, Simone Martins, and Jorge Moreno. A new approach for the rainbow spanning forest problem. *Soft Computing*, 24:3771–3780, 2020.

[15] Sudishna Ghoshal and Shyam Sundar. Two heuristics for the rainbow spanning forest problem. *European Journal of Operational Research*, 285(3):853–864, 2020.

[16] Sudishna Ghoshal and Shyam Sundar. A steady-state grouping genetic algorithm for the rainbow spanning forest problem. *SN Computer Science*, 4(4):321, Apr 2023.

[17] Hajo Broersma and Xueliang Li. Spanning Trees with many or few Colors in Edge-Colored Graphs. *Discussiones Mathematicae*, 17:259–269, 1997.

[18] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[19] Jun Li, MengChu Zhou, Qirui Sun, Xianzhong Dai, and Xiaolong Yu. Colored traveling salesman problem. *IEEE Transactions on Cybernetics*, 45(11):2390–2401, 2015.

[20] Xueliang Li and Xiaoyan Zhang. On the minimum monochromatic or multicolored subgraph partition problems. *Theoretical Computer Science*, 385(1):1–10, 2007.

[21] Abilio Lucena and Mauricio G.C Resende. Strong lower bounds for the prize collecting steiner problem in graphs. *Discrete Applied Mathematics*, 141(1):277–294, 2004. Brazilian Symposium on Graphs, Algorithms and Combinatorics.

[22] Ciaran McCreesh and Patrick Prosser. A parallel branch and bound algorithm for the maximum labelled clique problem. *Optimization Letters*, 9(5):949–960, Jun 2015.

[23] Jorge Moreno, Simone Martins, and Yuri Frota. A note on the rainbow cycle cover problem. *Networks*, 73(1):38–47, 2019.

[24] Selene Silvestri, Gilbert Laporte, and Raffaele Cerulli. The rainbow cycle cover problem. *Networks*, 68(4):260–270, 2016.

[25] Kei Uchizawa, Takanori Aoki, Takehiro Ito, Akira Suzuki, and Xiao Zhou. On the rainbow connectivity of graphs: Complexity and fpt algorithms. *Algorithmica*, 67(2):161–179, Oct 2013.

[26] Robin J. Wilson. *Introduction to Graph Theory*. Prentice Hall/Pearson, New York, 2010.

[27] Birol Yüceoğlu and Güvenç Şahin. A branch-and-price algorithm for the rainbow cycle cover problems. *Networks*, 74(1):3–15, 2019.

[28] Łukasz Kowalik and Juho Lauri. On finding rainbow and colorful paths. *Theoretical Computer Science*, 628:110–114, 2016.

Appendix A

Detailed Computational Results

In this appendix, we present detailed computational results to complement the results shown in Chapter 4. We recall that P_2 denotes Frota et al.'s formulation [14], while P_4 denotes our own line graph formulation. Results are displayed for all instances described in Section 4.6, with the computational environment described in the same section.

A.1 Detailed LPR Bounds Comparison

In this section, we compare the LPR bounds obtained by formulations P_4 and P_2 with the trivial lower bound LB and the best known solution $\overline{z}$, detailing the results shown in Section 4.7. Tables A.1, A.2, A.3, A.4 and A.5 show the results for instances with 20, 30, 40, 50 and 100 vertices, respectively. The first column of every table shows the instance's name. The second column shows the instance's best known upper bound, that is, the best solution found by either formulation. The third column shows its trivial lower bound. Finally, the last two columns show the LPR bounds obtained by formulations P_2 and P_4, respectively denoted by $w(P_2)$ and $w(P_4)$.

Instance	$\overline{z}$	LB	$w(P_2)$	$w(P_4)$
Rand_20_39_7_3.rnd	7.00	5.00	6.00	6.50
Rand_20_39_7_6.rnd	4.00	2.86	3.75	4.00
Rand_20_39_7_11.rnd	2.00	1.67	1.89	2.00
Rand_20_39_29_3.rnd	7.00	5.00	6.50	7.00
Rand_20_39_29_6.rnd	4.00	2.86	3.22	3.96
Rand_20_39_29_11.rnd	3.00	1.67	1.74	2.00
Rand_20_39_51_3.rnd	6.00	5.00	5.67	5.92
Rand_20_39_51_6.rnd	4.00	2.86	3.22	4.00
Rand_20_39_51_11.rnd	2.00	1.67	1.80	2.00
Rand_20_39_79_3.rnd	7.00	5.00	6.00	7.00
Rand_20_39_79_6.rnd	4.00	2.86	3.00	4.00
Rand_20_39_79_11.rnd	2.00	1.67	1.73	2.00
Rand_20_39_127_3.rnd	6.00	5.00	5.00	5.99
Rand_20_39_127_6.rnd	5.00	2.86	3.75	4.50
Rand_20_39_127_11.rnd	2.00	1.67	1.93	2.00
Rand_20_58_7_3.rnd	5.00	5.00	5.00	5.00
Rand_20_58_7_6.rnd	3.00	2.86	3.00	3.00
Rand_20_58_7_12.rnd	2.00	1.54	1.54	2.00
Rand_20_58_29_3.rnd	6.00	5.00	5.67	6.00
Rand_20_58_29_6.rnd	3.00	2.86	2.86	3.00
Rand_20_58_29_12.rnd	2.00	1.54	1.54	2.00
Rand_20_58_51_3.rnd	6.00	5.00	5.00	5.33
Rand_20_58_51_6.rnd	4.00	2.86	3.20	3.20
Rand_20_58_51_12.rnd	2.00	1.54	1.54	2.00
Rand_20_58_79_3.rnd	5.00	5.00	5.00	5.00
Rand_20_58_79_6.rnd	3.00	2.86	3.00	3.00
Rand_20_58_79_12.rnd	2.00	1.54	1.65	2.00
Rand_20_58_127_3.rnd	6.00	5.00	5.50	6.00
Rand_20_58_127_6.rnd	3.00	2.86	2.86	3.00
Rand_20_58_127_12.rnd	2.00	1.54	1.58	2.00
Rand_20_77_7_4.rnd	4.00	4.00	4.00	4.00
Rand_20_77_7_7.rnd	3.00	2.50	2.57	3.00
Rand_20_77_7_13.rnd	2.00	1.43	1.46	2.00
Rand_20_77_29_4.rnd	4.00	4.00	4.00	4.00
Rand_20_77_29_7.rnd	3.00	2.50	2.50	3.00
Rand_20_77_29_13.rnd	2.00	1.43	1.46	2.00
Rand_20_77_51_4.rnd	4.00	4.00	4.00	4.00
Rand_20_77_51_7.rnd	3.00	2.50	2.50	3.00
Rand_20_77_51_13.rnd	2.00	1.43	1.43	2.00
Rand_20_77_79_4.rnd	4.00	4.00	4.00	4.00
Rand_20_77_79_7.rnd	3.00	2.50	2.50	3.00
Rand_20_77_79_13.rnd	2.00	1.43	1.46	2.00
Rand_20_77_127_4.rnd	4.00	4.00	4.00	4.00
Rand_20_77_127_7.rnd	3.00	2.50	2.50	3.00
Rand_20_77_127_13.rnd	2.00	1.43	1.43	2.00

Table A.1: Bounds for instances with 20 vertices.

Instance	$\overline{z}$	LB	$w(P_2)$	$w(P_4)$
Rand_30_74_7_4.rnd	8.00	6.00	7.33	8.00
Rand_30_74_7_7.rnd	5.00	3.75	4.33	4.57
Rand_30_74_7_13.rnd	3.00	2.14	2.44	3.00
Rand_30_74_29_4.rnd	8.00	6.00	6.54	7.33
Rand_30_74_29_7.rnd	5.00	3.75	3.75	4.00
Rand_30_74_29_13.rnd	3.00	2.14	2.17	3.00
Rand_30_74_51_4.rnd	9.00	6.00	6.50	8.43
Rand_30_74_51_7.rnd	5.00	3.75	3.75	4.00
Rand_30_74_51_13.rnd	3.00	2.14	2.15	3.00
Rand_30_74_79_4.rnd	8.00	6.00	6.03	7.25
Rand_30_74_79_7.rnd	5.00	3.75	4.00	4.27
Rand_30_74_79_13.rnd	3.00	2.14	2.30	3.00
Rand_30_74_127_4.rnd	9.00	6.00	6.68	8.00
Rand_30_74_127_7.rnd	5.00	3.75	4.00	4.72
Rand_30_74_127_13.rnd	3.00	2.14	2.15	3.00
Rand_30_117_7_4.rnd	7.00	6.00	6.75	6.75
Rand_30_117_7_7.rnd	4.00	3.75	3.86	4.00
Rand_30_117_7_14.rnd	3.00	2.00	2.15	2.15
Rand_30_117_29_4.rnd	6.00	6.00	6.00	6.00
Rand_30_117_29_7.rnd	5.00	3.75	4.17	4.00
Rand_30_117_29_14.rnd	2.00	2.00	2.00	2.00
Rand_30_117_51_4.rnd	8.00	6.00	6.75	7.39
Rand_30_117_51_7.rnd	4.00	3.75	3.75	4.00
Rand_30_117_51_14.rnd	2.00	2.00	2.00	2.00
Rand_30_117_79_4.rnd	6.00	6.00	6.00	6.00
Rand_30_117_79_7.rnd	4.00	3.75	3.75	4.00
Rand_30_117_79_14.rnd	3.00	2.00	2.00	2.00
Rand_30_117_127_4.rnd	7.00	6.00	6.00	6.00
Rand_30_117_127_7.rnd	4.00	3.75	3.86	4.00
Rand_30_117_127_14.rnd	3.00	2.00	2.17	2.17
Rand_30_161_7_4.rnd	7.00	6.00	6.25	6.25
Rand_30_161_7_8.rnd	4.00	3.33	3.50	4.00
Rand_30_161_7_15.rnd	2.00	1.87	1.87	2.00
Rand_30_161_29_4.rnd	6.00	6.00	6.00	6.00
Rand_30_161_29_8.rnd	4.00	3.33	3.33	4.00
Rand_30_161_29_15.rnd	2.00	1.87	1.87	2.00
Rand_30_161_51_4.rnd	6.00	6.00	6.00	6.00
Rand_30_161_51_8.rnd	4.00	3.33	3.33	4.00
Rand_30_161_51_15.rnd	2.00	1.87	1.87	2.00
Rand_30_161_79_4.rnd	6.00	6.00	6.00	6.00
Rand_30_161_79_8.rnd	4.00	3.33	3.33	4.00
Rand_30_161_79_15.rnd	2.00	1.87	2.00	2.00
Rand_30_161_127_4.rnd	7.00	6.00	6.25	6.25
Rand_30_161_127_8.rnd	4.00	3.33	3.37	4.00
Rand_30_161_127_15.rnd	2.00	1.87	1.87	2.00

Table A.2: Bounds for instances with 30 vertices.

Instance	$\overline{z}$	LB	$w(P_2)$	$w(P_4)$
Rand_40_118_7_4.rnd	11.00	8.00	8.75	10.14
Rand_40_118_7_7.rnd	6.00	5.00	5.14	5.14
Rand_40_118_7_14.rnd	3.00	2.67	2.85	3.00
Rand_40_118_29_4.rnd	11.00	8.00	9.08	10.77
Rand_40_118_29_7.rnd	6.00	5.00	5.00	5.00
Rand_40_118_29_14.rnd	3.00	2.67	2.92	3.00
Rand_40_118_51_4.rnd	9.00	8.00	8.25	8.82
Rand_40_118_51_7.rnd	6.00	5.00	5.33	5.33
Rand_40_118_51_14.rnd	3.00	2.67	2.83	3.00
Rand_40_118_79_4.rnd	10.00	8.00	8.25	9.62
Rand_40_118_79_7.rnd	7.00	5.00	5.00	6.00
Rand_40_118_79_14.rnd	4.00	2.67	2.71	3.00
Rand_40_118_127_4.rnd	10.00	8.00	8.50	10.00
Rand_40_118_127_7.rnd	7.00	5.00	5.40	5.60
Rand_40_118_127_14.rnd	3.00	2.67	2.71	3.00
Rand_40_196_7_4.rnd	8.00	8.00	8.00	8.00
Rand_40_196_7_8.rnd	5.00	4.44	4.44	5.00
Rand_40_196_7_16.rnd	3.00	2.35	2.35	3.00
Rand_40_196_29_4.rnd	9.00	8.00	8.50	8.25
Rand_40_196_29_8.rnd	5.00	4.44	4.71	5.00
Rand_40_196_29_16.rnd	3.00	2.35	2.43	3.00
Rand_40_196_51_4.rnd	9.00	8.00	8.25	8.25
Rand_40_196_51_8.rnd	5.00	4.44	4.50	5.00
Rand_40_196_51_16.rnd	3.00	2.35	2.35	3.00
Rand_40_196_79_4.rnd	9.00	8.00	8.00	8.00
Rand_40_196_79_8.rnd	5.00	4.44	4.44	5.00
Rand_40_196_79_16.rnd	4.00	2.35	2.35	3.00
Rand_40_196_127_4.rnd	9.00	8.00	8.25	8.00
Rand_40_196_127_8.rnd	5.00	4.44	4.62	5.00
Rand_40_196_127_16.rnd	3.00	2.35	2.35	3.00
Rand_40_274_7_5.rnd	7.00	6.67	6.67	7.00
Rand_40_274_7_9.rnd	4.00	4.00	4.00	4.00
Rand_40_274_7_17.rnd	3.00	2.22	2.22	3.00
Rand_40_274_29_5.rnd	7.00	6.67	6.67	7.00
Rand_40_274_29_9.rnd	4.00	4.00	4.00	4.00
Rand_40_274_29_17.rnd	3.00	2.22	2.22	3.00
Rand_40_274_51_5.rnd	7.00	6.67	6.67	7.00
Rand_40_274_51_9.rnd	4.00	4.00	4.00	4.00
Rand_40_274_51_17.rnd	3.00	2.22	2.22	3.00
Rand_40_274_79_5.rnd	7.00	6.67	6.67	7.00
Rand_40_274_79_9.rnd	4.00	4.00	4.00	4.00
Rand_40_274_79_17.rnd	3.00	2.22	2.22	3.00
Rand_40_274_127_5.rnd	7.00	6.67	6.67	7.00
Rand_40_274_127_9.rnd	4.00	4.00	4.00	4.00
Rand_40_274_127_17.rnd	3.00	2.22	2.25	3.00

Table A.3: Bounds for instances with 40 vertices.

Instance	$\overline{z}$	LB	$w(P_2)$	$w(P_4)$
Rand_50_173_7_4.rnd	15.00	10.00	13.00	14.64
Rand_50_173_7_8.rnd	10.00*	5.56	6.91	6.95
Rand_50_173_7_15.rnd	4.00	3.12	3.14	4.00
Rand_50_173_29_4.rnd	11.00*	10.00	10.00	10.00
Rand_50_173_29_8.rnd	8.00*	5.56	5.75	6.00
Rand_50_173_29_15.rnd	6.00*	3.12	3.21	4.14
Rand_50_173_51_4.rnd	12.00	10.00	10.75	10.93
Rand_50_173_51_8.rnd	8.00*	5.56	6.50	6.33
Rand_50_173_51_15.rnd	4.00*	3.12	3.38	4.00
Rand_50_173_79_4.rnd	12.00	10.00	11.00	11.50
Rand_50_173_79_8.rnd	7.00	5.56	6.14	6.00
Rand_50_173_79_15.rnd	4.00	3.12	3.12	4.00
Rand_50_173_127_4.rnd	13.00*	10.00	10.33	10.69
Rand_50_173_127_8.rnd	10.00*	5.56	6.59	6.81
Rand_50_173_127_15.rnd	5.00*	3.12	3.31	4.00
Rand_50_295_7_5.rnd	9.00	8.33	9.00	9.00
Rand_50_295_7_9.rnd	5.00	5.00	5.00	5.00
Rand_50_295_7_17.rnd	3.00	2.78	2.78	3.00
Rand_50_295_29_5.rnd	9.00	8.33	8.33	9.00
Rand_50_295_29_9.rnd	5.00	5.00	5.00	5.00
Rand_50_295_29_17.rnd	3.00	2.78	2.78	3.00
Rand_50_295_51_5.rnd	9.00	8.33	8.60	9.00
Rand_50_295_51_9.rnd	5.00	5.00	5.00	5.00
Rand_50_295_51_17.rnd	3.00	2.78	2.88	3.00
Rand_50_295_79_5.rnd	9.00	8.33	8.33	9.00
Rand_50_295_79_9.rnd	5.00	5.00	5.00	5.00
Rand_50_295_79_17.rnd	3.00	2.78	2.78	3.00
Rand_50_295_127_5.rnd	9.00	8.33	8.33	9.00
Rand_50_295_127_9.rnd	5.00	5.00	5.00	5.00
Rand_50_295_127_17.rnd	3.00	2.78	2.78	3.00
Rand_50_418_7_5.rnd	9.00	8.33	8.33	9.00
Rand_50_418_7_9.rnd	5.00	5.00	5.00	5.00
Rand_50_418_7_18.rnd	3.00	2.63	2.63	3.00
Rand_50_418_29_5.rnd	9.00	8.33	8.33	9.00
Rand_50_418_29_9.rnd	5.00	5.00	5.00	5.00
Rand_50_418_29_18.rnd	3.00	2.63	2.63	3.00
Rand_50_418_51_5.rnd	9.00	8.33	8.33	9.00
Rand_50_418_51_9.rnd	5.00	5.00	5.00	5.00
Rand_50_418_51_18.rnd	3.00	2.63	2.63	3.00
Rand_50_418_79_5.rnd	9.00	8.33	8.33	9.00
Rand_50_418_79_9.rnd	5.00	5.00	5.00	5.00
Rand_50_418_79_18.rnd	3.00	2.63	2.63	3.00
Rand_50_418_127_5.rnd	9.00	8.33	8.40	9.00
Rand_50_418_127_9.rnd	5.00	5.00	5.00	5.00
Rand_50_418_127_18.rnd	3.00	2.63	2.63	3.00

Table A.4: Bounds for instances with 50 vertices.

Instance	$\overline{z}$	LB	$w(P_2)$	$w(P_4)$
Rand_100_595_7_5.rnd	22.00	16.67	17.25	16.67
Rand_100_595_7_10.rnd	14.00	9.09	9.12	9.09
Rand_100_595_7_19.rnd	9.00	5.00	5.28	5.00
Rand_100_595_29_5.rnd	24.00	16.67	18.20	16.67
Rand_100_595_29_10.rnd	13.00	9.09	9.09	9.09
Rand_100_595_29_19.rnd	10.00	5.00	5.10	5.00
Rand_100_595_51_5.rnd	22.00	16.67	19.73	19.33
Rand_100_595_51_10.rnd	13.00	9.09	9.33	9.09
Rand_100_595_51_19.rnd	9.00	5.00	5.00	5.00
Rand_100_595_79_5.rnd	23.00	16.67	17.60	16.67
Rand_100_595_79_10.rnd	13.00	9.09	9.09	9.09
Rand_100_595_79_19.rnd	9.00	5.00	5.00	5.00
Rand_100_595_127_5.rnd	19.00	16.67	16.67	16.67
Rand_100_595_127_10.rnd	13.00	9.09	9.33	9.09
Rand_100_595_127_19.rnd	11.00	5.00	5.00	5.00
Rand_100_1090_7_6.rnd	19.00	14.29	14.29	14.29
Rand_100_1090_7_11.rnd	13.00	8.33	8.33	8.33
Rand_100_1090_7_21.rnd	11.00	4.55	4.62	4.55
Rand_100_1090_29_6.rnd	18.00	14.29	14.29	14.29
Rand_100_1090_29_11.rnd	13.00	8.33	8.33	8.33
Rand_100_1090_29_21.rnd	11.00	4.55	4.57	4.55
Rand_100_1090_51_6.rnd	19.00	14.29	14.29	14.29
Rand_100_1090_51_11.rnd	18.00	8.33	8.60	8.33
Rand_100_1090_51_21.rnd	9.00	4.55	4.55	4.55
Rand_100_1090_79_6.rnd	18.00	14.29	14.29	14.29
Rand_100_1090_79_11.rnd	17.00	8.33	8.33	8.33
Rand_100_1090_79_21.rnd	9.00	4.55	4.55	4.55
Rand_100_1090_127_6.rnd	21.00	14.29	14.29	14.29
Rand_100_1090_127_11.rnd	13.00	8.33	8.33	8.33
Rand_100_1090_127_21.rnd	9.00	4.55	4.55	4.55
Rand_100_1585_7_6.rnd	23.00	14.29	14.29	14.29
Rand_100_1585_7_11.rnd	15.00	8.33	8.33	8.33
Rand_100_1585_7_22.rnd	N/A	4.35	N/A	4.35
Rand_100_1585_29_6.rnd	26.00	14.29	14.33	14.29
Rand_100_1585_29_11.rnd	16.00	8.33	8.33	8.33
Rand_100_1585_29_22.rnd	N/A	4.35	4.35	4.35
Rand_100_1585_51_6.rnd	19.00	14.29	14.29	14.29
Rand_100_1585_51_11.rnd	16.00	8.33	8.33	8.33
Rand_100_1585_51_22.rnd	N/A	4.35	N/A	4.35
Rand_100_1585_79_6.rnd	23.00	14.29	14.29	14.29
Rand_100_1585_79_11.rnd	19.00	8.33	8.33	8.33
Rand_100_1585_79_22.rnd	11.00	4.35	4.35	4.35
Rand_100_1585_127_6.rnd	17.00	14.29	14.29	14.29
Rand_100_1585_127_11.rnd	N/A	8.33	8.33	8.33
Rand_100_1585_127_22.rnd	11.00	4.35	N/A	4.35

Table A.5: Bounds for instances with 100 vertices.

A.2 Detailed Linear Programming Costs

In this section, we show detailed information about the separation costs of our line graph formulation, detailing the results shown in Section 4.7. Tables A.6, A.7, A.8, A.9 and A.10 show the results for instances with 20, 30, 40, 50 and 100 vertices, respectively. The first column of every table shows the instance's name. The second column shows the total time required to solve that instance, up to the time limit of 10400 seconds. The third column shows the percentage of the total time spent on GSEC separation. The fourth and fifth column show the percentages of the total time spent on exact and heuristic CSEC separation, respectively. Finally, the last column shows the percentage of the total time spent on reoptimization, that is, on solving the linear programming subproblems.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_20_39_7_3.rnd	0.11	0.00%	0.00%	0.28%	99.72%
Rand_20_39_7_6.rnd	91.02	19.35%	2.16%	0.12%	78.36%
Rand_20_39_7_11.rnd	13.05	8.55%	31.72%	0.17%	59.56%
Rand_20_39_29_3.rnd	18.35	31.40%	23.77%	0.09%	44.73%
Rand_20_39_29_6.rnd	348.80	10.28%	15.73%	0.04%	73.96%
Rand_20_39_29_11.rnd	0.23	27.05%	0.00%	1.07%	71.88%
Rand_20_39_51_3.rnd	6.52	30.15%	11.03%	0.41%	58.41%
Rand_20_39_51_6.rnd	10.08	13.90%	26.46%	0.17%	59.47%
Rand_20_39_51_11.rnd	10.91	34.66%	3.98%	0.23%	61.12%
Rand_20_39_79_3.rnd	117.66	16.60%	15.90%	0.09%	67.41%
Rand_20_39_79_6.rnd	22.63	10.44%	18.61%	0.12%	70.82%
Rand_20_39_79_11.rnd	1.08	79.99%	0.00%	0.78%	19.24%
Rand_20_39_127_3.rnd	59.04	23.19%	16.69%	0.08%	60.03%
Rand_20_39_127_6.rnd	37.62	11.66%	21.97%	0.08%	66.28%
Rand_20_39_127_11.rnd	1.90	3.97%	2.60%	0.33%	93.11%
Rand_20_58_7_3.rnd	32.99	9.97%	11.33%	0.28%	78.43%
Rand_20_58_7_6.rnd	6.78	14.47%	15.87%	0.64%	69.03%
Rand_20_58_7_12.rnd	19.02	33.31%	0.36%	0.51%	65.81%
Rand_20_58_29_3.rnd	45.27	6.22%	11.01%	0.32%	82.44%
Rand_20_58_29_6.rnd	54.74	8.14%	17.28%	0.27%	74.31%
Rand_20_58_29_12.rnd	48.43	18.68%	0.14%	0.38%	80.80%
Rand_20_58_51_3.rnd	2829.76	1.45%	0.37%	0.04%	98.14%
Rand_20_58_51_6.rnd	28.31	11.52%	19.45%	0.40%	68.63%
Rand_20_58_51_12.rnd	0.28	0.00%	0.00%	0.25%	99.75%
Rand_20_58_79_3.rnd	44.80	4.29%	11.74%	0.27%	83.70%
Rand_20_58_79_6.rnd	36.64	4.54%	13.36%	0.33%	81.78%
Rand_20_58_79_12.rnd	64.29	24.34%	1.75%	0.28%	73.63%
Rand_20_58_127_3.rnd	46.41	4.30%	4.75%	0.27%	90.68%
Rand_20_58_127_6.rnd	35.53	9.84%	16.44%	0.28%	73.43%
Rand_20_58_127_12.rnd	86.01	2.88%	3.28%	0.35%	93.49%
Rand_20_77_7_4.rnd	15.83	9.21%	10.36%	0.74%	79.69%
Rand_20_77_7_7.rnd	7.99	6.91%	4.49%	0.80%	87.80%
Rand_20_77_7_13.rnd	42.33	2.30%	3.66%	0.45%	93.59%
Rand_20_77_29_4.rnd	7.35	6.01%	7.94%	0.80%	85.24%
Rand_20_77_29_7.rnd	14.42	9.67%	14.88%	0.66%	74.79%
Rand_20_77_29_13.rnd	13.58	6.23%	6.38%	0.44%	86.95%
Rand_20_77_51_4.rnd	5.87	7.12%	5.83%	0.76%	86.29%
Rand_20_77_51_7.rnd	45.97	4.23%	3.31%	0.45%	92.01%
Rand_20_77_79_4.rnd	49.56	4.62%	4.53%	0.37%	90.48%
Rand_20_77_79_7.rnd	7.83	5.40%	18.77%	0.82%	75.01%
Rand_20_77_79_13.rnd	87.78	4.27%	2.71%	0.28%	92.74%
Rand_20_77_127_4.rnd	8.00	3.97%	4.63%	0.75%	90.65%
Rand_20_77_127_7.rnd	35.20	10.55%	19.00%	0.52%	69.93%

Table A.6: Separation costs for instances with 20 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_30_74_7_4.rnd	34.84	34.66%	0.00%	0.70%	64.64%
Rand_30_74_7_7.rnd	10400.10	5.99%	0.39%	0.03%	93.59%
Rand_30_74_7_13.rnd	3315.99	1.34%	1.69%	0.04%	96.93%
Rand_30_74_29_4.rnd	37.94	47.89%	4.24%	0.51%	47.36%
Rand_30_74_29_7.rnd	10400.03	0.62%	0.37%	0.03%	98.99%
Rand_30_74_29_13.rnd	2489.33	1.89%	2.44%	0.07%	95.61%
Rand_30_74_51_4.rnd	1170.27	14.39%	1.11%	0.09%	84.40%
Rand_30_74_51_7.rnd	10400.10	0.48%	0.09%	0.03%	99.40%
Rand_30_74_51_13.rnd	624.75	3.55%	3.71%	0.15%	92.60%
Rand_30_74_79_4.rnd	2891.89	10.06%	0.14%	0.06%	89.74%
Rand_30_74_79_7.rnd	10400.12	3.11%	0.23%	0.03%	96.63%
Rand_30_74_79_13.rnd	304.05	5.29%	1.61%	0.19%	92.91%
Rand_30_74_127_4.rnd	1644.96	12.79%	0.79%	0.06%	86.35%
Rand_30_74_127_7.rnd	10400.02	5.92%	0.14%	0.03%	93.91%
Rand_30_74_127_13.rnd	274.38	5.36%	4.29%	0.19%	90.16%
Rand_30_117_7_4.rnd	6749.60	6.98%	0.02%	0.10%	92.90%
Rand_30_117_7_7.rnd	10400.16	0.58%	0.27%	0.04%	99.11%
Rand_30_117_7_14.rnd	10400.20	0.86%	0.02%	0.04%	99.08%
Rand_30_117_29_4.rnd	466.24	2.52%	0.41%	0.24%	96.83%
Rand_30_117_29_7.rnd	10400.01	0.58%	0.27%	0.04%	99.10%
Rand_30_117_29_14.rnd	10400.14	0.91%	1.31%	0.04%	97.74%
Rand_30_117_51_4.rnd	10400.37	8.17%	0.00%	0.07%	91.76%
Rand_30_117_51_7.rnd	10400.19	0.53%	0.27%	0.05%	99.15%
Rand_30_117_51_14.rnd	10400.22	0.94%	0.59%	0.04%	98.42%
Rand_30_117_79_4.rnd	786.08	1.53%	0.87%	0.19%	97.42%
Rand_30_117_79_7.rnd	10400.04	0.52%	0.22%	0.05%	99.22%
Rand_30_117_79_14.rnd	10400.15	0.83%	1.16%	0.04%	97.97%
Rand_30_117_127_4.rnd	10400.12	0.81%	0.08%	0.05%	99.06%
Rand_30_117_127_7.rnd	7163.76	0.66%	0.46%	0.05%	98.83%
Rand_30_117_127_14.rnd	10400.13	0.94%	0.02%	0.04%	99.00%
Rand_30_161_7_4.rnd	6324.85	1.08%	0.07%	0.09%	98.77%
Rand_30_161_7_8.rnd	10400.25	0.67%	0.19%	0.10%	99.05%
Rand_30_161_7_15.rnd	10400.17	0.68%	0.68%	0.09%	98.55%
Rand_30_161_29_4.rnd	1698.50	0.66%	0.16%	0.14%	99.05%
Rand_30_161_29_8.rnd	979.21	2.05%	0.74%	0.38%	96.83%
Rand_30_161_29_15.rnd	3040.47	2.02%	1.17%	0.21%	96.60%
Rand_30_161_51_4.rnd	3428.13	0.69%	0.63%	0.12%	98.56%
Rand_30_161_51_8.rnd	7774.56	0.86%	0.30%	0.13%	98.71%
Rand_30_161_51_15.rnd	9018.33	0.82%	0.71%	0.09%	98.38%
Rand_30_161_79_4.rnd	348.99	1.15%	0.60%	0.30%	97.95%
Rand_30_161_79_8.rnd	7765.73	0.63%	0.18%	0.10%	99.09%
Rand_30_161_79_15.rnd	10400.06	1.04%	0.63%	0.08%	98.26%
Rand_30_161_127_4.rnd	10400.05	0.70%	0.06%	0.08%	99.16%
Rand_30_161_127_8.rnd	10400.21	0.68%	0.25%	0.10%	98.97%
Rand_30_161_127_15.rnd	10400.08	0.85%	0.70%	0.09%	98.36%

Table A.7: Separation costs for instances with 30 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_40_118_7_4.rnd	10400.02	6.40%	0.00%	0.10%	93.50%
Rand_40_118_7_7.rnd	10400.13	1.06%	0.01%	0.04%	98.89%
Rand_40_118_7_14.rnd	10400.10	0.86%	0.19%	0.04%	98.91%
Rand_40_118_29_4.rnd	10400.56	9.06%	0.00%	0.09%	90.85%
Rand_40_118_29_7.rnd	10400.06	0.63%	0.09%	0.04%	99.24%
Rand_40_118_29_14.rnd	10400.17	0.85%	0.17%	0.04%	98.93%
Rand_40_118_51_4.rnd	10400.41	5.34%	0.00%	0.10%	94.56%
Rand_40_118_51_7.rnd	10400.11	4.46%	0.00%	0.06%	95.48%
Rand_40_118_51_14.rnd	10400.14	0.79%	0.27%	0.05%	98.89%
Rand_40_118_79_4.rnd	10400.02	8.05%	0.03%	0.07%	91.84%
Rand_40_118_79_7.rnd	10400.11	2.61%	0.05%	0.06%	97.29%
Rand_40_118_79_14.rnd	10400.16	1.00%	0.40%	0.05%	98.56%
Rand_40_118_127_4.rnd	2065.05	10.07%	0.00%	0.26%	89.67%
Rand_40_118_127_7.rnd	10400.47	3.99%	0.00%	0.06%	95.96%
Rand_40_118_127_14.rnd	10400.07	0.80%	0.29%	0.05%	98.86%
Rand_40_196_7_4.rnd	10400.11	0.52%	0.07%	0.07%	99.34%
Rand_40_196_7_8.rnd	10400.48	0.58%	0.04%	0.14%	99.24%
Rand_40_196_7_16.rnd	10400.29	1.02%	0.11%	0.16%	98.72%
Rand_40_196_29_4.rnd	10400.16	0.70%	0.01%	0.10%	99.20%
Rand_40_196_29_8.rnd	10400.18	0.48%	0.05%	0.11%	99.36%
Rand_40_196_29_16.rnd	10400.39	0.94%	0.23%	0.14%	98.69%
Rand_40_196_51_4.rnd	10400.10	0.42%	0.03%	0.09%	99.46%
Rand_40_196_51_8.rnd	10400.14	0.57%	0.02%	0.14%	99.26%
Rand_40_196_51_16.rnd	10400.15	0.95%	0.08%	0.14%	98.82%
Rand_40_196_79_4.rnd	10400.18	0.73%	0.09%	0.11%	99.07%
Rand_40_196_79_8.rnd	10400.22	0.52%	0.02%	0.14%	99.31%
Rand_40_196_79_16.rnd	10400.25	0.97%	0.14%	0.15%	98.73%
Rand_40_196_127_4.rnd	10400.04	0.88%	0.01%	0.09%	99.02%
Rand_40_196_127_8.rnd	10400.02	0.46%	0.07%	0.09%	99.38%
Rand_40_196_127_16.rnd	10400.15	0.67%	0.15%	0.10%	99.08%
Rand_40_274_7_5.rnd	10400.23	0.54%	0.01%	0.24%	99.21%
Rand_40_274_7_9.rnd	10400.11	0.84%	0.01%	0.21%	98.94%
Rand_40_274_7_17.rnd	10400.25	1.40%	0.16%	0.27%	98.17%
Rand_40_274_29_5.rnd	10400.05	0.51%	0.04%	0.24%	99.22%
Rand_40_274_29_9.rnd	10400.23	0.69%	0.02%	0.23%	99.06%
Rand_40_274_29_17.rnd	10400.62	1.46%	0.15%	0.24%	98.15%
Rand_40_274_51_5.rnd	10400.03	0.49%	0.03%	0.22%	99.26%
Rand_40_274_51_9.rnd	10400.20	0.86%	0.06%	0.19%	98.89%
Rand_40_274_51_17.rnd	10400.87	1.57%	0.07%	0.29%	98.07%
Rand_40_274_79_5.rnd	10400.25	0.30%	0.01%	0.20%	99.49%
Rand_40_274_79_9.rnd	10400.31	0.68%	0.04%	0.22%	99.06%
Rand_40_274_79_17.rnd	10400.34	2.08%	0.11%	0.28%	97.54%
Rand_40_274_127_5.rnd	10400.05	0.37%	0.03%	0.20%	99.40%
Rand_40_274_127_9.rnd	10400.17	0.70%	0.03%	0.21%	99.07%
Rand_40_274_127_17.rnd	10401.57	1.64%	0.05%	0.22%	98.09%

Table A.8: Separation costs for instances with 40 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_50_173_7_4.rnd	10400.02	9.63%	0.00%	0.19%	90.18%
Rand_50_173_7_8.rnd	10400.47	34.61%	0.00%	0.14%	65.25%
Rand_50_173_7_15.rnd	10400.20	0.58%	0.03%	0.09%	99.30%
Rand_50_173_29_4.rnd	10400.14	1.10%	0.00%	0.09%	98.81%
Rand_50_173_29_8.rnd	10400.03	0.50%	0.01%	0.08%	99.42%
Rand_50_173_29_15.rnd	10400.13	0.72%	0.00%	0.06%	99.22%
Rand_50_173_51_4.rnd	10400.51	13.70%	0.00%	0.16%	86.14%
Rand_50_173_51_8.rnd	10400.14	0.69%	0.00%	0.06%	99.25%
Rand_50_173_51_15.rnd	10400.27	0.62%	0.01%	0.10%	99.27%
Rand_50_173_79_4.rnd	10400.06	7.00%	0.03%	0.11%	92.85%
Rand_50_173_79_8.rnd	10400.16	0.44%	0.01%	0.08%	99.48%
Rand_50_173_79_15.rnd	10400.22	0.80%	0.01%	0.12%	99.07%
Rand_50_173_127_4.rnd	10400.08	15.97%	0.00%	0.10%	83.93%
Rand_50_173_127_8.rnd	10401.21	39.73%	0.00%	0.06%	60.21%
Rand_50_173_127_15.rnd	10400.11	0.58%	0.02%	0.10%	99.30%
Rand_50_295_7_5.rnd	10400.22	0.37%	0.00%	0.18%	99.45%
Rand_50_295_7_9.rnd	10400.26	0.45%	0.00%	0.18%	99.37%
Rand_50_295_7_17.rnd	10400.25	1.11%	0.00%	0.20%	98.69%
Rand_50_295_29_5.rnd	10400.04	0.46%	0.02%	0.23%	99.28%
Rand_50_295_29_9.rnd	10400.23	0.50%	0.00%	0.15%	99.35%
Rand_50_295_29_17.rnd	10400.49	1.11%	0.01%	0.21%	98.67%
Rand_50_295_51_5.rnd	10400.26	0.40%	0.00%	0.20%	99.39%
Rand_50_295_51_9.rnd	10400.08	0.40%	0.00%	0.16%	99.44%
Rand_50_295_51_17.rnd	10400.26	1.66%	0.00%	0.20%	98.15%
Rand_50_295_79_5.rnd	10400.23	0.22%	0.00%	0.25%	99.52%
Rand_50_295_79_9.rnd	10400.28	0.31%	0.00%	0.29%	99.40%
Rand_50_295_79_17.rnd	10400.22	0.31%	0.00%	0.20%	99.49%
Rand_50_295_127_5.rnd	10400.70	0.94%	0.00%	0.21%	98.85%
Rand_50_295_127_9.rnd	10400.19	0.42%	0.01%	0.24%	99.32%
Rand_50_295_127_17.rnd	10400.21	0.54%	0.00%	0.16%	99.30%
Rand_50_418_7_5.rnd	10400.26	1.31%	0.00%	0.21%	98.49%
Rand_50_418_7_9.rnd	10400.55	0.18%	0.00%	0.23%	99.60%
Rand_50_418_7_18.rnd	10400.25	0.58%	0.00%	0.22%	99.20%
Rand_50_418_29_5.rnd	10400.23	4.58%	0.00%	0.16%	95.26%
Rand_50_418_29_9.rnd	10400.30	0.29%	0.00%	0.29%	99.42%
Rand_50_418_29_18.rnd	10400.25	0.53%	0.01%	0.18%	99.29%
Rand_50_418_51_5.rnd	10400.07	3.97%	0.01%	0.33%	95.69%
Rand_50_418_51_9.rnd	10400.34	0.26%	0.00%	0.34%	99.40%
Rand_50_418_51_18.rnd	10400.20	0.27%	0.00%	0.20%	99.53%
Rand_50_418_79_5.rnd	10400.21	8.40%	0.00%	0.23%	91.37%
Rand_50_418_79_9.rnd	10400.27	0.24%	0.00%	0.26%	99.50%
Rand_50_418_79_18.rnd	10400.48	0.36%	0.00%	0.34%	99.30%
Rand_50_418_127_5.rnd	10400.57	2.77%	0.01%	0.19%	97.03%
Rand_50_418_127_9.rnd	10400.04	0.24%	0.00%	0.19%	99.57%
Rand_50_418_127_18.rnd	10400.84	0.28%	0.00%	0.20%	99.52%
Rand_50_418_127_18.rnd	10400.35	5.14%	0.03%	0.35%	94.48%

Table A.9: Separation costs for instances with 50 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_100_595_7_5.rnd	10400.35	0.73%	0.00%	0.18%	99.09%
Rand_100_595_7_10.rnd	10400.42	0.07%	0.00%	0.27%	99.66%
Rand_100_595_7_19.rnd	10400.31	1.18%	0.00%	0.19%	98.63%
Rand_100_595_29_5.rnd	10400.05	0.07%	0.00%	0.14%	99.79%
Rand_100_595_29_10.rnd	10400.40	0.22%	0.00%	0.38%	99.41%
Rand_100_595_29_19.rnd	10400.21	5.20%	0.00%	0.25%	94.54%
Rand_100_595_51_5.rnd	10413.07	83.56%	0.00%	0.63%	15.81%
Rand_100_595_51_10.rnd	10400.34	0.51%	0.00%	0.25%	99.24%
Rand_100_595_51_19.rnd	10400.13	5.30%	0.00%	0.23%	94.47%
Rand_100_595_79_5.rnd	10400.33	0.04%	0.00%	0.17%	99.79%
Rand_100_595_79_10.rnd	10400.30	0.20%	0.00%	0.21%	99.59%
Rand_100_595_79_19.rnd	10400.34	0.18%	0.00%	0.27%	99.55%
Rand_100_595_127_5.rnd	10400.35	0.29%	0.00%	0.20%	99.51%
Rand_100_595_127_10.rnd	10400.19	0.13%	0.00%	0.22%	99.65%
Rand_100_595_127_19.rnd	10400.33	2.52%	0.00%	0.26%	97.22%
Rand_100_1090_7_6.rnd	10401.18	0.00%	0.00%	0.16%	99.84%
Rand_100_1090_7_11.rnd	10400.25	0.42%	0.00%	0.16%	99.42%
Rand_100_1090_7_21.rnd	10400.23	7.93%	0.00%	0.05%	92.02%
Rand_100_1090_29_6.rnd	10400.24	0.00%	0.00%	0.22%	99.78%
Rand_100_1090_29_11.rnd	10400.89	2.06%	0.00%	0.16%	97.78%
Rand_100_1090_29_21.rnd	10403.69	6.20%	0.00%	0.16%	93.64%
Rand_100_1090_51_6.rnd	10400.99	0.46%	0.00%	0.08%	99.46%
Rand_100_1090_51_11.rnd	10400.56	1.96%	0.00%	0.03%	98.01%
Rand_100_1090_51_21.rnd	10401.29	3.63%	0.00%	0.02%	96.35%
Rand_100_1090_79_6.rnd	10401.02	0.00%	0.00%	0.16%	99.84%
Rand_100_1090_79_11.rnd	10401.69	1.06%	0.00%	0.04%	98.90%
Rand_100_1090_79_21.rnd	10400.61	0.00%	0.00%	0.06%	99.94%
Rand_100_1090_127_6.rnd	10401.01	0.00%	0.00%	0.18%	99.82%
Rand_100_1090_127_11.rnd	10402.60	3.90%	0.00%	0.13%	95.97%
Rand_100_1090_127_21.rnd	10400.23	0.00%	0.00%	0.04%	99.96%
Rand_100_1585_7_6.rnd	10400.44	0.00%	0.00%	0.19%	99.81%
Rand_100_1585_7_11.rnd	10400.20	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_7_22.rnd	10400.66	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_29_6.rnd	10400.42	0.00%	0.00%	0.26%	99.74%
Rand_100_1585_29_11.rnd	10400.44	0.00%	0.00%	0.04%	99.96%
Rand_100_1585_29_22.rnd	10400.31	0.00%	0.00%	6.87%	93.13%
Rand_100_1585_51_6.rnd	10400.38	0.00%	0.00%	0.22%	99.78%
Rand_100_1585_51_11.rnd	10401.59	0.00%	0.00%	0.23%	99.77%
Rand_100_1585_51_22.rnd	10400.45	0.46%	0.00%	0.00%	99.54%
Rand_100_1585_79_6.rnd	10400.44	1.09%	0.00%	0.12%	98.80%
Rand_100_1585_79_11.rnd	10400.42	0.00%	0.00%	0.18%	99.82%
Rand_100_1585_79_22.rnd	10400.49	0.90%	0.00%	0.00%	99.10%
Rand_100_1585_127_6.rnd	10400.80	0.30%	0.00%	0.30%	99.40%
Rand_100_1585_127_11.rnd	10400.27	2.70%	0.00%	0.04%	97.26%
Rand_100_1585_127_22.rnd	10400.38	0.35%	0.00%	0.16%	99.49%

Table A.10: Separation costs for instances with 50 vertices.

A.3 Detailed Branch-and-Cut Results

In this section, we compare the optimal solutions found by Frota et al.'s formulation with our own, detailing the results shown in Section 4.6. Tables A.11, A.12, A.13, A.14 and A.15 show the results for instances with 20, 30, 40, 50 and 100 vertices, respectively. The first column of every table shows the instance's name. The second column shows the optimal solution found by our branch-and-cut algorithm, denoted by BC^L, while the third column shows the time required to find that solution. The fourth column shows the optimal solution found by Frota et al.'s formulation, denoted by BC^C, while the fifth column shows the time required to find that solution. When either formulation times out without finding any solution at all, we fill the corresponding cells with "N/A".

Instance	BC^L	Time	BC^C	Time
Rand_20_39_7_3.rnd	7.00	0.09	7.00	0.36
Rand_20_39_7_6.rnd	4.00	0.98	4.00	0.98
Rand_20_39_7_11.rnd	2.00	0.12	2.00	0.44
Rand_20_39_29_3.rnd	7.00	0.07	7.00	0.24
Rand_20_39_29_6.rnd	4.00	1.77	4.00	0.82
Rand_20_39_29_11.rnd	3.00	270.82	3.00	0.97
Rand_20_39_51_3.rnd	6.00	0.05	6.00	0.57
Rand_20_39_51_6.rnd	4.00	0.96	4.00	0.52
Rand_20_39_51_11.rnd	2.00	0.15	2.00	0.43
Rand_20_39_79_3.rnd	7.00	0.04	7.00	0.99
Rand_20_39_79_6.rnd	4.00	6.99	4.00	1.08
Rand_20_39_79_11.rnd	2.00	1.29	2.00	0.84
Rand_20_39_127_3.rnd	6.00	0.08	6.00	0.99
Rand_20_39_127_6.rnd	5.00	5.30	5.00	0.84
Rand_20_39_127_11.rnd	2.00	6.02	2.00	0.42
Rand_20_58_7_3.rnd	5.00	0.61	5.00	0.52
Rand_20_58_7_6.rnd	3.00	3.60	3.00	0.16
Rand_20_58_7_12.rnd	2.00	0.07	2.00	3.69
Rand_20_58_29_3.rnd	6.00	0.16	6.00	0.69
Rand_20_58_29_6.rnd	3.00	4.01	3.00	0.86
Rand_20_58_29_12.rnd	2.00	0.19	2.00	0.45
Rand_20_58_51_3.rnd	6.00	8.10	6.00	1.78
Rand_20_58_51_6.rnd	4.00	5.32	4.00	0.88
Rand_20_58_51_12.rnd	2.00	0.39	2.00	0.65
Rand_20_58_79_3.rnd	5.00	0.09	5.00	0.46
Rand_20_58_79_6.rnd	3.00	4.16	3.00	0.16
Rand_20_58_79_12.rnd	2.00	0.45	2.00	2.58
Rand_20_58_127_3.rnd	6.00	0.43	6.00	0.93
Rand_20_58_127_6.rnd	3.00	2.49	3.00	2.18
Rand_20_58_127_12.rnd	2.00	0.92	2.00	1.63
Rand_20_77_7_4.rnd	4.00	2.66	4.00	0.10
Rand_20_77_7_7.rnd	3.00	0.60	3.00	1.86
Rand_20_77_7_13.rnd	2.00	5.87	2.00	1.03
Rand_20_77_29_4.rnd	4.00	2.18	4.00	0.17
Rand_20_77_29_7.rnd	3.00	0.06	3.00	0.58
Rand_20_77_29_13.rnd	2.00	0.41	2.00	1.24
Rand_20_77_51_4.rnd	4.00	3.61	4.00	0.10
Rand_20_77_51_7.rnd	3.00	0.05	3.00	0.25
Rand_20_77_51_13.rnd	2.00	0.67	2.00	6.16
Rand_20_77_79_4.rnd	4.00	5.48	4.00	0.12
Rand_20_77_79_7.rnd	3.00	0.95	3.00	0.86
Rand_20_77_79_13.rnd	2.00	0.37	2.00	0.59
Rand_20_77_127_4.rnd	4.00	4.29	4.00	0.39
Rand_20_77_127_7.rnd	3.00	4.47	3.00	0.46
Rand_20_77_127_13.rnd	2.00	0.55	2.00	0.12

Table A.11: Optimal solutions found in instances with 20 vertices.

Instance	BC^L	Time	BC^C	Time
Rand_30_74_7_4.rnd	8.00	3.12	8.00	1.66
Rand_30_74_7_7.rnd	5.00	63.23	5.00	13.20
Rand_30_74_7_13.rnd	3.00	52.86	3.00	12.43
Rand_30_74_29_4.rnd	8.00	12.92	8.00	7.11
Rand_30_74_29_7.rnd	5.00*	10400.03	5.00	26.24
Rand_30_74_29_13.rnd	3.00	0.28	3.00	6.10
Rand_30_74_51_4.rnd	9.00	32.82	9.00	67.00
Rand_30_74_51_7.rnd	5.00*	10400.02	5.00	14.65
Rand_30_74_51_13.rnd	3.00	4.59	3.00	7.59
Rand_30_74_79_4.rnd	8.00	81.05	8.00	90.92
Rand_30_74_79_7.rnd	5.00	3773.28	5.00	6.30
Rand_30_74_79_13.rnd	3.00	14.11	3.00	27.82
Rand_30_74_127_4.rnd	9.00	291.11	9.00	60.30
Rand_30_74_127_7.rnd	5.00	1528.66	5.00	15.53
Rand_30_74_127_13.rnd	3.00	0.52	3.00	11.67
Rand_30_117_7_4.rnd	7.00	248.57	7.00	3.29
Rand_30_117_7_7.rnd	4.00	279.14	4.00	11.31
Rand_30_117_7_14.rnd	3.00*	10400.06	3.00	8.49
Rand_30_117_29_4.rnd	6.00	255.52	6.00	2.36
Rand_30_117_29_7.rnd	5.00*	10400.06	5.00	16.70
Rand_30_117_29_14.rnd	3.00*	10400.01	2.00	0.72
Rand_30_117_51_4.rnd	8.00	126.57	8.00	21.45
Rand_30_117_51_7.rnd	4.00	5298.33	4.00	43.48
Rand_30_117_51_14.rnd	3.00*	10400.06	2.00	0.85
Rand_30_117_79_4.rnd	6.00	174.81	6.00	3.86
Rand_30_117_79_7.rnd	4.00	16.97	4.00	31.46
Rand_30_117_79_14.rnd	3.00*	10400.05	3.00	21.89
Rand_30_117_127_4.rnd	7.00*	10400.01	7.00	580.04
Rand_30_117_127_7.rnd	4.00	23.00	4.00	23.11
Rand_30_117_127_14.rnd	3.00*	10400.03	3.00	24.71
Rand_30_161_7_4.rnd	7.00	1119.18	7.00	4.93
Rand_30_161_7_8.rnd	4.00	9.86	4.00	43.28
Rand_30_161_7_15.rnd	3.00*	10400.05	2.00	87.86
Rand_30_161_29_4.rnd	6.00	956.29	6.00	31.71
Rand_30_161_29_8.rnd	4.00	153.39	4.00	95.53
Rand_30_161_29_15.rnd	2.00	2946.49	2.00	39.15
Rand_30_161_51_4.rnd	6.00	473.58	6.00	0.49
Rand_30_161_51_8.rnd	4.00	2.25	4.00	91.90
Rand_30_161_51_15.rnd	2.00	2192.60	2.00	90.01
Rand_30_161_79_4.rnd	6.00	174.21	6.00	4.17
Rand_30_161_79_8.rnd	4.00	180.58	4.00	98.78
Rand_30_161_79_15.rnd	2.00	1244.13	2.00	1.27
Rand_30_161_127_4.rnd	7.00	3466.92	7.00	27.70
Rand_30_161_127_8.rnd	4.00	36.77	4.00	132.80
Rand_30_161_127_15.rnd	2.00	1711.88	2.00	207.25

Table A.12: Optimal solutions found in instances with 30 vertices.

Instance	BC^L	Time	BC^C	Time
Rand_40_118_7_4.rnd	11.00	2963.36	11.00	259.61
Rand_40_118_7_7.rnd	6.00*	10400.03	6.00	209.20
Rand_40_118_7_14.rnd	4.00*	10400.04	3.00	243.59
Rand_40_118_29_4.rnd	11.00	187.39	11.00	149.95
Rand_40_118_29_7.rnd	6.00*	10400.02	6.00	127.87
Rand_40_118_29_14.rnd	4.00*	10400.03	3.00	75.73
Rand_40_118_51_4.rnd	9.00	4608.52	9.00	5083.09
Rand_40_118_51_7.rnd	6.00*	10400.04	6.00	203.04
Rand_40_118_51_14.rnd	4.00*	10400.06	3.00	47.88
Rand_40_118_79_4.rnd	10.00*	10400.06	10.00	2597.89
Rand_40_118_79_7.rnd	7.00*	10400.01	N/A	10400.00
Rand_40_118_79_14.rnd	4.00*	10400.04	4.00	7562.52
Rand_40_118_127_4.rnd	10.00	631.80	10.00	208.73
Rand_40_118_127_7.rnd	7.00*	10400.11	7.00	1164.30
Rand_40_118_127_14.rnd	4.00*	10400.02	3.00	60.01
Rand_40_196_7_4.rnd	8.00	1274.35	8.00	58.22
Rand_40_196_7_8.rnd	5.00	641.86	5.00	598.49
Rand_40_196_7_16.rnd	3.00	23.28	3.00	686.38
Rand_40_196_29_4.rnd	9.00*	10400.04	9.00	13.19
Rand_40_196_29_8.rnd	6.00*	10400.04	5.00	389.87
Rand_40_196_29_16.rnd	4.00*	10400.04	3.00	561.56
Rand_40_196_51_4.rnd	9.00*	10400.35	9.00	14.65
Rand_40_196_51_8.rnd	6.00*	10400.04	5.00	366.33
Rand_40_196_51_16.rnd	3.00	4793.89	3.00	1046.02
Rand_40_196_79_4.rnd	9.00*	10400.04	9.00	1913.57
Rand_40_196_79_8.rnd	5.00	24.78	5.00	608.26
Rand_40_196_79_16.rnd	3.00	1057.04	4.00	677.61
Rand_40_196_127_4.rnd	9.00*	10400.03	9.00	32.03
Rand_40_196_127_8.rnd	6.00*	10400.03	5.00	385.29
Rand_40_196_127_16.rnd	3.00	61.06	3.00	2959.15
Rand_40_274_7_5.rnd	7.00	512.28	7.00	12.62
Rand_40_274_7_9.rnd	5.00*	10400.16	4.00	6.47
Rand_40_274_7_17.rnd	3.00	360.77	3.00	5193.12
Rand_40_274_29_5.rnd	8.00*	10400.11	7.00	567.20
Rand_40_274_29_9.rnd	5.00*	10400.15	4.00	22.65
Rand_40_274_29_17.rnd	4.00*	10400.04	3.00	5172.75
Rand_40_274_51_5.rnd	7.00	263.57	7.00	540.13
Rand_40_274_51_9.rnd	5.00*	10400.05	4.00	7.46
Rand_40_274_51_17.rnd	3.00	22.60	3.00	4894.52
Rand_40_274_79_5.rnd	8.00*	10400.08	7.00	430.36
Rand_40_274_79_9.rnd	5.00*	10400.16	4.00	238.74
Rand_40_274_79_17.rnd	3.00	3093.67	3.00	9147.26
Rand_40_274_127_5.rnd	7.00	348.37	7.00	18.72
Rand_40_274_127_9.rnd	4.00	3056.09	4.00	18.72
Rand_40_274_127_17.rnd	4.00*	10400.12	3.00	6.96

Table A.13: Optimal solutions found in instances with 40 vertices.

Instance	BC^L	Time	BC^C	Time
Rand_50_173_7_4.rnd	15.00*	10400.31	15.00	8663.11
Rand_50_173_7_8.rnd	9.00*	10400.06	10.00*	10400.00
Rand_50_173_7_15.rnd	6.00*	10400.03	4.00	3265.9
Rand_50_173_29_4.rnd	11.00*	10400.05	11.00*	10400.00
Rand_50_173_29_8.rnd	8.00*	10400.05	8.00*	10400.00
Rand_50_173_29_15.rnd	6.00*	10400.08	7.00*	10400.00
Rand_50_173_51_4.rnd	12.00*	10400.01	12.00	5513.6
Rand_50_173_51_8.rnd	8.00*	10400.00	8.00*	10400.00
Rand_50_173_51_15.rnd	5.00*	10400.08	4.00*	10400.00
Rand_50_173_79_4.rnd	12.00	165.03	12.00	845.15
Rand_50_173_79_8.rnd	7.00*	10400.03	7.00	1177.98
Rand_50_173_79_15.rnd	5.00*	10400.03	4.00	2827.76
Rand_50_173_127_4.rnd	12.00*	10400.01	13.00*	10400.00
Rand_50_173_127_8.rnd	10.00*	10400.01	13.00*	10400.00
Rand_50_173_127_15.rnd	5.00*	10400.06	N/A	N/A
Rand_50_295_7_5.rnd	11.00*	10400.13	9.00	1189.44
Rand_50_295_7_9.rnd	7.00*	10400.14	5.00	98.03
Rand_50_295_7_17.rnd	4.00*	10400.19	3.00	2126.48
Rand_50_295_29_5.rnd	10.00*	10400.18	9.00	1178.50
Rand_50_295_29_9.rnd	6.00*	10400.06	5.00	24.20
Rand_50_295_29_17.rnd	4.00*	10400.09	3.00	2602.04
Rand_50_295_51_5.rnd	10.00*	10400.08	9.00	1452.15
Rand_50_295_51_9.rnd	6.00*	10400.11	5.00	1987.22
Rand_50_295_51_17.rnd	4.00*	10400.07	3.00	5556.96
Rand_50_295_79_5.rnd	9.00	130.16	9.00	2726.83
Rand_50_295_79_9.rnd	6.00*	10400.17	5.00	2717.88
Rand_50_295_79_17.rnd	4.00*	10400.15	3.00	4717.74
Rand_50_295_127_5.rnd	9.00	125.17	9.00	1579.32
Rand_50_295_127_9.rnd	7.00*	10400.09	6.00*	10400.00
Rand_50_295_127_17.rnd	5.00*	10400.11	3.00	1791.5
Rand_50_418_7_5.rnd	9.00	731.22	9.00	8775.36
Rand_50_418_7_9.rnd	7.00*	10400.39	5.00	27.66
Rand_50_418_7_18.rnd	4.00*	10400.19	N/A	N/A
Rand_50_418_29_5.rnd	10.00*	10400.38	9.00	3239.52
Rand_50_418_29_9.rnd	6.00*	10400.45	5.00	27.20
Rand_50_418_29_18.rnd	4.00*	10400.24	3.00	5324.25
Rand_50_418_51_5.rnd	9.00	283.17	9.00	5574.19
Rand_50_418_51_9.rnd	7.00*	10400.33	5.00	28.46
Rand_50_418_51_18.rnd	4.00*	10400.14	N/A	N/A
Rand_50_418_79_5.rnd	9.00	842.20	9.00	10400.00
Rand_50_418_79_9.rnd	6.00*	10400.17	N/A	N/A
Rand_50_418_79_18.rnd	5.00*	10400.17	N/A	N/A
Rand_50_418_127_5.rnd	10.00*	10400.58	N/A	N/A
Rand_50_418_127_9.rnd	6.00*	10400.31	5.00	329.41
Rand_50_418_127_18.rnd	4.00*	10400.09	N/A	N/A

Table A.14: Optimal solutions found in instances with 50 vertices.

Instance	BC^L	Time	BC^C	Time
Rand_100_595_7_5.rnd	22.00*	10400.26	N/A	N/A
Rand_100_595_7_10.rnd	14.00*	10400.10	N/A	N/A
Rand_100_595_7_19.rnd	9.00*	10400.04	N/A	N/A
Rand_100_595_29_5.rnd	24.00*	10400.27	N/A	N/A
Rand_100_595_29_10.rnd	13.00*	10400.02	N/A	N/A
Rand_100_595_29_19.rnd	10.00*	10400.06	N/A	N/A
Rand_100_595_51_5.rnd	22.00*	10400.01	N/A	N/A
Rand_100_595_51_10.rnd	13.00*	10400.03	N/A	N/A
Rand_100_595_51_19.rnd	9.00*	10400.27	N/A	N/A
Rand_100_595_79_5.rnd	23.00*	10400.22	N/A	N/A
Rand_100_595_79_10.rnd	13.00*	10400.09	N/A	N/A
Rand_100_595_79_19.rnd	9.00*	10400.32	N/A	N/A
Rand_100_595_127_5.rnd	22.00*	10400.05	N/A	N/A
Rand_100_595_127_10.rnd	13.00*	10400.02	N/A	N/A
Rand_100_595_127_19.rnd	9.00*	10400.30	N/A	N/A
Rand_100_1090_7_6.rnd	19.00*	10400.83	N/A	N/A
Rand_100_1090_7_11.rnd	13.00*	10400.70	N/A	N/A
Rand_100_1090_7_21.rnd	11.00*	10400.84	N/A	N/A
Rand_100_1090_29_6.rnd	18.00*	10400.97	N/A	N/A
Rand_100_1090_29_11.rnd	13.00*	10400.57	N/A	N/A
Rand_100_1090_29_21.rnd	11.00*	10401.19	N/A	N/A
Rand_100_1090_51_6.rnd	19.00*	10401.20	N/A	N/A
Rand_100_1090_51_11.rnd	18.00*	10400.88	N/A	N/A
Rand_100_1090_51_21.rnd	9.00*	10400.27	N/A	N/A
Rand_100_1090_79_6.rnd	18.00*	10400.85	N/A	N/A
Rand_100_1090_79_11.rnd	17.00*	10401.05	N/A	N/A
Rand_100_1090_79_21.rnd	9.00*	10400.18	N/A	N/A
Rand_100_1090_127_6.rnd	21.00*	10400.26	N/A	N/A
Rand_100_1090_127_11.rnd	13.00*	10400.37	N/A	N/A
Rand_100_1090_127_21.rnd	9.00*	10400.19	N/A	N/A
Rand_100_1585_7_6.rnd	23.00*	10400.30	N/A	N/A
Rand_100_1585_7_11.rnd	15.00*	10401.69	N/A	N/A
Rand_100_1585_7_22.rnd	N/A	10400.47	N/A	N/A
Rand_100_1585_29_6.rnd	26.00*	10400.40	N/A	N/A
Rand_100_1585_29_11.rnd	16.00*	10400.41	N/A	N/A
Rand_100_1585_29_22.rnd	N/A	10423.10	N/A	N/A
Rand_100_1585_51_6.rnd	19.00*	10400.47	N/A	N/A
Rand_100_1585_51_11.rnd	16.00*	10400.30	N/A	N/A
Rand_100_1585_51_22.rnd	N/A	10400.45	N/A	N/A
Rand_100_1585_79_6.rnd	23.00*	10400.51	N/A	N/A
Rand_100_1585_79_11.rnd	19.00*	10400.42	N/A	N/A
Rand_100_1585_79_22.rnd	11.00*	10400.53	N/A	N/A
Rand_100_1585_127_6.rnd	17.00*	10401.82	N/A	N/A
Rand_100_1585_127_11.rnd	N/A	10400.47	N/A	N/A
Rand_100_1585_127_22.rnd	11.00*	10400.42	N/A	N/A

Table A.15: Optimal solutions found in instances with 100 vertices.

A.4 Detailed Branch-and-Cut Costs

In this section, we detail the costs of every operation in the branch-and-cut algorithm, detailing the results shown in Section 4.6. Tables A.16, A.17, A.18, A.19 and A.20 show the results for instances with 20, 30, 40, 50 and 100 vertices, respectively. The first column of every table shows the instance's name. The second column shows the total time required to solve that instance, up to the time limit of 10400 seconds. The third column shows the percentage of the total time spent on GSEC separation. The fourth and fifth column show the percentages of the total time spent on exact and heuristic CSEC separation, respectively. Finally, the last column shows the percentage of the total time spent on reoptimization, that is, on solving the linear programming subproblems.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_20_39_7_3.rnd	0.09	23.83%	0.00%	0.00%	76.17%
Rand_20_39_7_6.rnd	0.98	6.42%	0.00%	0.73%	92.85%
Rand_20_39_7_11.rnd	0.12	26.86%	0.00%	0.65%	72.49%
Rand_20_39_29_3.rnd	0.07	0.00%	0.00%	1.51%	98.49%
Rand_20_39_29_6.rnd	1.77	8.81%	0.00%	0.44%	90.76%
Rand_20_39_29_11.rnd	270.82	1.50%	3.90%	0.17%	94.43%
Rand_20_39_51_3.rnd	0.05	0.00%	0.00%	1.25%	98.75%
Rand_20_39_51_6.rnd	0.96	11.87%	0.00%	0.72%	87.42%
Rand_20_39_51_11.rnd	0.15	27.82%	0.00%	0.70%	71.48%
Rand_20_39_79_3.rnd	0.04	0.00%	0.00%	1.13%	98.87%
Rand_20_39_79_6.rnd	6.99	5.25%	4.31%	1.08%	89.36%
Rand_20_39_79_11.rnd	1.29	9.70%	6.65%	0.72%	82.94%
Rand_20_39_127_3.rnd	0.08	24.83%	0.00%	0.34%	74.84%
Rand_20_39_127_6.rnd	5.30	21.51%	2.89%	0.65%	74.94%
Rand_20_39_127_11.rnd	6.02	8.63%	38.35%	0.42%	52.60%
Rand_20_58_7_3.rnd	0.61	2.31%	6.57%	0.29%	90.84%
Rand_20_58_7_6.rnd	3.60	6.33%	5.02%	0.77%	87.88%
Rand_20_58_7_12.rnd	0.07	0.00%	0.00%	0.90%	99.10%
Rand_20_58_29_3.rnd	0.16	20.70%	0.00%	0.77%	78.54%
Rand_20_58_29_6.rnd	4.01	5.07%	5.08%	0.89%	88.96%
Rand_20_58_29_12.rnd	0.19	23.19%	0.00%	0.41%	76.41%
Rand_20_58_51_3.rnd	8.10	2.59%	0.56%	0.73%	96.12%
Rand_20_58_51_6.rnd	5.32	1.61%	0.00%	0.65%	97.74%
Rand_20_58_51_12.rnd	0.39	11.54%	21.79%	0.84%	65.83%
Rand_20_58_79_3.rnd	0.09	0.00%	0.00%	0.79%	99.21%
Rand_20_58_79_6.rnd	4.16	5.02%	0.00%	0.61%	94.37%
Rand_20_58_79_12.rnd	0.45	0.00%	0.00%	0.66%	99.34%
Rand_20_58_127_3.rnd	0.43	7.32%	0.00%	0.53%	92.15%
Rand_20_58_127_6.rnd	2.49	3.52%	2.11%	0.96%	93.41%
Rand_20_58_127_12.rnd	0.92	21.76%	0.00%	0.97%	77.27%
Rand_20_77_7_4.rnd	2.66	2.98%	0.00%	0.69%	96.34%
Rand_20_77_7_7.rnd	0.60	9.72%	0.00%	1.37%	88.91%
Rand_20_77_7_13.rnd	5.87	5.63%	0.00%	0.53%	93.85%
Rand_20_77_29_4.rnd	2.18	0.00%	0.00%	0.78%	99.22%
Rand_20_77_29_7.rnd	0.06	0.00%	0.00%	1.63%	98.37%
Rand_20_77_29_13.rnd	0.41	22.50%	0.00%	2.60%	74.91%
Rand_20_77_51_4.rnd	3.61	2.95%	0.00%	0.58%	96.47%
Rand_20_77_51_7.rnd	0.05	0.00%	0.00%	1.99%	98.01%
Rand_20_77_51_13.rnd	0.67	6.12%	0.00%	0.56%	93.32%
Rand_20_77_79_4.rnd	5.48	1.59%	0.00%	0.75%	97.67%
Rand_20_77_79_7.rnd	0.95	0.00%	0.00%	0.79%	99.21%
Rand_20_77_79_13.rnd	0.37	10.37%	0.00%	0.39%	89.24%
Rand_20_77_127_4.rnd	4.29	0.00%	0.00%	0.98%	99.02%
Rand_20_77_127_7.rnd	4.47	0.00%	0.00%	0.70%	99.30%
Rand_20_77_127_13.rnd	0.55	11.05%	0.00%	0.94%	88.01%

Table A.16: Separation costs for instances with 20 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_30_74_7_4.rnd	3.12	37.24%	0.00%	0.84%	61.91%
Rand_30_74_7_7.rnd	63.23	0.40%	0.11%	0.92%	98.57%
Rand_30_74_7_13.rnd	52.86	0.38%	0.00%	0.29%	99.33%
Rand_30_74_29_4.rnd	12.92	32.27%	0.00%	0.59%	67.14%
Rand_30_74_29_7.rnd	10400.03	0.02%	0.01%	0.05%	99.93%
Rand_30_74_29_13.rnd	0.28	0.00%	0.00%	1.00%	99.00%
Rand_30_74_51_4.rnd	32.82	20.22%	0.00%	0.59%	79.19%
Rand_30_74_51_7.rnd	10400.02	0.01%	0.01%	0.04%	99.94%
Rand_30_74_51_13.rnd	4.59	0.53%	2.44%	0.88%	96.15%
Rand_30_74_79_4.rnd	81.05	1.08%	0.53%	1.35%	97.04%
Rand_30_74_79_7.rnd	3773.28	0.03%	0.01%	0.17%	99.79%
Rand_30_74_79_13.rnd	14.11	0.44%	0.00%	0.55%	99.01%
Rand_30_74_127_4.rnd	291.11	15.94%	1.66%	0.22%	82.17%
Rand_30_74_127_7.rnd	1528.66	0.06%	0.04%	0.26%	99.65%
Rand_30_74_127_13.rnd	0.52	9.47%	0.00%	0.88%	89.65%
Rand_30_117_7_4.rnd	248.57	0.29%	0.14%	0.76%	98.81%
Rand_30_117_7_7.rnd	279.14	0.10%	0.00%	0.13%	99.77%
Rand_30_117_7_14.rnd	10400.06	0.01%	0.00%	0.04%	99.96%
Rand_30_117_29_4.rnd	255.52	0.18%	0.11%	0.23%	99.48%
Rand_30_117_29_7.rnd	10400.06	0.02%	0.01%	0.06%	99.90%
Rand_30_117_29_14.rnd	10400.01	0.00%	0.00%	0.05%	99.95%
Rand_30_117_51_4.rnd	126.57	0.38%	0.17%	1.09%	98.36%
Rand_30_117_51_7.rnd	5298.33	0.03%	0.00%	0.04%	99.93%
Rand_30_117_51_14.rnd	10400.06	0.00%	0.00%	0.04%	99.96%
Rand_30_117_79_4.rnd	174.81	1.09%	1.44%	1.24%	96.23%
Rand_30_117_79_7.rnd	16.97	2.67%	0.00%	0.31%	97.03%
Rand_30_117_79_14.rnd	10400.05	0.00%	0.00%	0.05%	99.95%
Rand_30_117_127_4.rnd	10400.01	0.03%	0.01%	0.21%	99.75%
Rand_30_117_127_7.rnd	23.00	1.19%	0.00%	0.41%	98.40%
Rand_30_117_127_14.rnd	10400.03	0.00%	0.00%	0.04%	99.95%
Rand_30_161_7_4.rnd	1119.18	0.08%	0.03%	0.16%	99.73%
Rand_30_161_7_8.rnd	9.86	0.00%	0.00%	0.53%	99.47%
Rand_30_161_7_15.rnd	10400.05	0.02%	0.00%	0.03%	99.95%
Rand_30_161_29_4.rnd	956.29	0.00%	0.00%	0.09%	99.91%
Rand_30_161_29_8.rnd	153.39	0.37%	0.00%	0.04%	99.60%
Rand_30_161_29_15.rnd	2946.49	0.04%	0.01%	0.05%	99.90%
Rand_30_161_51_4.rnd	473.58	0.02%	0.00%	0.16%	99.82%
Rand_30_161_51_8.rnd	2.25	0.00%	0.00%	1.31%	98.69%
Rand_30_161_51_15.rnd	2192.60	0.01%	0.00%	0.06%	99.93%
Rand_30_161_79_4.rnd	174.21	0.00%	0.00%	0.25%	99.75%
Rand_30_161_79_8.rnd	180.58	0.13%	0.00%	0.25%	99.62%
Rand_30_161_79_15.rnd	1244.13	0.03%	0.00%	0.09%	99.87%
Rand_30_161_127_4.rnd	3466.92	0.06%	0.01%	0.30%	99.62%
Rand_30_161_127_8.rnd	36.77	6.39%	0.00%	0.46%	93.14%
Rand_30_161_127_15.rnd	1711.88	0.02%	0.00%	0.09%	99.89%

Table A.17: Separation costs for instances with 30 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_40_118_7_4.rnd	2963.36	7.22%	0.00%	0.18%	92.61%
Rand_40_118_7_7.rnd	10400.03	0.01%	0.00%	0.03%	99.97%
Rand_40_118_7_14.rnd	10400.04	0.01%	0.00%	0.04%	99.96%
Rand_40_118_29_4.rnd	187.39	15.37%	0.00%	0.55%	84.09%
Rand_40_118_29_7.rnd	10400.02	0.01%	0.00%	0.04%	99.95%
Rand_40_118_29_14.rnd	10400.03	0.01%	0.00%	0.03%	99.96%
Rand_40_118_51_4.rnd	4608.52	0.05%	0.01%	0.31%	99.62%
Rand_40_118_51_7.rnd	10400.04	0.01%	0.00%	0.08%	99.91%
Rand_40_118_51_14.rnd	10400.06	0.00%	0.00%	0.04%	99.95%
Rand_40_118_79_4.rnd	10400.06	0.14%	0.03%	0.25%	99.58%
Rand_40_118_79_7.rnd	10400.01	0.01%	0.00%	0.14%	99.85%
Rand_40_118_79_14.rnd	10400.04	0.00%	0.00%	0.06%	99.94%
Rand_40_118_127_4.rnd	631.80	11.87%	0.00%	0.38%	87.75%
Rand_40_118_127_7.rnd	10400.11	0.01%	0.00%	0.18%	99.81%
Rand_40_118_127_14.rnd	10400.02	0.00%	0.00%	0.05%	99.95%
Rand_40_196_7_4.rnd	1274.35	0.00%	0.00%	0.27%	99.73%
Rand_40_196_7_8.rnd	641.86	0.10%	0.00%	0.23%	99.67%
Rand_40_196_7_16.rnd	23.28	0.00%	0.00%	1.35%	98.65%
Rand_40_196_29_4.rnd	10400.04	0.01%	0.00%	0.03%	99.96%
Rand_40_196_29_8.rnd	10400.04	0.00%	0.00%	0.04%	99.96%
Rand_40_196_29_16.rnd	10400.04	0.00%	0.00%	0.03%	99.96%
Rand_40_196_51_4.rnd	10400.35	0.02%	0.00%	0.28%	99.70%
Rand_40_196_51_8.rnd	10400.04	0.00%	0.00%	0.03%	99.96%
Rand_40_196_51_16.rnd	4793.89	0.01%	0.00%	0.04%	99.95%
Rand_40_196_79_4.rnd	10400.04	0.02%	0.00%	0.14%	99.85%
Rand_40_196_79_8.rnd	24.78	0.90%	0.00%	1.26%	97.84%
Rand_40_196_79_16.rnd	1057.04	0.15%	0.00%	0.15%	99.71%
Rand_40_196_127_4.rnd	10400.03	0.00%	0.00%	0.06%	99.93%
Rand_40_196_127_8.rnd	10400.03	0.00%	0.00%	0.06%	99.94%
Rand_40_196_127_16.rnd	61.06	2.17%	0.00%	1.02%	96.81%
Rand_40_274_7_5.rnd	512.28	0.21%	0.00%	0.64%	99.15%
Rand_40_274_7_9.rnd	10400.16	0.00%	0.00%	0.07%	99.93%
Rand_40_274_7_17.rnd	360.77	0.30%	0.00%	0.81%	98.89%
Rand_40_274_29_5.rnd	10400.11	0.01%	0.00%	0.04%	99.95%
Rand_40_274_29_9.rnd	10400.15	0.01%	0.00%	0.06%	99.93%
Rand_40_274_29_17.rnd	10400.04	0.17%	0.00%	0.06%	99.77%
Rand_40_274_51_5.rnd	263.57	0.39%	0.00%	0.54%	99.07%
Rand_40_274_51_9.rnd	10400.05	0.00%	0.00%	0.07%	99.94%
Rand_40_274_51_17.rnd	22.60	40.22%	0.00%	0.36%	59.43%
Rand_40_274_79_5.rnd	10400.08	0.02%	0.01%	0.08%	99.90%
Rand_40_274_79_9.rnd	10400.16	0.06%	0.00%	0.07%	99.87%
Rand_40_274_79_17.rnd	3093.67	0.25%	0.00%	0.05%	99.70%
Rand_40_274_127_5.rnd	348.37	0.16%	0.00%	0.88%	98.96%
Rand_40_274_127_9.rnd	3056.09	0.01%	0.01%	0.36%	99.62%
Rand_40_274_127_17.rnd	10400.12	0.03%	0.00%	0.07%	99.90%

Table A.18: Separation costs for instances with 40 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_50_173_7_4.rnd	3606.45	16.87%	0.00%	0.26%	82.87%
Rand_50_173_7_8.rnd	10400.08	7.51%	0.00%	0.04%	92.45%
Rand_50_173_7_15.rnd	1889.25	0.33%	0.00%	0.13%	99.53%
Rand_50_173_29_4.rnd	4917.04	0.15%	0.00%	0.08%	99.77%
Rand_50_173_29_8.rnd	10400.05	0.13%	0.00%	0.07%	99.80%
Rand_50_173_29_15.rnd	10400.06	0.13%	0.00%	0.05%	99.81%
Rand_50_173_51_4.rnd	10400.40	39.77%	0.00%	0.46%	59.77%
Rand_50_173_51_8.rnd	10400.05	0.10%	0.00%	0.05%	99.85%
Rand_50_173_51_15.rnd	10400.07	0.09%	0.00%	0.08%	99.83%
Rand_50_173_79_4.rnd	375.73	76.53%	0.00%	0.42%	23.06%
Rand_50_173_79_8.rnd	10400.02	0.11%	0.00%	0.05%	99.83%
Rand_50_173_79_15.rnd	10400.04	0.09%	0.00%	0.05%	99.85%
Rand_50_173_127_4.rnd	3896.37	33.51%	0.00%	0.34%	66.16%
Rand_50_173_127_8.rnd	10401.62	59.20%	0.00%	0.26%	40.54%
Rand_50_173_127_15.rnd	10400.06	0.10%	0.00%	0.07%	99.83%
Rand_50_295_7_5.rnd	10400.27	0.27%	0.00%	0.23%	99.50%
Rand_50_295_7_9.rnd	10400.32	0.31%	0.00%	0.19%	99.50%
Rand_50_295_7_17.rnd	10400.21	0.83%	0.01%	0.17%	99.00%
Rand_50_295_29_5.rnd	1382.01	0.75%	0.00%	0.46%	98.79%
Rand_50_295_29_9.rnd	10400.14	0.29%	0.00%	0.12%	99.59%
Rand_50_295_29_17.rnd	10400.27	1.09%	0.00%	0.16%	98.74%
Rand_50_295_51_5.rnd	10400.27	0.25%	0.00%	0.15%	99.60%
Rand_50_295_51_9.rnd	10400.19	0.32%	0.00%	0.18%	99.50%
Rand_50_295_51_17.rnd	10400.14	1.93%	0.01%	0.12%	97.94%
Rand_50_295_79_5.rnd	681.07	0.33%	0.00%	0.62%	99.05%
Rand_50_295_79_9.rnd	10400.24	0.22%	0.01%	0.22%	99.55%
Rand_50_295_79_17.rnd	10400.15	1.02%	0.00%	0.14%	98.83%
Rand_50_295_127_5.rnd	233.67	2.80%	0.00%	0.42%	96.78%
Rand_50_295_127_9.rnd	10400.35	0.25%	0.00%	0.19%	99.56%
Rand_50_295_127_17.rnd	10400.12	0.97%	0.00%	0.13%	98.90%
Rand_50_418_7_5.rnd	312.04	0.44%	0.00%	0.47%	99.09%
Rand_50_418_7_9.rnd	10400.03	0.62%	0.00%	0.36%	99.02%
Rand_50_418_7_18.rnd	10400.28	4.67%	0.00%	0.30%	95.03%
Rand_50_418_29_5.rnd	1992.24	0.10%	0.00%	0.24%	99.66%
Rand_50_418_29_9.rnd	10400.36	0.48%	0.00%	0.22%	99.29%
Rand_50_418_29_18.rnd	10400.43	7.79%	0.00%	0.27%	91.94%
Rand_50_418_51_5.rnd	2499.80	0.37%	0.02%	0.33%	99.28%
Rand_50_418_51_9.rnd	10400.23	0.61%	0.00%	0.31%	99.08%
Rand_50_418_51_18.rnd	10400.28	4.39%	0.00%	0.37%	95.24%
Rand_50_418_79_5.rnd	985.80	0.16%	0.00%	0.18%	99.66%
Rand_50_418_79_9.rnd	10400.36	0.37%	0.01%	0.53%	99.09%
Rand_50_418_79_18.rnd	10400.51	4.28%	0.00%	0.36%	95.35%
Rand_50_418_127_5.rnd	10400.05	0.29%	0.00%	0.35%	99.35%
Rand_50_418_127_9.rnd	10400.26	0.55%	0.00%	0.31%	99.14%
Rand_50_418_127_18.rnd	10400.23	4.84%	0.01%	0.16%	94.99%

Table A.19: Separation costs for instances with 50 vertices.

Instance	Time	GSEC Cost	CSEC Exact Cost	CSEC Heuristic Cost	LP Cost
Rand_100_595_7_5.rnd	10400.39	0.56%	0.00%	0.27%	99.17%
Rand_100_595_7_10.rnd	10400.10	9.76%	0.00%	0.00%	90.24%
Rand_100_595_7_19.rnd	10400.06	3.29%	0.00%	0.25%	96.47%
Rand_100_595_29_5.rnd	10400.08	1.52%	0.00%	0.18%	98.30%
Rand_100_595_29_10.rnd	10400.02	18.00%	0.00%	0.00%	82.00%
Rand_100_595_29_19.rnd	10400.17	4.77%	0.00%	0.06%	95.18%
Rand_100_595_51_5.rnd	10400.61	77.55%	0.00%	0.37%	22.08%
Rand_100_595_51_10.rnd	10400.03	11.43%	0.00%	0.00%	88.57%
Rand_100_595_51_19.rnd	10400.24	0.32%	0.00%	0.23%	99.45%
Rand_100_595_79_5.rnd	10404.65	0.18%	0.00%	0.19%	99.63%
Rand_100_595_79_10.rnd	10400.09	15.46%	0.00%	0.00%	84.54%
Rand_100_595_79_19.rnd	10400.27	5.46%	0.00%	0.17%	94.37%
Rand_100_595_127_5.rnd	10400.31	0.63%	0.00%	0.24%	99.13%
Rand_100_595_127_10.rnd	10400.02	13.56%	0.00%	0.00%	86.44%
Rand_100_595_127_19.rnd	10400.33	6.15%	0.00%	0.13%	93.72%
Rand_100_1090_7_6.rnd	10400.97	0.00%	0.00%	0.29%	99.71%
Rand_100_1090_7_11.rnd	10400.26	0.00%	0.00%	0.11%	99.89%
Rand_100_1090_7_21.rnd	10400.86	13.78%	0.00%	0.04%	86.18%
Rand_100_1090_29_6.rnd	10400.22	0.87%	0.00%	0.30%	98.83%
Rand_100_1090_29_11.rnd	10400.16	3.42%	0.00%	0.10%	96.48%
Rand_100_1090_29_21.rnd	10405.19	10.74%	0.00%	0.27%	88.99%
Rand_100_1090_51_6.rnd	10401.20	0.00%	0.00%	0.06%	99.94%
Rand_100_1090_51_11.rnd	10401.11	0.00%	0.00%	0.14%	99.86%
Rand_100_1090_51_21.rnd	10400.29	0.00%	0.00%	0.01%	99.99%
Rand_100_1090_79_6.rnd	10400.17	0.00%	0.00%	0.37%	99.63%
Rand_100_1090_79_11.rnd	10401.07	1.86%	0.00%	0.05%	98.09%
Rand_100_1090_79_21.rnd	10400.21	0.00%	0.00%	0.13%	99.87%
Rand_100_1090_127_6.rnd	10401.08	0.28%	0.00%	0.17%	99.55%
Rand_100_1090_127_11.rnd	10400.65	0.82%	0.00%	0.38%	98.80%
Rand_100_1090_127_21.rnd	10401.46	0.00%	0.00%	0.12%	99.88%
Rand_100_1585_7_6.rnd	10402.61	0.00%	0.00%	0.15%	99.85%
Rand_100_1585_7_11.rnd	10402.02	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_7_22.rnd	10400.58	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_29_6.rnd	10400.53	0.00%	0.00%	0.18%	99.82%
Rand_100_1585_29_11.rnd	10400.43	3.21%	0.00%	0.14%	96.64%
Rand_100_1585_29_22.rnd	10459.35	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_51_6.rnd	10400.48	0.00%	0.00%	0.32%	99.68%
Rand_100_1585_51_11.rnd	10400.29	0.00%	0.00%	0.35%	99.65%
Rand_100_1585_51_22.rnd	10400.51	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_79_6.rnd	10400.46	0.00%	0.00%	0.40%	99.60%
Rand_100_1585_79_11.rnd	10400.51	0.00%	0.00%	0.23%	99.77%
Rand_100_1585_79_22.rnd	10401.55	1.62%	0.00%	0.00%	98.38%
Rand_100_1585_127_6.rnd	10400.44	1.42%	0.00%	0.12%	98.46%
Rand_100_1585_127_11.rnd	10400.53	0.00%	0.00%	0.00%	100.00%
Rand_100_1585_127_22.rnd	10400.43	0.00%	0.00%	0.19%	99.81%

Table A.20: Separation costs for instances with 100 vertices.