
# THE MINIMUM DELAY UPGRADING MINIMUM SPANNING TREE PROBLEM

Bruno Schelk Bernardo dos Santos

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Laura Silvia Bahiense da Silva Leite
Luidi Gelabert Simonetti

Rio de Janeiro
Fevereiro de 2025

THE MINIMUM DELAY UPGRADING MINIMUM SPANNING TREE PROBLEM

Bruno Schelk Bernardo dos Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Laura Silvia Bahiense da Silva Leite
Luidi Gelabert Simonetti

Aprovada por: Prof. Laura Silvia Bahiense da Silva Leite
Prof. Luidi Gelabert Simonetti
Prof. Alexandre Salles da Cunha
Prof. Yuri Abitbol de Menezes Frota

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2025

dos Santos, Bruno Schelk Bernardo

The Minimum Delay Upgrading Minimum Spanning Tree Problem/Bruno Schelk Bernardo dos Santos. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIV, 41 p.: il.; 29, 7cm.

Orientadores: Laura Silvia Bahiense da Silva Leite

Luidi Gelabert Simonetti

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 38 – 41.

1. Minimum Spanning Tree Problem. 2. Node-based upgrade. 3. Integer Programming. 4. Branch-and-Cut Algorithm. I. , Laura Silvia Bahiense da Silva Leite *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*"Ninguém vai te bater tão duro quanto a vida. Mas não se trata de você bater forte, e sim do quanto você aguenta apanhar e seguir em frente. O quanto você consegue aguentar e continuar tentando. É assim que se vence!"*

*Rocky Balboa*

*Dedico este trabalho aos meus queridos e eternos professores.*

# Agradecimentos

Gostaria de expressar minha gratidão aqueles que, direta ou indiretamente, contribuíram para a conclusão deste trabalho. Como seria impossível listar cada um, deixo, desde já, meus sinceros agradecimentos a todos.

Inicialmente, agradeço aos meus orientadores, os professores Laura Bahiense (primeiro as damas) e Luidi Simonetti. A qualidade técnica da professora Laura é inquestionável, por isso decidi destacar aqui seu lado humano como orientadora. Muito obrigado por olhar para mim e enxergar não apenas um número, mas uma pessoa, suscetível a acertos e erros. Obrigado por estender a mão quando precisei, mas também por me dar "puxão de orelha" nos momentos em que era necessário sair da inércia. Nunca vou esquecer do dia em que me disse: "... isso eu falaria para minha filha", e quando, no momento mais difícil da minha carreira acadêmica, me aconselhou a parar, respirar fundo, recuperar as energias e voltar mais forte.

Da mesma forma, agradeço ao professor Luidi Simonetti. Obrigado por estar sempre disponível para me ajudar, seja com questões técnicas ou sobre a vida num geral. Percebo o quanto amadureci profissionalmente graças aos seus conselhos. Obrigado por cada reunião que tivemos, que, para mim, eram verdadeiros seminários, de valor inestimável. Agradeço por compartilhar comigo sua vasta experiência, suas ideias fora da curva, que até hoje me surpreendem, e também pelos momentos descontraídos, especialmente nossas conversas sobre futebol.

Aos dois, peço desculpas por não conseguir expressar em palavras o quanto sou grato. Espero um dia poder ser um professor e orientador como vocês, pois os tenho como grande fonte de inspiração. Novamente, muito obrigado por acreditarem em mim e neste projeto!

Também gostaria de citar alguns professores que marcaram minha trajetória acadêmica. O brilhante professor Welington de Oliveira, responsável por me apresentar ao mundo científico e à otimização. A inseparável dupla Michel Tcheou e Lisandro Lovisolo, que sempre acreditou no meu potencial e me motivou a continuar na vida

acadêmica, mesmo quando eu duvidava de mim mesmo. A professora Djalene Rocha, cujas aulas sempre reforçavam o verdadeiro valor do estudo.

Sou grato também aos professores Alexandre Salles da Cunha e Yuri Abitbol de Menezes Frota por aceitarem fazer parte da minha banca de defesa. É uma honra contar com a presença de ambos.

Aos colegas do LabOtim, meu muito obrigado. Sempre encontrei em vocês apoio e disposição para ajudar, sem qualquer tipo de vaidade. Trabalhar em um ambiente repleto de pessoas extremamente competentes e, ao mesmo tempo, descontraídas fez toda a diferença. Desejo sucesso a todos! Um agradecimento especial ao Cadu, que cursou praticamente todas as disciplinas de mestrado comigo. Obrigado pelo companheirismo, compreensão e ajuda, principalmente nos momentos difíceis.

Agradeço também a todos que conheci no CEPEL. Em especial, ao Luís Fernando, meu supervisor, que me apresentou pela primeira vez ao problema estudado nesta dissertação. Obrigado por todos os ensinamentos profissionais e por todo apoio. Ao professor André Diniz, que foi meu professor na UERJ e chefe de departamento no CEPEL, sou grato pelos aprendizados e oportunidades. A todos os colegas estagiários e bolsistas que tive o prazer de conhecer nas salas C26 e D26, meu muito obrigado.

Meu agradecimento também ao Léo, amigo de graduação que tive o prazer de levar para a vida. Obrigado por todas as conversas, desabafos e momentos de descontração. Obrigado por sempre me motivar, dizendo que acreditava em mim. Um grande abraço para você, meu amigo!

Ao psicólogo Mateus, que talvez tenha sido a pessoa que mais ouviu falar sobre as dificuldades e desafios encontrados durante o mestrado, deixo minha gratidão. Muito obrigado por toda a ajuda, sempre com profissionalismo e empatia.

Agradeço à CAPES pelo apoio financeiro para realização deste trabalho e ao Guty e Ricardo, do PESC, por toda a assistência na parte burocrática.

Por fim, gostaria de destacar a pessoa que esteve ao meu lado em TODOS os momentos: minha esposa, Débora Schelk. Obrigado pelo seu carinho, seu otimismo diante das adversidades e por sempre apoiar minhas escolhas profissionais. Agradeço imensamente pelo suporte na parte gráfica deste trabalho. Não poderia ter contado com uma designer melhor. Me desculpe por, em momentos de estresse, acabar descontando em você, e também pelas vezes em que não pude estar ao seu lado por causa do trabalho. Sou profundamente grato por sua compreensão e por fazer parte desta conquista junto comigo. Sua companhia torna a vida significativamente mais leve e feliz.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# THE MINIMUM DELAY UPGRADING MINIMUM SPANNING TREE PROBLEM

Bruno Schelk Bernardo dos Santos

Fevereiro/2025

Orientadores: Laura Silvia Bahiense da Silva Leite
Luidi Gelabert Simonetti

Programa: Engenharia de Sistemas e Computação

Em várias operações práticas, há um esforço contínuo para aprimorar sistemas e melhorar a eficiência. Para redes existentes, realizar melhorias geralmente é mais econômico do que construir novos componentes ou reconstruir a rede por completo. The Minimum Delay Upgrading Minimum Spanning Tree Problem (MDUMSTP) concentra-se na alocação de recursos limitados para atualizar uma rede existente e minimizar o atraso total da árvore geradora mínima após essas atualizações. Este problema é um caso específico de atualizações baseadas em nós, em que a atualização de um nó gera um custo, mas reduz os pesos de todas as arestas conectadas a ele. Neste trabalho, para resolver o MDUMSTP, propomos uma nova formulação baseada em Programação linear Inteira (IP) e a comparamos com o modelo exato encontrado na literatura. Ao incorporar a separação e adição de desigualdades válidas, aprimoramos a formulação, alcançando limites mais rígidos e melhor desempenho computacional. Além disso, introduzimos um procedimento de pré-processamento que reduz significativamente o tamanho das instâncias. Experimentos numéricos mostram que nosso algoritmo Branch-and-Cut supera o método exato existente na literatura em diversas instâncias de teste, e ainda, resolve até a otimalidade instâncias previamente não resolvidas.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

# THE MINIMUM DELAY UPGRADING MINIMUM SPANNING TREE PROBLEM

Bruno Schelk Bernardo dos Santos

February/2025

Advisors: Laura Silvia Bahiense da Silva Leite
Luidi Gelabert Simonetti

Department: Systems Engineering and Computer Science

In several practical operations, there is a continuous effort to enhance systems and improve efficiency. For existing networks, upgrading is generally more cost-effective than constructing new components or rebuilding the network entirely. The Minimum Delay Upgrading Minimum Spanning Tree Problem (MDUMSTP) focuses on allocating limited resources to upgrade an existing network and minimize the total delay of the minimum spanning tree after these upgrades. This problem is a specific case of node-based upgrades, where upgrade a node incurs a cost but reduces the weights of all its connected edges. In this work, to solve the MDUMSTP, we propose a new Integer linear Programming (IP) formulation and compared it with the exact model found in the literature. By incorporating the separation and addition of valid inequalities, we enhance the formulation, leading to tighter bounds and improved computational performance. Moreover, we introduce a preprocessing procedure that significantly reduces the size of the instances. Numerical experiments show that our Branch-and-Cut algorithm outperforms existing exact model in the literature for several test instances and, furthermore, solves to optimality previously unsolved instances.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| BCNUP | Budget Constrained Network Upgrading Problems, p. 1 |
| GSECs | Generalized Subtour-Elimination Constraints, p. 10 |
| IP | Integer linear Programming, p. 4 |
| MDUMSTP | Minimum Delay Upgrade Minimum Spanning Tree Problem, p. 3 |
| MSTP | Minimum Spanning Tree Problem, p. 2 |
| MST | Minimum Spanning Tree, p. 2 |

# Chapter 1

# Introduction

In several practical operations, there is a continuous effort to enhance systems and improve efficiency. For existing networks, upgrading is generally more cost-effective than constructing new components or rebuilding the network entirely.

In this context, the *Budget Constrained Network Upgrading Problems* (BCNUP) KRUMKE *et al.* (1998) focus on how to allocate limited resources to upgrade an existing network to maximize its efficiency.

For this type of problem, the main characteristics are:

i) **Existing network**: A previously established network, composed of nodes (which can represent cities, routers, substations, etc.) and edges (which can represent roads, network connections, transmission lines, etc.);

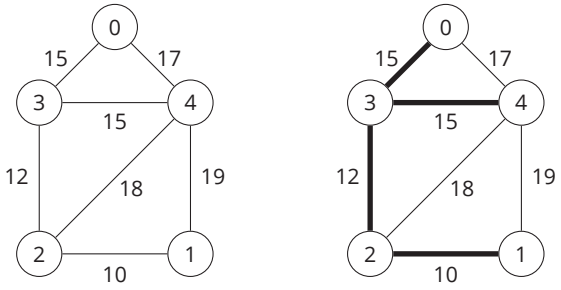
ii) **Possible upgrades**: Each network component (nodes or edges) can be upgraded. This could mean increasing capacity, reducing latency, improve security, etc.;

iii) **Upgrade costs**: Each possible upgrade has an associated cost. For example, increasing the capacity of a road requires money;

iv) **Limited budget**: There is a limited amount of resources (budget) available for upgrades;

v) **Objective**: To maximize a given network performance criterion (such as total capacity, efficiency, response time, etc.).

Let $G = (V, E)$ be a simple, connected and undirected graph, where $V$ and $E$ are the set of vertices and edges, respectively, and furthermore, for each edge $e = \{i, j\} \in E$, a weight $d_e$ is associated. A tree $T = (V_T, E_T)$, such that $V_T \subseteq V$ and $E_T \subseteq E$, is a connected and acyclic subgraph of $G$. The tree is called spanning (OBRUČA, 1968) when it contains all the vertices of the graph, that is, when $V_T = V$. Let $\mathcal{T}$ be the set of all spanning trees of $G$. A *Minimum Spanning Tree* (MST) is a spanning tree that among all those contained in $\mathcal{T}$ the sum of the edge weights is the smallest, and consequently, the *Minimum Spanning Tree Problem* (MSTP) (GRAHAM and HELL, 1985) consists in finding it. An example of a minimum spanning tree problem is illustrated in Figure 1.1.




(a) (b)

Figure 1.1: Example of a graph (a) and its minimum spanning tree (b).

The problem mentioned above (MSTP) is famous in the area of combinatorial optimization since 1926 (BORUVKA, 1926), with numerous applications in everyday life referring to various network models: telecommunication, computers, transport, among others (GRAHAM and HELL, 1985). As is known, such a problem is easy to solve since there is at least one polynomial-time algorithm capable of solving it in the worst case. Two of the most famous methods are *Kruskal* (KRUSKAL, 1956) and *Prim* (PRIM,

1957) algorithms, with computational-time complexity $O(|E|log|V|)$ and $O(|V|^2)$, respectively.

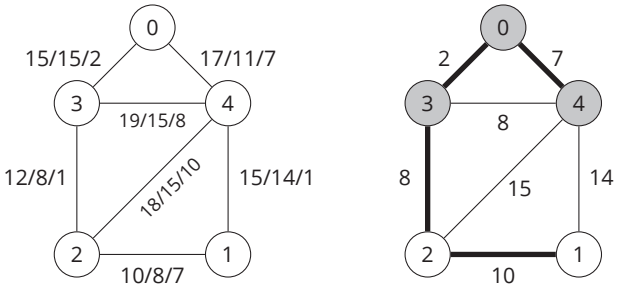
Variants of the MSTP are found in abundance in the literature. This fact happens because, even though MSTP is easy to solve, by adding a simple set of constraints can make its resolution significantly more complex. For example: the *Degree-Constrained Minimum Spanning Tree Problem* (DE ALMEIDA *et al.*, 2012), which establishes a maximum degree constraint to each vertex of the spanning tree; *Bounded Diameter Minimum Spanning Tree Problem* (HO and LEE, 1989), which imposes a maximum diameter constraint for the spanning tree; among others. Such problems belong to the NP-Hard class (VAN LEEUWEN, 1991), that is, according to the computational theory, no polynomial-time algorithm has been found to solve them in the worst case to date.

In this work, we study the *Minimum Delay Upgrade Minimum Spanning Tree Problem* (MDUMSTP), a particular case of the node-based upgrade problem. In this model, a node $i \in V$ can be upgraded at a cost $c_i \geq 0$, which reduces the weights of all edges connected to that node. We are given three integers $d_e^0 \geq d_e^1 \geq d_e^2 \geq 0$, for each edge $e \in E$, where the $d_e^l$ (the edge $e$ in level $l$) value represents the weight or delay of $e$ when exactly $l$ of its endpoints are upgraded.

Let us consider an upgrade configuration $U$, and a subset $V_U \subseteq V$, where all nodes in $V_U$ are upgraded, while the rest are not. An upgrade configuration $U$ is considered feasible if the cost of upgrades with respect to $U$, $C(U) = \sum_{i \in V_U} c_i$, does not exceed a given upgrading budget $B \geq 0$. Let $\mathcal{U}$ be the family of all upgrade configurations, and $D(T;U)$ the total delay (sum of all weight edges) of the minimum spanning tree $T \in \mathcal{T}$ under the upgrade configuration $U \in \mathcal{U}$. We are interested in the pair $(T^*;U^*)$ that minimizes the total delay of the minimum spanning tree after upgrades, that is, $(T^*;U^*) = \arg\min\{D(T;U)|C(U) \leq B; U \in \mathcal{U}, T \in \mathcal{T}\}$ .

An illustration for the MDUMSTP is shown in Figure 1.2, in which the upgrade costs for the nodes [0, 1, 2, 3, 4] are [6, 6, 12, 13, 9], with a budget of 28. The upgraded

nodes are 0, 3, and 4 (highlighted in gray). This results in a solution value of 27, which corresponds to the sum of the weights of the edges in the minimum spanning tree after the upgrades.




(a) (b)

Figure 1.2: Example of an instance (a) and its solution (b) for the MDUMSTP.

## 1.1 Our Contributions

In this work, to solve the Minimum Delay Upgrading Minimum Spanning Tree Problem (MDUMSTP), we propose a new Integer linear Programming (IP) formulation and compared it with the exact model found in the literature. By incorporating the separation and addition of valid inequalities, we enhance the formulation, leading to tighter bounds and improved computational performance. Moreover, we introduce a preprocessing procedure that significantly reduces the size of the instances. Numerical experiments show that our Branch-and-Cut algorithm outperforms the existing exact model in the literature for several test instances and, furthermore, solves to optimality previously unsolved instances.

## 1.2 Dissertation Outline

The dissertation is organized as follows. Chapter 2 reviews related work, discussing previous studies on solution methods for the MDUMSTP. Chapter 3 introduces an existing mathematical formulation, a new proposed model, and an existing primal heuristic. Chapter 4 presents a Branch-and-Cut algorithm found in the literature and one proposed by us. In Chapter 5, we develop a new preprocessing procedure. Chapter 6 provides numerical results for the proposed preprocessing procedure, compares our model with the existing one from the literature across several instances, and evaluates the performance of the primal heuristic. Finally, in Chapter 7 there are conclusions and directions for future research.

# Chapter 2

# Related Work

Several problems focused on finding budget constrained optimal upgrading configurations have been proposed in the literature. In general, there are two types of problems involving graph upgrades: edges-based upgrade and node-based upgrade. Both approaches aim to optimize a specific network performance criterion.

In problems of the edges-based upgrade type, each edge can be upgraded with a certain associated cost to decrease its weight, with the goal of: making a minimum spanning tree spending at most an upgrade budget (DRANGMEISTER *et al.*, 1998, KRUMKE *et al.*, 1997); creating a network such that the longest path is minimized and the total cost associated with reductions does not exceed a given budget (HAMBRUSCH and TU, 1997); among others (KRUMKE *et al.*, 1996, 1998, MARATHE *et al.*, 1998).

In this work we will not address edge-based upgrade problems but rather node-based upgrades. In this case, each node can be upgraded with an associated cost to decrease the weight of the edges adjacent to it, in order to: designing a spanning tree such that the cost spent on upgrades is minimized and the bottleneck (e.g., the maximum weight of an edge in a subgraph) is at most a given parameter (KRUMKE *et al.*, 1999a); making a spanning tree such that the bottleneck is minimized while spending at most an upgrade budget (KRUMKE *et al.*, 1999a); and others (BARBOSA *et al.*, 2021, BLANCO and

MARÍN, 2019, MEDYA *et al.*, 2018).

Many node-based upgrade problems were inspired by the model proposed by (PAIK and SAHNI, 1995), one of the first references on the topic. In this work, the author defined $d_e^1 = \alpha d_e^0$ and $d_e^2 = \alpha d_e^1$, $\forall e \in E$, for a given $\alpha \in (0, 1)$. Additionally, the computational complexity of the five variants of network upgrade problems presented is analyzed, demonstrating that they range from problems solvable in polynomial-time to NP-hard ones.

Although there are numerous variations of node-based upgrade problems, few works have addressed the MDUMSTP. Generalizing the node upgrade model introduced by (PAIK and SAHNI, 1995), the problem was defined for the first time in (KRUMKE *et al.*, 1999b) under the nomenclature *Dual Upgrading MST Problem* (which will not be adopted in this work but instead as was defined by (ALVAREZ-MIRANDA and SINNL, 2017)). According to the authors, they provided an $(1, (1 + \epsilon)^2 O(\log |V|))$-approximation algorithm, that in the context of bicriteria optimization, implies that the algorithm either: (i) yields a solution where the value $C(U)$ is at most $(1+\epsilon)^2 O(\log |V|)$ times the given budget $B$, and $D(T, U)$ represents the minimum value of a solution that meets the budget constraint; or (ii) accurately reports that no subgraph satisfies the budget constraint $C(U) \leq B$.

The first to solve the problem and certify optimality was (ALVAREZ-MIRANDA and SINNL, 2017). The paper presents a Integer linear Programming-based model to address the problem, including Branch-and-Cut algorithms and Lagrangian Relaxation approaches. Furthermore, the authors conducted a thorough analysis of valid inequalities, primal heuristics, variable fixing procedures and branching priorities. They also created two sets of instances, which are used in this work and discussed in Section 6.1.

# Chapter 3

# Mathematical Formulation

In this chapter, we will present the mathematical formulation of MDUMSTP, including both the formulation found in the literature and our own. We will define the main variables and constraints of the optimization problem and also describe a primal heuristic for solving it.

## 3.1 Model Proposed by (ALVAREZ-MIRANDA and SINNL, 2017)

First, define the bi-directed graph $G_A = (V, A)$, so that $A = \{(i,j),(j,i) \mid \forall e : \{i,j\} \in E\}$, in other words, each edge is replaced by two directed arcs (one in each direction). Furthermore, the edge weight values are redefined so that: $d^0_{ij} = d^0_{ji} = d^0_e$, $d^1_{ij} = d^1_{ji} = d^1_e$ and $d^2_{ij} = d^2_{ji} = d^2_e$, $\forall e : \{i,j\} \in E$.

Let $x_i \in \{0,1\},\ \forall\, i \in V$, be a binary variable such that, $x_i = 1$ if the node $i$ is upgraded, and $x_i = 0$ otherwise. Let $y^l_{ij} \in \{0,1\},\ \forall\,(i,j) \in A$, $l \in \{0,1,2\}$, be a binary variable such that, $y^l_{ij} = 1$ if edges $(i,j)$ at level $l$ are part of the solution; and $y^l_{ij} = 0$ otherwise.

For a given set of nodes $S \subseteq V$, let $\delta^-(S) = \{(i,j) \in A \,|\, i \in V \setminus S, j \in S\}$ (resp. $\delta^+(S) = \{(i,j) \in A \,|\, i \in S, j \in V \setminus S\}$), i.e., the set of incoming (resp. outgoing) arcs of a given subset of nodes $S \subseteq V$.

The optimization model for the MDUMSTP is defined as:

$$\min \quad \sum_{(i,j)\in A} (d_{ij}^0 y_{ij}^0 + d_{ij}^1 y_{ij}^1 + d_{ij}^2 y_{ij}^2) \tag{3.1}$$

s.t.

$$\sum_{i\in V} c_i x_i \leq B \tag{3.2}$$

$$y_{ij}^1 + y_{ji}^1 + 2y_{ij}^2 + 2y_{ji}^2 \leq x_i + x_j, \qquad \forall\, (i,j) \in E \tag{3.3}$$

$$\sum_{(i,j)\in\delta^-(S)} (y_{ij}^0 + y_{ij}^1 + y_{ij}^2) \geq 1, \qquad \forall\, S \subseteq V, S \setminus \{r\} \tag{3.4}$$

$$\sum_{(i,j)\in\delta^-(j)} (y_{ij}^0 + y_{ij}^1 + y_{ij}^2) = 1, \qquad \forall\, j \in V \setminus \{r\} \tag{3.5}$$

$$y_{ij}^2 + y_{ji}^2 \leq x_j, \qquad \forall (i,j) \in A \tag{3.6}$$

$$\sum_{(i,j)\in A} y_{ij}^2 \leq \sum_{i\in V} x_i - 1 \tag{3.7}$$

$$\sum_{(i,j)\in\delta^-(j)} y_{ij}^2 \leq x_j, \qquad \forall\, j \in V \setminus \{r\} \tag{3.8}$$

$$y_{ij}^l \in \{0,1\}, \qquad \forall\, (i,j) \in A, l \in \{0,1,2\} \tag{3.9}$$

$$x_i \in \{0,1\}, \qquad \forall\, i \in V \tag{3.10}$$

The optimization model for solving the MDUMSTP presented in (3.1)-(3.10) was proposed by (ALVAREZ-MIRANDA and SINNL, 2017). The objective function to be minimized in (3.1) represents the total (sum of) weights of the edges chosen for $T \in \mathcal{T}$, with their respective levels. In (3.2), the cost of an upgrade configuration $U$ must not exceed a given budget $B$. Constraint (3.3) determines the level of the edge $(i,j)$ based on whether or not $x_i$ and $x_j$ are upgraded, with three possibilities: if $x_i + x_j = 0$, the edge $(i,j)$ will be at level 0; if $x_i + x_j = 1$, the edge $(i,j)$ can be at level 1; and if $x_i + x_j = 2$, the edge $(i,j)$ can be at level 2. A necessary condition for $T \in \mathcal{T}$ to be a

directed arborescence is given by the *cut set constraint* (3.4), ensuring a directed path from a predefined root node $r \in V$ (in all instances we set $r = 0$) to every other node, with all variables arcs in $\delta^-(r)$ were fixed to zero since an arborescence has no incoming arcs to the root node, resulting in a cycle-free graph with $V - 1$ edges.

Although constraints (3.5)–(3.8) are not necessary for a complete characterization of the feasible solution set $(T, U)$, with $T \in \mathcal{T}$ and $U \in \mathcal{U}$, these valid inequalities strengthen the model. When considering $S = \{j\}$ for each $j \in V \setminus \{r\}$, inequality (3.4) is replaced by equality (3.5), ensuring that one edge must arrive at node $j$. Constraints (3.6) and (3.7) are derived from the *Generalized Subtour-Elimination Constraints* (GSECs) (see, e.g., (GOEMANS, 1994)). In our problem context, they ensure that for any subset $S$ of upgraded nodes, at most $|S| - 1$ edges $(i, j)$ at level 2 are selected, for all $i, j \in S$ and $(i, j) \in A$. Since the complete family of GSECs has exponential size, only those for $|S| = 2$ (3.6) and a modified version for $|S| = |V|$ (3.7) are considered. Constraint (3.7) is valid under the assumption that only non-trivial solutions are considered, meaning that at least one upgrade is done. Furthermore, constraint (3.8) states that if node $j \in V \setminus \{r\}$ is upgraded, it can have at most one incoming arc at level 2; otherwise, it has none.

Finally, the constraints (3.9) and (3.10) define as binary the variables: $y_{ij}^l$, $\forall\, (i, j) \in A, l \in \{0, 1, 2\}$; and $x_i$, $\forall\, i \in V$.

## 3.2 Our Proposed Model

In our approach, we propose creating a model in which selecting an edge automatically determines both the corresponding level and whether the node has been upgraded, while simultaneously forming a minimum spanning tree, rather than treating these aspects separately.

Consider $A^+ = A \cup \{(r', r)\}$ and $V^+ = V \cup r'$. For a given set of nodes $S \subseteq V$,

let $\sigma^-(S) = \{(i,j) \in A^+ \,|\, i \in V^+ \setminus S, j \in S\}$ (resp. $\sigma^+(S) = \{(i,j) \in A^+ \,|\, i \in S, j \in V^+ \setminus S\}$), i.e., the set of incoming (resp. outgoing) arcs of a given subset of nodes $S \subseteq V$.

For this approach, the variable $y^1_{ij}$ was replaced by two others ($y^{10}_{ij} + y^{01}_{ji}$), which represents a level 1 edge with upgrade at node $i$ and not at node $j$. The edge variables have been redefined so that $y^{l_1 l_2}_{ij}$ indicates that the edge $(i,j)$ is at level $l_1 + l_2$, where $l_1, l_2 \in \{0,1\}$. The variable $x_k$, which indicated the upgrade at node $k$, for all $k \in V$, has been eliminated, for comparison, the upgrade at node $k$ is given by $\sum_{(i,k)\in\sigma^-(k)}(y^{01}_{ik}+y^{11}_{ik})$, for all $k \in V$. Since $G_A$ has no incoming arcs to the root node $r$, an artificial node $r'$ was introduced to indicate whether node $r$ has been upgraded. As it does not make sense the artificial root to be upgraded, we imposed that $y^{10}_{r'r}$ and $y^{11}_{r'r}$ are set to 0. Still, no arc can reach the root r unless it comes from from $r'$.

Our optimization model for solving the MDUMSTP is defined as:

$$\min \quad \sum_{(i,j)\in A} (d^0_{ij} y^{00}_{ij} + d^1_{ij}(y^{01}_{ij} + y^{10}_{ij}) + d^2_{ij} y^{11}_{ij}) \tag{3.11}$$

s.t.

$$\sum_{k\in V} c_k \sum_{(i,k)\in\sigma^-(k)} (y^{01}_{ik} + y^{11}_{ik}) \leq B, \tag{3.12}$$

$$y^{l0}_{ij} + y^{l1}_{ij} \leq \sum_{(h,i)\in\sigma^-(i)\setminus\{(j,i)\}} (y^{0l}_{hi} + y^{1l}_{hi}), \qquad \forall\,(i,j) \in A, l \in \{0,1\} \tag{3.13}$$

$$\sum_{(i,j)\in\sigma^-(S)} (y^{00}_{ij} + y^{01}_{ij} + y^{10}_{ij} + y^{11}_{ij}) \geq 1, \qquad \forall\, S \subseteq V^+, S \setminus \{r'\} \tag{3.14}$$

$$\sum_{(i,j)\in\sigma^-(j)} (y^{00}_{ij} + y^{01}_{ij} + y^{10}_{ij} + y^{11}_{ij}) = 1, \qquad \forall\, j \in V \tag{3.15}$$

$$y^{l_1 l_2}_{ij} \in \{0,1\}, \qquad \forall\,(i,j) \in A^+; l_1, l_2 \in \{0,1\} \tag{3.16}$$

Similar to (3.1), the objective function minimized in (3.11) represents the total (sum of) weight of the selected edges in $T \in \mathcal{T}$, considering their respective levels. The

constraint in (3.12) ensures that the cost of an upgrade configuration does not exceed a given budget $B$, as in (3.2). To maintain coherence between edge levels, constraint (3.13) enforces that an edge with (without) upgrade can only leave node $i$ if an edge with (without) upgrade arrives at node $i$, $\forall\, i \in V$. The cut-set constraint in (3.14) prevents subcycles, while the valid inequality in (3.15) guarantees that one edge must arrive at node $i$, $\forall\, i \in V$. Finally, constraint (3.16) defines as binary the variables $y_{ij}^{l_1 l_2}, \forall\, (i,j) \in A^+, l_1, l_2 \in \{0,1\}$.

## 3.3 Primal Heuristic

To develop an efficient and high-quality primal heuristic, (ALVAREZ-MIRANDA and SINNL, 2017) proposed an approach based on, during the convergence of the optimization algorithm, linear relaxations tend to guide effective upgrade configurations.

Let $\tilde{x}$ and $\tilde{y}$ represent the values of the upgrade variables and arc variables, respectively, in the LP-relaxation at a branch-and-bound node. Feasible solutions are then constructed as follows:

1. Sort the upgrade variables $\tilde{x}$ in descending order;
2. While the budget allows, select upgrades based on $\tilde{x}$;
3. Compute a minimum spanning tree using weights induced by $\tilde{x}$ through polynomial-time algorithms (KRUSKAL, 1956, PRIM, 1957).

To apply the above heuristic to our model presented in Section 3.2, the value of the upgrade variable $\tilde{x}$ at each vertex $k \in V$ is given by $\sum_{(i,k)\in\sigma^-(k)}(\tilde{y}_{ik}^{01} + \tilde{y}_{ik}^{11})$. Still, in both models, the primal heuristic is applied at each iteration of the Branch-and-Cut, but only at the root node.

# Chapter 4

# Branch-and-Cut Algorithm

In many cases, when an optimization model has a large number of constraints, they are not initially included but are dynamically added as needed when violations are detected, following a specific separation procedure.

## 4.1 Proposed by (ALVAREZ-MIRANDA and SINNL, 2017)

As done in (ALVAREZ-MIRANDA and SINNL, 2017), consider $\tilde{y}$ the values of the arc variables obtained from the LP-relaxation solution of the MDUMSTP at a specific node in the Branch-and-Bound tree. The separation procedure involves determining the max-flow (GOLDBERG and TARJAN, 1988) from $r$ to each other node $k \in V \setminus \{r\}$ in $G_A$, where the capacities of the arcs $(i,j), \forall\,(i,j) \in A$, are given by $\tilde{w}_{ij} = \tilde{y}^0_{ij} + \tilde{y}^1_{ij} + \tilde{y}^2_{ij} + \Delta$, with $\Delta = (\epsilon/10)/|A|$ a given parameter (which we chose to introduce) that ensures the selection of the cut with the smallest number of edges, in the case of multiple cuts have the same value. If the maximum flow from $r$ to $k$ is less than $1 - \epsilon$, with a tolerance $\epsilon = 0.01$, the corresponding cut-set $\delta^-(S)$ indicates a violated connectivity cut (3.4). In case the LP-relaxation solution is integer, when the first violated cut is found, the separation procedure is aborted for the remaining $k$ nodes.

For each maximum flow calculated from $r$ to some $k$, we enhance the separation

routine by updating the capacities of the arcs $(i, j)$ as $\min(1, 1 - \text{maxflow}(r, k) + \tilde{w}_{ij})$. In this manner, the arc $(i, j)$ becomes less attractive when applying the separation routine to another node $k' \neq k$. This procedure aims to identify stronger cuts that are not redundant or dominated by others, while also preventing excessive growth in the size of the linear programming problem to be solved. This approach is less restrictive than orthogonal cuts (LUCENA and RESENDE, 2004), where an arc $(i, j)$ cannot appear in a cut for another node $k' \neq k$ (as done in (ALVAREZ-MIRANDA and SINNL, 2017)).

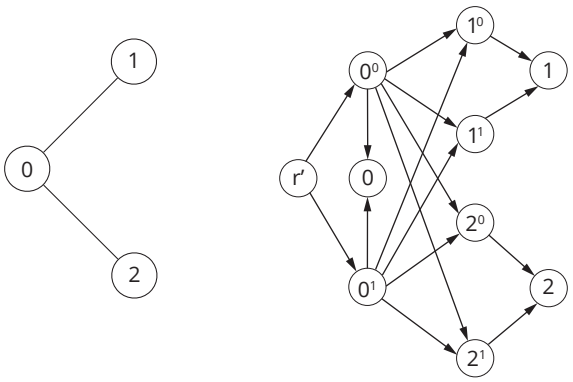
In our numerical experiments, we observed that instead of always choosing $k$ in a fixed order (e.g., V in ascending order), the order of selection varied throughout each separation process, following a deterministic logic for better reproducibility. We chose $k$ according to the ordered set of nodes $V$, but advancing 10 positions with each separation procedure applied to all nodes.

To avoid adding cut sets (3.4) that do not improve the lower bound during branch-and-bound, the separation process is aborted if, after five iterations, the lower bound at a given branch-and-bound node changes by less than 0.001 units.

## 4.2 Proposed by Us

In order to strengthen our model presented in Section 3.2, we developed a procedure for separating valid inequalities.

Therefore, let $H = (V_H, A_H)$, be a graph such that $V_H = \{r'\} \cup \{i^l : i \in V, l \in \{0, 1\}\} \cup \{i : i \in V\}$ and $A_H = \{(r', r^l) : l \in \{0, 1\}\} \cup \{(i^{l_1}, j^{l_2}) : (i, j) \in A; l_1, l_2 \in \{0, 1\}\} \cup \{(i^l, i) : i \in V, l \in \{0, 1\}\}$. We can also view this as an expanded graph $H$. In Figure 4.1, we present an instance (Figure 4.1(a)) and its corresponding expanded graph H (Figure 4.1(b)).




(a) (b)

Figure 4.1: Example of an instance (a) and its corresponding expanded graph H (b).

Consider $\tilde{y}$ the values of the arc variables obtained from the LP-relaxation solution of the MDUMSTP at a specific node in the Branch-and-Bound tree. The separation works by calculating the max-flow (GOLDBERG and TARJAN, 1988) from $r'$ to each of the other nodes $k \in V$ in H, with capacities of arcs: $(i^{l_1}, j^{l_2})$ set to $\tilde{y}_{ij}^{l_1 l_2} + \Delta, \forall\,(i,j) \in A, l_1, l_2 \in \{0,1\}$; $(i^l, i)$ set to 1, with $i \in V, l \in \{0,1\}$; and $(r', r^l)$ set to $\tilde{y}_{r' r^l}^{0l}$, with $l \in \{0,1\}$ . If the maximum flow from $r'$ to $k$ is less than 1 - $\epsilon$, the corresponding cut-set indicates a violated cut in H, where $\Delta = (\epsilon/10)/|A_H|$ is a given parameter that ensures the selection of the cut with the smallest number of edges, in case of multiple cuts have the same value, and in addition to a given tolerance parameter $\epsilon = 0.01$.

For each maximum flow calculated from $r'$ to some $k$, we enhance the separation routine by updating the capacities of the arcs in $H$ to $\min(1, 1 - \text{maxflow}(r', k) + \tilde{h})$, with $\tilde{h}$ is the capacity of $H$ before the update. Also, instead of always choosing $k$ in a fixed order (e.g., V in ascending order), we chose $k$ according to the ordered set of nodes $V$, but advancing $10 \cdot$ (the number of separation procedure applied) + |V|/2 positions with each separation procedure applied to all nodes.

Due to the fact that the number of edges in $H$ is up to four times greater than in $G_A$, computing the maximum flow can be significantly more computationally expensive.

To mitigate this impact, we interrupt the separation after calculating $|V|/5$ maximum flows or when five violated cuts are identified.

In our approach, we perform two separation procedures in each iteration: one as described above; and the other in a similar way to what was described in Section 4.1, but replacing the constraint (3.4) by (3.14), and the capacities of the arcs $(i, j)$ are given by $\tilde{w}_{ij} = \tilde{y}_{ij}^{00} + \tilde{y}_{ij}^{01} + \tilde{y}_{ij}^{10} + \tilde{y}_{ij}^{11} + \Delta$.

Finally, to avoid adding cuts that do not improve the lower bound during branch-and-bound, the separation process is aborted if, after five iterations, the lower bound at a given branch-and-bound node changes by less than 0.001 units.

# Chapter 5

# Pre-Processing

To reduce computational cost, in this chapter we will present a pre-processing procedure that identifies edges and arcs that do not belong to the optimal solution. Once identified, we will remove them.

The procedure is an adaptation of a classical approach to MSTP. If there is an alternative path between two vertices $i, j$ in a graph whose total edge weight does not exceed that of $(i, j)$, then $(i, j)$ can be excluded from the solution without loss of optimality (see Figure 5.1, subfigures (a), (h) and (o)). A proof of this statement can be found in (DIAS and SIMONETTI), where the authors adapted this idea to the problem they studied. In our context, we will take into account the edge levels. An illustration of how the pre-processing works is shown in Figure 5.1. In this Figure, we have seven instances, and each instance is identified with a number between 1 and 7. In the first line of Figure 5.1 we show the instance; in the second line, we highlight in red or blue the components that are analyzed by the pre-processing; lastly, in the third line, we show how the instance became after applying the pre-processing.

Let $G_P$ be an undirected graph, where each edge $(i, j)$ in $G_P$ has capacity $d^0_{ij}$, then for any edge $(i, j)$ in $E$:

i) **Edge pre-processing procedure**: If there exists a path between $i$ and $j$ that

does not include $(i, j)$ and whose cost is less than or equal to $d^2_{ij}$, then the edge $(i, j)$ at levels 2, 1 and 0 can be removed from the problem (see Figure 5.1, subfigures (b), (i) and (p), and lines 9–10 of Algorithm 1);

ii) **Levels 1 and 0 edges pre-processing procedure**: If there exists a path between $i$ and $j$ that does not include $(i, j)$ and whose cost is less than or equal to $d^1_{ij}$, then the edges $(i, j)$ at levels 1 and 0 can be removed from the problem (see Figure 5.1, subfigures (c), (j) and (q), and lines 27–28 of Algorithm 1);

iii) **Level 2 edge pre-processing procedure**: If by adjusting the capacities of the edges incident to $i$ and $j$, such that $(i, j) = d^1_{i,j}$ for all $(i, j) \in \delta^-(\{i, j\})$, there exists a path between $i$ and $j$ that does not include $(i, j)$ with a cost less than or equal to $d^2_{ij}$, then the level 2 edge $(i, j)$ can be removed from the problem (see Figure 5.1, subfigures (d), (k) and (r), and lines 12–25 of Algorithm 1);

iv) **Level 1 edge pre-processing procedure**: If by adjusting the capacities of the edges incident to $i$ such that $(i, j) = d^1_{i,j}$ for all $(i, j) \in \delta^-(\{i\})$, there exists a path between $i$ and $j$ that does not include $(i, j)$ and has a cost less than or equal to $d^1_{ij}$, then, by applying the same procedure to the edges incident to $j$ (setting the capacities of the edges incident to $i$ equal to $d^0_{ij}$), the level 1 edge $(i, j)$ can be removed from the problem (see Figure 5.1, subfigures (e), (l) and (s), and lines 30–47 of Algorithm 1). In instance five, we used the colors red and blue to highlight the two possible combinations that lead to the elimination of the level 1 edge;

v) **Level 0 edge pre-processing procedure**: If there exists a path between $i$ and $j$ that does not include $(i, j)$ and whose cost is less than or equal to $d^0_{ij}$, then the level 0 edge $(i, j)$ can be removed from the problem (see Figure 5.1, subfigures (f), (m) and (t), and lines 48–50 of Algorithm 1);

vi) **Level 1 arc pre-processing procedure**: If by adjusting the capacities of the

edges incident to $i$ such that $(i,j) = d^1_{i,j}$ for all $(i,j) \in \delta^-(\{i\})$, there exists a path between $i$ and $j$ that does not include $(i,j)$ and has a cost less than or equal to $d^1_{ij}$, then only the variables $y^{10}_{ij}$ and $y^{01}_{ji}$ can be removed from the problem in our proposed model in Section 3.2 (see Figure 5.1, subfigures (g), (n) and (u), in which the color green associated to the level 1 edge indicates this edge remains active in just one direction, and lines 30–44 of Algorithm 1).

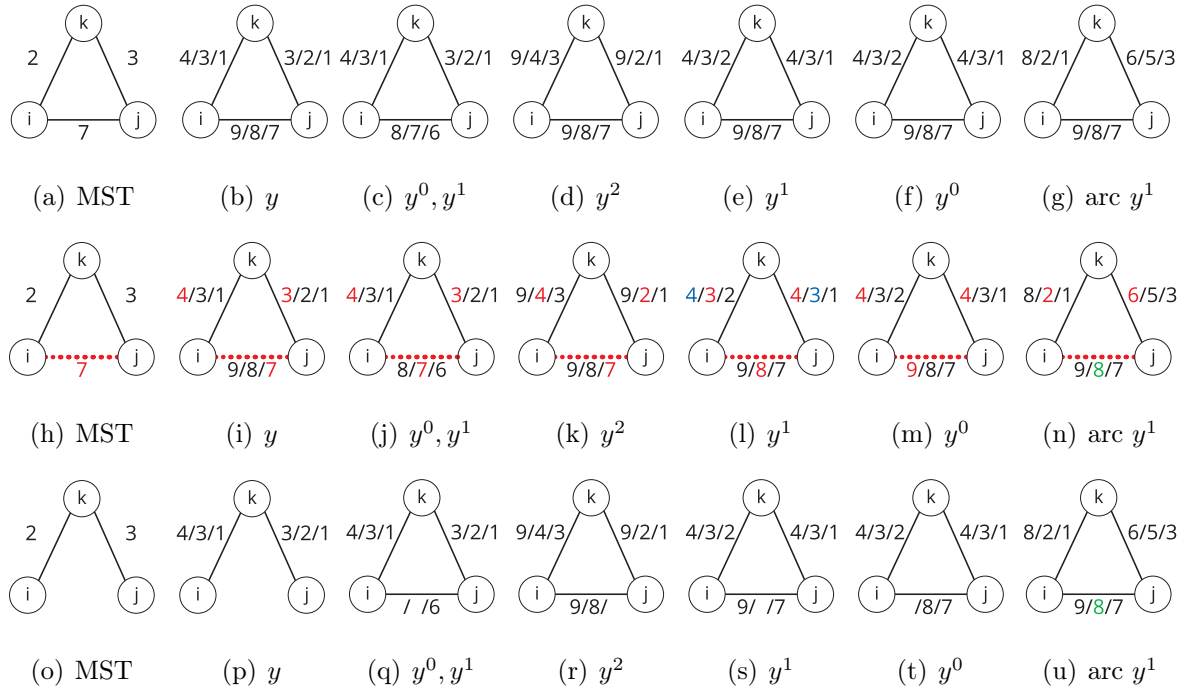



Figure 5.1: Pre-processing illustration.

The pseudocode for preprocessing procedure described above is presented in Algorithm 1. The matrix dist$(i,j)$, computed by the Floyd-Warshall algorithm (FLOYD, 1962) (see Algorithm 1, lines 1–7), contains the shortest distances between all vertex pairs in $V$, considering the edge weights given by $d^0$. The algorithm was adapted in line 6 by replacing "greater" by "greater or equal", allowing the removal of edges or arcs if an alternative path with the same sum of edge weights exists. Still, the Numba library (LAM *et al.*, 2015) is used to enhance the runtime performance of Algorithm 1. Numba

is a performance optimization tool for Python that compiles functions into machine code at runtime, significantly accelerating numerical computations. It employs Just-In-Time (JIT) compilation, allowing Python code to execute much faster.

**Algorithm 1:** Pre-processing.

**Input:** For all i,j in V define $dist(i,j) := \{d^0_{ij}, if\ (i,j) \in E; 0, if\ i = j; \infty, otherwise\}$ and $pred(i,j) := False$.
**for** $k, i, j \in V, i \neq k$ *and* $j \neq k$ *and* $i \neq j$ **do**
  **if** $dist(i,j) \geq dist(i,k) + dist(k,j)$ *and* $dist(i,k) + dist(k,j) \neq \infty$ **then**
    $dist(i,j) \leftarrow dist(i,k) + dist(k,j)$
    $pred(i,j) \leftarrow True$
  **end**
**end**
**for** $(i,j) \in E$ **do**
  **if** $d^2_{ij} \geq dist(i,j)$ *and* $pred(i,j) == True$ **then**
    Remove edge $(i,j)$
  **else**
    $BreakLoop := False$
    **for** $k \in$ *vertices adjacent to i,* $k \neq j$ **do**
      **if** $BreakLoop == True$ **then**
        break
      **else**
        **for** $l \in$ *vertices adjacent to j,* $l \neq i$ **do**
          **if** $d^2_{ij} \geq d^1_{ik} + d^1_{lj} + dist(k,l)$ **then**
            Remove edge (i,j) at level 2
            $BreakLoop \leftarrow True$
            break
          **end**
        **end**
      **end**
    **end**
  **end**
  **if** $d^1_{i,j} \geq dist(i,j)$ *and* $pred(i,j) == True$ **then**
    Remove edge (i,j) at level 1 and 0
  **else**
    $TEST := 0$
    **for** $k \in$ *vertices adjacent to i,* $k \neq j$ **do**
      **if** $d^1_{ij} \geq d^1_{ki} + dist(k,j)$ **then**
        Remove arc (i,j) at level 1
        $TEST += 1$
        break
      **end**
    **end**
    **for** $k \in$ *vertices adjacent to j,* $k \neq i$ **do**
      **if** $d^1_{ij} \geq d^1_{kj} + dist(k,i)$ **then**
        Remove arc (j,i) at level
        $TEST += 1$
        break
      **end**
    **end**
    **if** $TEST == 2$ **then**
      Remove edge (i,j) at level 1
    **end**
    **if** $d^0_{ij} \geq dist(i,j)$ *and* $pred(i,j) == True$ **then**
      Remove edge (i,j) at level 0
    **end**
  **end**
**end**

# Chapter 6

# Numerical Experiments

In this chapter, we present the computational results associated with the resolution of the formulations presented in the Chapter 3. The computer we used in the experiments is equipped with an Intel(R) Core(TM) i7-8565U processor, with 1.80GHz processing speed and 32GB DDR4 RAM memory. The operating system used was Windows 10 Home Single Language. To implement the algorithms, we use the Python programming language (VANROSSUM and DRAKE, 2010), version 3.9, with the assistance of the Igraph (CSARDI and NEPUSZ, 2006) library for graph manipulation and Numpy (HARRIS *et al.*, 2020) for all data processing. For all Integer Linear Programming problem solving procedure, we use the solver Cplex 22.1.1.0 in single-thread mode. The maximum CPU time and RAM memory allocated to solve each problem were limited to 1800 seconds and 32 GB, respectively. Furthermore, all Cplex cuts (except *{0,1/2}-cuts* when indicated), as well as Cplex preprocessing and heuristics, have been disabled.

According to (ALVAREZ-MIRANDA and SINNL, 2017), giving higher branching priorities to the x variables can be helpful to better reduce the search space in earlier stages of the Branch-and-Bound tree, boosting the efficacy of his algorithm. Then, in his model we apply this strategy , but not in ours. All others Cplex parameters were left at the default values. In both models, the primal heuristic described in Section 3.3

is applied at each iteration of the Branch-and-Cut, but only at the root node.

We compared seven different configurations (four for the model by (ALVAREZ-MIRANDA and SINNL, 2017) and three for ours):

**(i)** **MIRANDA** – model presented in Section 3.1, with separation of cut set constraints (3.4) like in Section 4.1 and higher branching priorities for the upgrade variables $x$;

**(ii)** **MIRANDA_ZH** – which additionally includes the *{0,1/2}-cuts* (CAPRARA and FISCHETTI, 1996, KOSTER *et al.*, 2009), a native CPLEX cut that is a particular case (with multipliers in $\{0, 1/2\}$) of the Gomory cut (GOMORY, 1958);

**(iii)** **MIRANDA_ZHR** – cut sets are separated only at the root node in configuration MIRANDA_ZH;

**(iv)** **MIRANDA_ZHRP** – applying the preprocessing of Algorithm 1 in MIRANDA_ZHR;

**(v)** **PROPOSED** – our model from Section 3.2, with separation of cuts as in Section 4.2 only at the root node;

**(vi)** **PROPOSED_P** – which additionally applies the preprocessing of Algorithm 1;

**(vii)** **PROPOSED_PD** – adding constraint (3.13) on demand in PROPOSED_P.

## 6.1 Instances

For our computational study, we use two types of instance sets: i) EUCLIDEAN, complete random Euclidean graphs; and ii) C, based on the instance set C of the well-known SteinLIB instance library (KOCH *et al.*, 2001). These instances were created and provided by (ALVAREZ-MIRANDA and SINNL, 2017) and we are grateful for that.

To generate the EUCLIDEAN set, the authors randomly selected $|V|$ points within a $100 \times 100$ plane. The delay values $d^2_{ij}$ were assigned as the Euclidean distance between points $i$ and $j$, rounded up to the next integer, for all $i, j \in V$. The derived delay values

are calculated as $d^1_{ij} = \lceil \alpha^1_{ij} d^2_{ij} \rceil$ and $d^0_{ij} = \lceil \alpha^0_{ij} d^1_{ij} \rceil$, respectively, with $\alpha^1_{ij}$ and $\alpha^0_{ij}$ being randomly chosen from [1.1, 1.3]. The upgrade costs $c_i \in [1, 10]$ are integers randomly assigned for each node $i \in V$. There are ten instances for each $|V|$ in {100, 250, 500}, labeled as $e|V| - k$, where $k \in \{1, ..., 10\}$. As these instances represent complete graphs, the number of edges $|E|$ are {4950, 31125, 124750}, respectively.

To transform the Steiner tree instances from C, the original edge weight of an instance (a random integer in [1, 10]) is used as $d^2$. The values $d^1$, $d^0$ and $c$ are constructed as for EUCLIDEAN. There are 20 instances in this set, and all have the number of nodes $|V| = 500$. The instances are named c01 - c20, and have different numbers of edges $|E|$, more specifically {c01-c05, c06-c10, c11-c15, c16-c20} = {625, 1000, 2500, 12500}.

We tested three different values of budget $B$ for each instance. The budget values are given by $B = \lceil \sum_{i \in V} c_i \cdot b \rceil$, with $b \in \{0.1, 0.2, 0.3\}$.

## 6.2 Pre-Processing Results

The results of the pre-processing performed by Algorithm 1 in Chapter 5 are presented in Table 6.1, which shows the minimum, maximum, and mean number of edges removed at each level, with their respective percentages (indicated between parentheses, rounded to the closest integer). In the notation used, $y^l$ represents the number of edges removed at level $l$, with $l \in \{0, 1, 2\}$, while $y$ the complete removal of the edge at all levels. The percentage of level 1 arcs eliminated was calculated as follows $y^1\, arc/(2 \cdot (|E| - y^1))$.

As the weights of level 2 edges in the EUCLIDEAN instances are defined based on euclidean distances (see Section 6.1), these edges cannot be removed, because they must respect the triangular inequality. As expected, the effect of pre-processing is greater in the denser graphs (c16-c20 and EUCLIDEAN), while it is less noticeable in the sparser ones (c01-c15). The low variation in the number of removed edges and arcs within the same set of instances indicates consistency in the approach. Furthermore,

the pre-processing time was minimal, without causing any considerable impact on the performance of the problem-solving process.

The main results obtained show that, in instances c16-c20, 50% (on average) of level 0 edges, 42% of level 1 edges, 36% of level 2 edges, and 36% of complete edges were removed. For e100, e250, and e500 instances, 63%, 74%, and 80% of level 0 edges were removed, as well as 17%, 29%, and 38% of level 1 edges, respectively. In general, approximately 11% of level 1 arcs were removed from EUCLIDEAN instances.

Table 6.1: Pre-processing results.

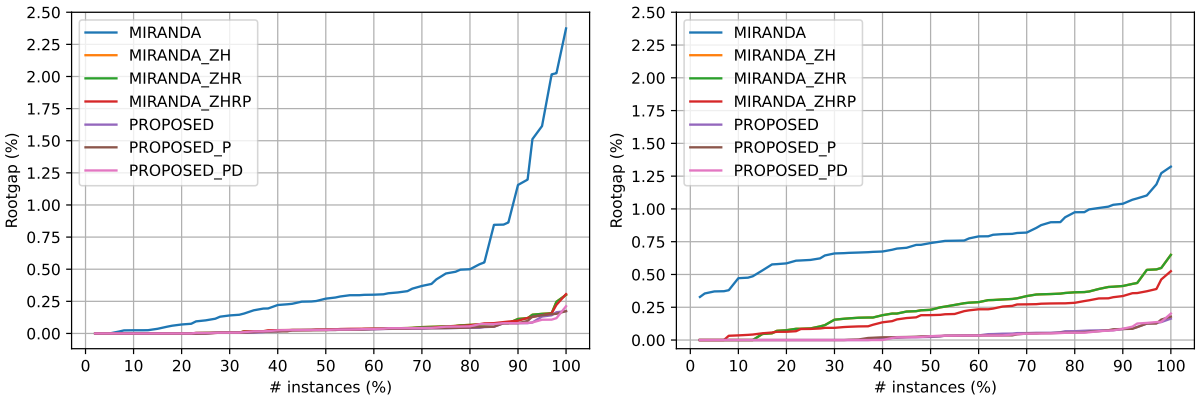
| Instance | | y | $y^0$ | $y^1$ | $y^2$ | $y^1$ arc | \|E\| | t(s) |
|---|---|---|---|---|---|---|---|---|
| | mean | - | - | - | - | - | | |
| c01-c05 | min | - | - | - | - | - | 625 | 0.25 |
| | max | - | 1 ( 0%) | - | - | 1 ( 0%) | | |
| | mean | 2 ( 0%) | 6 ( 0%) | 3 ( 0%) | 2 ( 0%) | 1 ( 0%) | | |
| c06-c10 | min | 1 ( 0%) | 3 ( 0%) | 1 ( 0%) | 1 ( 0%) | - | 1000 | 0.27 |
| | max | 4 ( 0%) | 10 ( 1%) | 4 ( 0%) | 4 ( 0%) | 2 ( 0%) | | |
| | mean | 53 ( 2%) | 150 ( 6%) | 78 ( 3%) | 54 ( 2%) | 11 ( 0%) | | |
| c11-c15 | min | 47 ( 2%) | 144 ( 6%) | 64 ( 3%) | 48 ( 2%) | 9 ( 0%) | 2500 | 0.29 |
| | max | 57 ( 2%) | 156 ( 6%) | 87 ( 3%) | 57 ( 2%) | 13 ( 0%) | | |
| | mean | 4515 (36%) | 6237 (50%) | 5297 (42%) | 4534 (36%) | 94 ( 1%) | | |
| c16-c20 | min | 4444 (36%) | 6175 (49%) | 5225 (42%) | 4466 (36%) | 79 ( 1%) | 12500 | 0.45 |
| | max | 4561 (36%) | 6287 (50%) | 5345 (43%) | 4583 (37%) | 103 ( 1%) | | |
| | mean | - | 3111 (63%) | 858 (17%) | - | 814 (10%) | | |
| e100 | min | - | 3076 (62%) | 772 (16%) | - | 756 ( 9%) | 4950 | 0.01 |
| | max | - | 3148 (64%) | 909 (18%) | - | 858 (11%) | | |
| | mean | - | 22914 (74%) | 9130 (29%) | - | 4974 (11%) | | |
| e250 | min | - | 22680 (73%) | 8810 (28%) | - | 4776 (11%) | 31125 | 0.06 |
| | max | - | 23176 (74%) | 9367 (30%) | - | 5170 (12%) | | |
| | mean | - | 99614 (80%) | 47881 (38%) | - | 18006 (12%) | | |
| e500 | min | - | 99026 (79%) | 46760 (37%) | - | 17625 (11%) | 124750 | 0.50 |
| | max | - | 100185 (80%) | 49070 (39%) | - | 18283 (12%) | | |

$y$: All edge levels have been removed
$y^l$: Edges at level l removed, with $l \in \{0, 1, 2\}$
$y^1$ arc: Level 1 arcs removed, pencentage calculed $y^1 arc/(2 \cdot (|E| - y^1))$

## 6.3 Rootgap and Runtime Results

First, we provide a general comparison of the root gap (Figure 6.1) and the runtime (Figure 6.2) across the different configurations for the models proposed by (ALVAREZ-MIRANDA and SINNL, 2017) and us to identify the best performing ones. Then, we analyze the results in more detail using Tables 6.2, 6.3 and 6.4.

In Figure 6.1, we analyze the root gaps of different configurations of the algorithms proposed by (ALVAREZ-MIRANDA and SINNL, 2017) and ourselves in sets C (Figure 6.1(a)) and EUCLIDEAN (Figure 6.1(b)) with $|V| \in \{100, 250\}$. The rootgap of an instance is calculated as $100 \cdot$ (z* - RB)/z*, such that z* denotes the cost of the best solution found for the instance (using any configuration) and RB is the root bound obtained by the specific configuration. The figure shows the percentage of cases (horizontal axis) with a root gap (in percentage) lower than indicated on the vertical axis.




(a) Set C

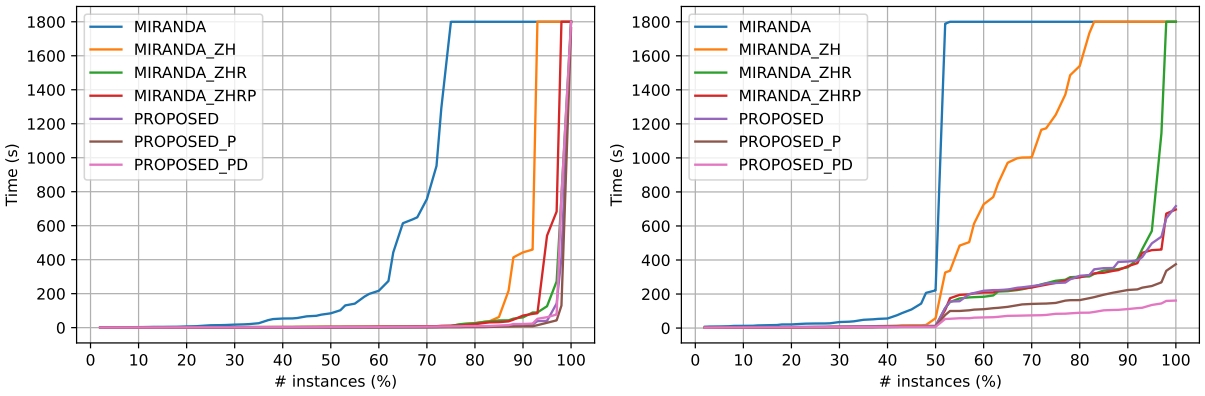
(b) Set EUCLIDEAN with |V| ∈ {100, 250}

Figure 6.1: Rootgap comparison of algorithms.

The MIRANDA configuration exhibits a substantially larger root gap compared to the other configurations across both set of instances. It is observed that adding the {0,1/2}-cut significantly reduces the root gap. In the EUCLIDEAN cases, preprocessing proved beneficial, likely because reducing the number of variables in the model and, con-

sequently, the computational cost of separating inequalities applied by CPLEX enabled more cuts to be added. For our model, all configurations showed similar performance, slightly better than (ALVAREZ-MIRANDA and SINNL, 2017) in the C cases and significantly better in the EUCLIDEAN cases.

In Figure 6.2, we analyze the computational time of different configurations of the algorithms proposed by (ALVAREZ-MIRANDA and SINNL, 2017) and ourselves in sets C (Figure 6.2(a)) and EUCLIDEAN (Figure 6.2(b)) with $|V| \in \{100, 250\}$. The figure shows the percentage of cases solved (horizontal axis) below the time indicated on the vertical axis. It is worth noting that, in Figure 6.2(b), the 50% mark represents the exact division between the $e100$ instances (below 50%) and the $e250$ instances (above 50%), due to the difference in density and, consequently, the difficulty in solving them.




(a) Set C

(b) Set EUCLIDEAN with |V| $\in \{100, 250\}$

Figure 6.2: Runtime comparison of algorithms.

It is observed that, for the C and EUCLIDEAN sets, MIRANDA configuration exhibits lower computational performance compared to all other configurations. The addition of {0,1/2}-cut significantly reduces the computational time, as does the inclusion of cuts only at the root node. On the other hand, pre-processing did not have a significant impact on its performance, likely because removing an edge from graph $G_A$ requires

eliminating all levels associated with it, rather than just a part of the edge.

For our model, in the C set, the run time does not vary significantly between the different configurations, and the same happens in the EUCLIDEAN set with $|V| = 100$. However, for the EUCLIDEAN set with $|V| = 250$, substantial improvements were observed when applying pre-processing, likely due to the fact that removing any arc (regardless of its level) reduces graph $H$ and, consequently, the complexity of separating valid inequalities. In the cases with $|V| = 250$ for the PROPOSED_PD configuration, performance improved even further, due to the high computational cost of solving a large model with all the constraints added a priori.

Finally, it is clear that, in general, our algorithms are faster than (ALVAREZ-MIRANDA and SINNL, 2017) in both sets of instances.

In Table 6.2, we present the results for the set of instances C using MIRANDA_-ZHRP, PROPOSED_P, and PROPOSED_PD configurations. The table reports the value of the best-found solution z*, the runtime in seconds t(s) (with TL indicating that the time limit of 1800 seconds was reached), the root gap in percentage g(%), whether the instance was solved to optimality, and the optimality gap (calculated as $100 \cdot (z^* - LB)/z^*$ , where LB is the lower bound) otherwise. Additionally, the number of branch-and-bound nodes (nodes) is included. For better visualization, we highlight in bold the configurations that achieve the best results in terms of time or gap.

It can be observed that in no instance did MIRANDA_ZHRP perform better in terms of runtime, which was predominantly achieved by PROPOSED_P in almost all cases (except for instance the c19 with b=0.1 which was better by PROPOSED_PD). One possible explanation is that the advantage of adding constraints on demand (PROPOSED_PD) becomes more significant for very large and dense instances, which is not the case for those analyzed in the Table 6.2. For these instances, it is more advantageous to include all constraints a priori.

Table 6.2: Results for the instance set C.

| Instance | b | MIRANDA_ZHRP | | | | PROPOSED_P | | | | PROPOSED_PD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes |
| c01 | 0.1 | 3422 | 1.62 | **0.00** | 0 | 3422 | **0.16** | **0.00** | 0 | 3422 | 0.94 | **0.00** | 0 |
| c01 | 0.2 | 3163 | 3.06 | **0.01** | 7 | 3163 | **0.45** | **0.01** | 7 | 3163 | 1.48 | **0.01** | 11 |
| c01 | 0.3 | 2978 | 1.88 | **0.03** | 0 | 2978 | **0.09** | **0.03** | 0 | 2978 | 0.94 | **0.03** | 0 |
| c02 | 0.1 | 3283 | 3.00 | **0.02** | 0 | 3283 | **0.25** | 0.02 | 0 | 3283 | 2.88 | 0.05 | 125 |
| c02 | 0.2 | 3029 | 2.38 | **0.00** | 0 | 3029 | **0.80** | **0.00** | 0 | 3029 | 2.48 | **0.00** | 3 |
| c02 | 0.3 | 2855 | 3.59 | **0.00** | 0 | 2855 | **0.56** | **0.00** | 0 | 2855 | 3.98 | **0.00** | 0 |
| c03 | 0.1 | 3299 | 2.78 | **0.01** | 0 | 3299 | **0.45** | **0.01** | 0 | 3299 | 1.45 | **0.01** | 0 |
| c03 | 0.2 | 3034 | 2.17 | **0.00** | 1 | 3034 | **0.69** | 0.01 | 7 | 3034 | 2.45 | 0.01 | 54 |
| c03 | 0.3 | 2854 | 1.92 | **0.02** | 0 | 2854 | **0.98** | 0.03 | 0 | 2854 | 1.16 | 0.03 | 0 |
| c04 | 0.1 | 3337 | 3.11 | **0.00** | 0 | 3337 | **0.25** | **0.00** | 0 | 3337 | 1.12 | **0.00** | 0 |
| c04 | 0.2 | 3085 | 2.84 | **0.02** | 0 | 3085 | **0.61** | **0.02** | 0 | 3085 | 0.80 | **0.02** | 0 |
| c04 | 0.3 | 2910 | 3.27 | **0.03** | 0 | 2910 | **0.56** | **0.03** | 0 | 2910 | 1.41 | **0.03** | 0 |
| c05 | 0.1 | 3315 | 1.66 | **0.00** | 0 | 3315 | **0.20** | **0.00** | 0 | 3315 | 1.14 | **0.00** | 0 |
| c05 | 0.2 | 3068 | 2.84 | **0.02** | 0 | 3068 | **0.20** | **0.02** | 0 | 3068 | 1.08 | **0.02** | 0 |
| c05 | 0.3 | 2887 | 1.89 | **0.00** | 0 | 2887 | **0.27** | **0.00** | 0 | 2887 | 0.92 | **0.00** | 0 |
| c06 | 0.1 | 2534 | 8.12 | **0.00** | 5 | 2534 | **2.00** | **0.00** | 0 | 2534 | 3.88 | **0.00** | 0 |
| c06 | 0.2 | 2318 | 4.48 | **0.03** | 0 | 2318 | **0.42** | **0.03** | 0 | 2318 | 2.62 | **0.03** | 0 |
| c06 | 0.3 | 2172 | 2.62 | **0.01** | 0 | 2172 | **0.41** | **0.01** | 0 | 2172 | 0.91 | **0.01** | 0 |
| c07 | 0.1 | 2532 | 3.20 | **0.00** | 0 | 2532 | **0.44** | **0.00** | 0 | 2532 | 1.86 | **0.00** | 0 |
| c07 | 0.2 | 2323 | 3.31 | **0.03** | 0 | 2323 | **1.33** | **0.03** | 0 | 2323 | 2.06 | **0.03** | 1 |
| c07 | 0.3 | 2179 | 3.61 | **0.00** | 0 | 2179 | **0.41** | **0.00** | 0 | 2179 | 1.86 | **0.00** | 0 |
| c08 | 0.1 | 2461 | 5.05 | **0.00** | 0 | 2461 | **0.66** | **0.00** | 0 | 2461 | 2.92 | 0.04 | 0 |
| c08 | 0.2 | 2247 | 3.34 | **0.00** | 0 | 2247 | **1.55** | **0.00** | 0 | 2247 | 2.66 | 0.01 | 0 |
| c08 | 0.3 | 2105 | 3.62 | **0.00** | 5 | 2105 | **0.09** | **0.00** | 0 | 2105 | 1.62 | **0.00** | 0 |
| c09 | 0.1 | 2415 | 5.86 | **0.00** | 0 | 2415 | **0.38** | **0.00** | 0 | 2415 | 2.00 | **0.00** | 0 |
| c09 | 0.2 | 2208 | 2.75 | **0.03** | 0 | 2208 | **0.36** | **0.03** | 0 | 2208 | 2.52 | **0.03** | 0 |
| c09 | 0.3 | 2069 | 3.69 | **0.03** | 0 | 2069 | **1.80** | **0.03** | 48 | 2069 | 2.00 | **0.03** | 0 |
| c10 | 0.1 | 2463 | 4.72 | **0.00** | 0 | 2463 | **1.23** | **0.00** | 7 | 2463 | 4.73 | **0.00** | 19 |
| c10 | 0.2 | 2258 | 5.12 | **0.00** | 0 | 2258 | **1.38** | **0.00** | 2 | 2258 | 4.16 | **0.00** | 0 |
| c10 | 0.3 | 2116 | 5.27 | **0.04** | 0 | 2116 | **0.36** | **0.04** | 0 | 2116 | 1.98 | **0.04** | 0 |
| c11 | 0.1 | 1509 | 7.38 | 0.06 | 0 | 1509 | **3.16** | **0.05** | 0 | 1509 | 6.33 | 0.06 | 0 |
| c11 | 0.2 | 1351 | 10.61 | **0.04** | 0 | 1351 | **1.48** | **0.04** | 0 | 1351 | 7.03 | **0.04** | 11 |
| c11 | 0.3 | 1237 | 3.16 | **0.03** | 0 | 1237 | **0.16** | **0.03** | 0 | 1237 | 3.97 | **0.03** | 0 |
| c12 | 0.1 | 1553 | 6.08 | **0.03** | 0 | 1553 | **0.17** | **0.03** | 0 | 1553 | 5.20 | **0.03** | 0 |
| c12 | 0.2 | 1386 | 5.75 | 0.04 | 0 | 1386 | **0.70** | **0.01** | 0 | 1386 | 5.00 | 0.07 | 8 |
| c12 | 0.3 | 1271 | 5.53 | 0.06 | 0 | 1271 | **0.69** | **0.05** | 0 | 1271 | 2.70 | **0.05** | 0 |
| c13 | 0.1 | 1524 | 7.11 | **0.05** | 0 | 1524 | **2.97** | **0.05** | 0 | 1524 | 7.92 | 0.08 | 56 |
| c13 | 0.2 | 1364 | 5.77 | **0.04** | 0 | 1364 | **1.33** | **0.04** | 0 | 1364 | 6.00 | 0.07 | 60 |
| c13 | 0.3 | 1248 | 8.64 | **0.08** | 11 | 1248 | **6.22** | **0.08** | 93 | 1248 | 8.97 | **0.08** | 307 |
| c14 | 0.1 | 1530 | 2.75 | **0.04** | 0 | 1530 | **0.62** | **0.04** | 0 | 1530 | 5.47 | **0.04** | 0 |
| c14 | 0.2 | 1367 | 4.75 | **0.04** | 0 | 1367 | **1.97** | **0.04** | 0 | 1367 | 4.66 | 0.07 | 31 |
| c14 | 0.3 | 1241 | 3.62 | 0.04 | 0 | 1241 | **0.70** | **0.03** | 0 | 1241 | 1.92 | **0.03** | 0 |
| c15 | 0.1 | 1529 | 3.98 | 0.05 | 0 | 1529 | 7.95 | **0.02** | 11 | 1529 | **2.70** | **0.02** | 0 |
| c15 | 0.2 | 1366 | 12.70 | 0.04 | 10 | 1366 | **4.17** | 0.04 | 0 | 1366 | 7.17 | **0.02** | 0 |
| c15 | 0.3 | 1249 | 4.19 | **0.04** | 0 | 1249 | **2.06** | **0.04** | 0 | 1249 | 5.20 | 0.05 | 126 |
| c16 | 0.1 | 949 | 541.38 | 0.31 | 5077 | 949 | **42.84** | 0.15 | 189 | 949 | 52.05 | **0.12** | 649 |
| c16 | 0.2 | 817 | 683.34 | 0.07 | 59 | 817 | **0.66** | **0.04** | 0 | 817 | 61.58 | **0.04** | 428 |
| c16 | 0.3 | 749 | 30.77 | 0.16 | 9 | 749 | **0.56** | 0.17 | 0 | 749 | 5.42 | **0.11** | 0 |
| c17 | 0.1 | 942 | 71.11 | 0.09 | 466 | 942 | **0.78** | **0.04** | 0 | 942 | 21.42 | **0.04** | 64 |
| c17 | 0.2 | 814 | 32.16 | 0.08 | 42 | 814 | **6.91** | **0.04** | 0 | 814 | 12.27 | **0.04** | 0 |
| c17 | 0.3 | 745 | 15.27 | 0.12 | 0 | 745 | **0.52** | 0.13 | 0 | 745 | 9.62 | **0.11** | 0 |
| c18 | 0.1 | 944 | TL | 0.12 | 17560 | 944 | **129.92** | **0.14** | 4061 | 944 | 776.19 | 0.21 | 43737 |
| c18 | 0.2 | 820 | TL | 0.16 | 63 | 819 | **1762.52** | **0.04** | 1115 | 820 | TL | 0.16 | 780 |
| c18 | 0.3 | 749 | 19.98 | 0.10 | 0 | 749 | **0.52** | **0.08** | 0 | 749 | 23.83 | **0.08** | 396 |
| c19 | 0.1 | 934 | 80.92 | 0.14 | 425 | 934 | 28.67 | 0.15 | 108 | 934 | **14.64** | **0.09** | 0 |
| c19 | 0.2 | 817 | 46.92 | 0.06 | 24 | 817 | **5.41** | **0.00** | 0 | 817 | 21.30 | **0.00** | 0 |
| c19 | 0.3 | 748 | 84.81 | 0.09 | 2960 | 748 | **14.52** | **0.05** | 69 | 748 | 77.22 | **0.05** | 371 |
| c20 | 0.1 | 939 | 27.72 | 0.13 | 0 | 939 | **2.89** | **0.08** | 0 | 939 | 9.59 | **0.08** | 0 |
| c20 | 0.2 | 819 | 37.00 | 0.03 | 4 | 819 | **8.16** | **0.00** | 0 | 819 | 9.20 | **0.00** | 0 |
| c20 | 0.3 | 754 | 11.14 | 0.04 | 0 | 754 | **0.53** | 0.10 | 0 | 754 | 3.98 | **0.03** | 0 |

z*: cost of the best found solution.
t(s): the runtime in seconds (with TL indicating that the time limit of 1800 seconds was reached).
g(%): the rootgap (if the instance is solved to optimality), or the optimality gap otherwise, both in percentage.
nodes: number of branch-and-bound nodes.

The gaps are very similar to all configurations, except for the larger cases (c16-20), where MIRANDA_ZHRP was not better in any instance. However, the probable reason why the gap of model proposed by (ALVAREZ-MIRANDA and SINNL, 2017) sometimes outperforms ours is due to the addition of $\{0,1/2\}$-cuts, as well as a stopping criterion (see Sections 4.1 and 4.2) that may stop the addition of cuts, preventing the linear relaxation from reaching its theoretical limit.

For instance c18 with b=0.2, a significant amount of time was spent on the separation and addition of cuts for integer solutions that violated the cut-set constraints. Since this issue occurred in only one instance, we did not investigate potential strategies (which are likely not difficult to devise) to address this occurred.

In Table 6.3 we analyze the results for EUCLIDEAN instances with $|V| \in \{100, 250\}$. For the instances e100, the distribution of cases in which each configuration was the fastest is well-balanced, with approximately one-third of the cases favoring each algorithm. Although, the total time spent to resolve all these instances were 162s for MIRANDA_ZHRP, 169s for PROPOSED_P, and 149s for PROPOSED_PD, with average times of 5.4s, 5.6s, and 5.0s, respectively. For the instances e250, PROPOSED_PD is significantly faster in all cases, while MIRANDA_ZHRP is the slowest, besides showing the worst gap, it also the highest number of nodes in the branch-and-bound tree. Our model in almost all cases solves the problem only at the root node.

Table 6.3: Results for the instance set EUCLIDEAN with |V| ∈ {100,250}.

| Instance | b | MIRANDA_ZHRP | | | | PROPOSED_P | | | | PROPOSED_PD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes |
| e100-1 | 0.1 | 973 | **3.02** | 0.05 | 6 | 973 | 3.03 | **0.03** | 0 | 973 | 4.36 | **0.03** | 8 |
| e100-1 | 0.2 | 908 | **3.12** | **0.10** | 0 | 908 | 4.33 | 0.13 | 3 | 908 | 3.48 | 0.13 | 5 |
| e100-1 | 0.3 | 867 | **3.00** | **0.10** | 0 | 867 | 8.25 | **0.10** | 47 | 867 | 3.22 | 0.14 | 0 |
| e100-2 | 0.1 | 1018 | 5.69 | 0.17 | 45 | 1018 | **4.95** | **0.05** | 0 | 1018 | 5.05 | 0.07 | 0 |
| e100-2 | 0.2 | 955 | 8.84 | 0.23 | 22 | 955 | 7.23 | 0.05 | 0 | 955 | **5.91** | **0.04** | 0 |
| e100-2 | 0.3 | 913 | **7.17** | 0.34 | 19 | 913 | 10.91 | **0.08** | 0 | 913 | 7.41 | 0.14 | 0 |
| e100-3 | 0.1 | 972 | 6.86 | **0.09** | 5 | 972 | 5.47 | 0.18 | 4 | 972 | **4.77** | 0.20 | 29 |
| e100-3 | 0.2 | 906 | 5.92 | **0.00** | 0 | 906 | **5.66** | 0.13 | 0 | 906 | 7.02 | **0.00** | 7 |
| e100-3 | 0.3 | 859 | 6.86 | 0.12 | 35 | 859 | 5.36 | **0.02** | 3 | 859 | **3.94** | **0.02** | 0 |
| e100-4 | 0.1 | 994 | **3.67** | **0.00** | 0 | 994 | 5.92 | **0.00** | 0 | 994 | 5.31 | **0.00** | 0 |
| e100-4 | 0.2 | 924 | **5.11** | 0.14 | 5 | 924 | 6.53 | **0.00** | 0 | 924 | 7.50 | **0.00** | 0 |
| e100-4 | 0.3 | 875 | 5.22 | 0.09 | 0 | 875 | 7.59 | **0.00** | 0 | 875 | **4.50** | **0.00** | 0 |
| e100-5 | 0.1 | 1021 | 10.09 | 0.22 | 331 | 1021 | 7.45 | 0.16 | 55 | 1021 | **6.98** | **0.13** | 161 |
| e100-5 | 0.2 | 945 | 5.02 | 0.08 | 9 | 945 | **4.62** | **0.00** | 3 | 945 | 4.97 | **0.00** | 16 |
| e100-5 | 0.3 | 895 | **4.05** | 0.17 | 18 | 895 | 4.92 | **0.06** | 0 | 895 | 4.38 | **0.06** | 11 |
| e100-6 | 0.1 | 1020 | 8.61 | **0.09** | 0 | 1020 | 6.62 | **0.09** | 0 | 1020 | **5.02** | **0.09** | 0 |
| e100-6 | 0.2 | 954 | 8.97 | 0.16 | 8 | 954 | **6.91** | **0.00** | 0 | 954 | 7.02 | **0.00** | 0 |
| e100-6 | 0.3 | 906 | 9.02 | 0.23 | 51 | 906 | **5.06** | **0.00** | 0 | 906 | 5.44 | **0.00** | 0 |
| e100-7 | 0.1 | 1025 | 3.80 | 0.03 | 0 | 1025 | **3.69** | **0.00** | 0 | 1025 | 5.38 | **0.00** | 0 |
| e100-7 | 0.2 | 955 | 4.98 | **0.00** | 30 | 955 | **4.50** | 0.05 | 6 | 955 | 4.55 | **0.00** | 0 |
| e100-7 | 0.3 | 907 | 4.95 | 0.10 | 0 | 907 | **4.56** | **0.06** | 0 | 907 | 5.00 | **0.06** | 4 |
| e100-8 | 0.1 | 1014 | **4.36** | **0.06** | 9 | 1014 | 5.75 | **0.06** | 11 | 1014 | 5.75 | **0.06** | 30 |
| e100-8 | 0.2 | 940 | 5.45 | 0.04 | 3 | 940 | 6.00 | **0.00** | 4 | 940 | **4.61** | **0.00** | 0 |
| e100-8 | 0.3 | 892 | 3.36 | 0.08 | 0 | 892 | 4.55 | **0.06** | 8 | 892 | **2.62** | **0.06** | 0 |
| e100-9 | 0.1 | 954 | 4.23 | 0.04 | 0 | 954 | 4.00 | **0.02** | 0 | 954 | **3.67** | **0.02** | 0 |
| e100-9 | 0.2 | 892 | 6.08 | **0.00** | 4 | 892 | **4.50** | **0.00** | 1 | 892 | 5.69 | **0.00** | 12 |
| e100-9 | 0.3 | 848 | 3.91 | **0.07** | 0 | 848 | 4.45 | **0.07** | 0 | 848 | **3.67** | **0.07** | 0 |
| e100-10 | 0.1 | 1016 | 2.80 | 0.04 | 0 | 1016 | 3.45 | **0.03** | 0 | 1016 | **2.75** | **0.03** | 0 |
| e100-10 | 0.2 | 954 | **3.97** | **0.06** | 0 | 954 | 6.09 | 0.08 | 0 | 954 | 3.98 | 0.07 | 0 |
| e100-10 | 0.3 | 907 | **3.66** | 0.10 | 0 | 907 | 6.16 | **0.02** | 0 | 907 | 4.58 | **0.02** | 0 |
| e250-1 | 0.1 | 1595 | 106.20 | 0.19 | 96 | 1595 | **76.64** | **0.00** | 0 | 1595 | 77.05 | **0.00** | 0 |
| e250-1 | 0.2 | 1482 | 672.56 | 0.46 | 5053 | 1482 | 268.67 | **0.07** | 0 | 1482 | **119.22** | 0.10 | 0 |
| e250-1 | 0.3 | 1399 | 238.16 | 0.28 | 535 | 1399 | 124.61 | **0.00** | 0 | 1399 | **117.16** | **0.00** | 0 |
| e250-2 | 0.1 | 1611 | 321.20 | 0.20 | 33 | 1611 | 208.11 | **0.00** | 0 | 1611 | **83.75** | **0.00** | 0 |
| e250-2 | 0.2 | 1503 | 249.42 | 0.27 | 307 | 1503 | 237.42 | **0.02** | 0 | 1503 | **144.56** | **0.02** | 0 |
| e250-2 | 0.3 | 1425 | 336.20 | 0.36 | 207 | 1425 | 375.27 | 0.04 | 1 | 1425 | **135.69** | **0.00** | 1 |
| e250-3 | 0.1 | 1613 | 278.55 | 0.27 | 345 | 1613 | 138.89 | **0.00** | 0 | 1613 | **61.03** | **0.00** | 0 |
| e250-3 | 0.2 | 1495 | 457.92 | 0.39 | 747 | 1495 | 336.17 | **0.05** | 0 | 1495 | **159.45** | **0.05** | 0 |
| e250-3 | 0.3 | 1413 | 461.34 | 0.36 | 1333 | 1413 | 195.97 | **0.04** | 0 | 1413 | **111.58** | 0.05 | 0 |
| e250-4 | 0.1 | 1656 | 311.95 | 0.28 | 142 | 1656 | 227.23 | 0.06 | 0 | 1656 | **104.23** | **0.05** | 0 |
| e250-4 | 0.2 | 1534 | 297.56 | 0.33 | 264 | 1534 | 134.00 | **0.04** | 0 | 1534 | **86.50** | 0.05 | 0 |
| e250-4 | 0.3 | 1454 | 224.17 | 0.23 | 86 | 1454 | 116.77 | **0.03** | 0 | 1454 | **54.50** | **0.03** | 0 |
| e250-5 | 0.1 | 1612 | 220.34 | 0.26 | 181 | 1612 | 100.42 | **0.00** | 0 | 1612 | **63.78** | **0.00** | 0 |
| e250-5 | 0.2 | 1493 | 208.86 | 0.30 | 195 | 1493 | 175.59 | **0.00** | 0 | 1493 | **62.52** | **0.00** | 0 |
| e250-5 | 0.3 | 1417 | 696.72 | 0.53 | 9350 | 1417 | 247.03 | **0.04** | 0 | 1417 | **84.45** | **0.04** | 0 |
| e250-6 | 0.1 | 1697 | 325.00 | 0.32 | 398 | 1697 | 147.62 | **0.00** | 0 | 1697 | **106.88** | **0.00** | 0 |
| e250-6 | 0.2 | 1572 | 363.14 | 0.06 | 0 | 1572 | 213.31 | **0.02** | 10 | 1572 | **106.38** | **0.02** | 9 |
| e250-6 | 0.3 | 1493 | 381.36 | 0.26 | 198 | 1493 | 163.78 | **0.03** | 0 | 1493 | **89.94** | **0.03** | 0 |
| e250-7 | 0.1 | 1661 | 198.31 | 0.14 | 17 | 1661 | 118.78 | **0.00** | 0 | 1661 | **53.48** | **0.00** | 0 |
| e250-7 | 0.2 | 1545 | 442.39 | 0.32 | 293 | 1545 | 181.66 | **0.00** | 0 | 1545 | **94.66** | 0.03 | 0 |
| e250-7 | 0.3 | 1463 | 254.73 | 0.28 | 209 | 1463 | 223.62 | **0.03** | 0 | 1463 | **161.58** | **0.03** | 0 |
| e250-8 | 0.1 | 1661 | 232.16 | 0.27 | 127 | 1661 | 143.41 | 0.03 | 0 | 1661 | **75.50** | **0.00** | 2 |
| e250-8 | 0.2 | 1540 | 194.94 | 0.20 | 77 | 1540 | 161.23 | **0.03** | 1 | 1540 | **73.27** | 0.04 | 0 |
| e250-8 | 0.3 | 1453 | 175.34 | 0.19 | 132 | 1453 | 100.34 | **0.03** | 0 | 1453 | **71.81** | **0.03** | 0 |
| e250-9 | 0.1 | 1631 | 209.31 | 0.28 | 88 | 1631 | 143.95 | **0.00** | 0 | 1631 | **65.88** | **0.00** | 0 |
| e250-9 | 0.2 | 1512 | 291.05 | 0.28 | 201 | 1512 | 110.77 | **0.00** | 0 | 1512 | **57.84** | **0.00** | 0 |
| e250-9 | 0.3 | 1430 | 340.36 | 0.37 | 396 | 1430 | 141.84 | **0.03** | 0 | 1430 | **57.75** | **0.03** | 1 |
| e250-10 | 0.1 | 1628 | 203.75 | 0.19 | 47 | 1628 | 164.27 | **0.00** | 0 | 1628 | **90.42** | **0.00** | 0 |
| e250-10 | 0.2 | 1516 | 265.45 | 0.30 | 183 | 1516 | 104.36 | **0.03** | 0 | 1516 | **72.62** | 0.05 | 0 |
| e250-10 | 0.3 | 1435 | 226.91 | 0.24 | 163 | 1435 | 108.31 | **0.02** | 0 | 1435 | **74.55** | 0.03 | 1 |

z*: cost of the best found solution.
t(s): the runtime in seconds (with TL indicating that the time limit of 1800 seconds was reached).
g(%): the rootgap (if the instance is solved to optimality), or the optimality gap otherwise, both in percentage.
nodes: number of branch-and-bound nodes.

We decided to analyze the instances EUCLIDEAN e500 separately, as they are the most challenging to solve among those considered in this work. These instances correspond to complete graphs with $|V| = 500$ and $|E| = 124750$. In Figure 6.3, we illustrate both the lower (bottom of each bar) and upper (top of each bar) bounds obtained for each configuration (MIRANDA_ZHRP, PROPOSED_P and PROPOSED_PD). Greater bars represent larger gaps, whereas smaller bars indicate lower gaps. Visually, one can observe that configuration MIRANDA_ZHRP exhibits the largest gaps, followed by PROPOSED_P and PROPOSED_PD. Additionally, for all cases, regarding the lower bounds, we have $PROPOSED\_PD > PROPOSED\_P > MIRANDA\_ZHRP$, while for the upper bounds, $PROPOSED\_PD < PROPOSED\_P < MIRANDA\_ZHRP$.

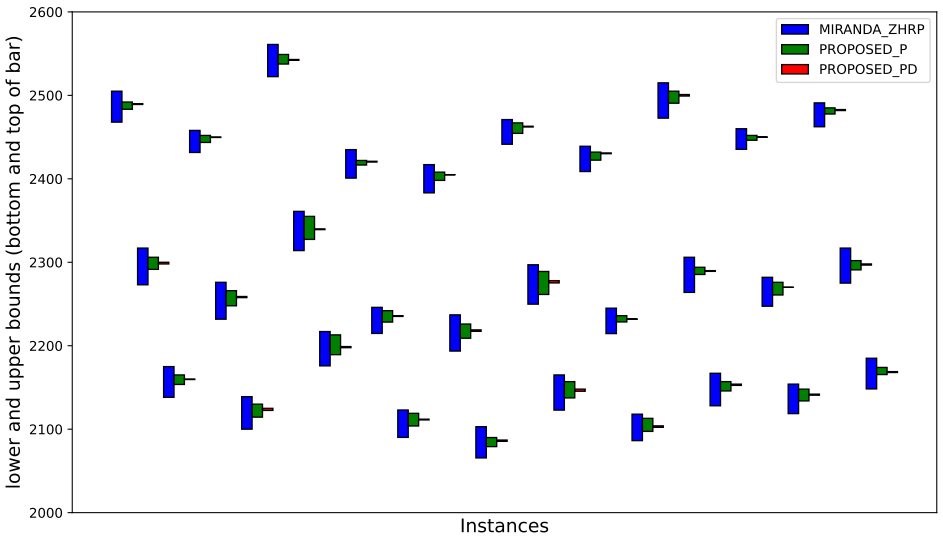



Figure 6.3: Lower and upper bounds for the EUCLIDEAN set with |V| = 500.

A more detailed analysis can be found in Table 6.4. As observed, only the PROPOSED_PD configuration was able to solve this type of instance (16 out of 30). In all cases where PROPOSED_PD achieved optimal solutions, the root gap was very small. In the cases where PROPOSED_PD was unable to reach an optimal solution, the optimality gap remained extremely close to zero, suggesting that with additional time, the

problem would likely be solved to optimality. This demonstrates that PROPOSED_PD is a promising alternative as the graph size increases.

Table 6.4: Results for the instance set EUCLIDEAN with |V| = 500.

| Instance | b | MIRANDA_ZHRP | | | | PROPOSED_P | | | | PROPOSED_PD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes | z* | t(s) | g(%) | nodes |
| e500-1 | 0.1 | 2505 | TL | 1.48 | 0 | 2492 | TL | 0.35 | 0 | **2490** | 1228.48 | **0.04** | 0 |
| e500-1 | 0.2 | 2317 | TL | 1.90 | 0 | 2306 | TL | 0.63 | 0 | **2300** | TL | **0.09** | 0 |
| e500-1 | 0.3 | 2175 | TL | 1.69 | 0 | 2165 | TL | 0.53 | 0 | **2160** | 1695.88 | **0.02** | 39 |
| e500-2 | 0.1 | 2458 | TL | 1.07 | 0 | 2452 | TL | 0.34 | 0 | **2450** | 1035.22 | **0.02** | 0 |
| e500-2 | 0.2 | 2276 | TL | 1.94 | 0 | 2266 | TL | 0.80 | 0 | **2259** | TL | **0.06** | 0 |
| e500-2 | 0.3 | 2139 | TL | 1.82 | 0 | 2130 | TL | 0.73 | 0 | **2125** | TL | **0.12** | 0 |
| e500-3 | 0.1 | 2561 | TL | 1.51 | 0 | 2549 | TL | 0.45 | 0 | **2543** | 1622.27 | **0.04** | 12 |
| e500-3 | 0.2 | 2361 | TL | 1.99 | 0 | 2355 | TL | 1.17 | 0 | **2340** | 1489.59 | **0.04** | 0 |
| e500-3 | 0.3 | 2217 | TL | 1.85 | 0 | 2213 | TL | 1.07 | 0 | **2199** | TL | **0.06** | 0 |
| e500-4 | 0.1 | 2435 | TL | 1.40 | 0 | 2422 | TL | 0.23 | 0 | **2421** | 1106.36 | **0.04** | 0 |
| e500-4 | 0.2 | 2246 | TL | 1.39 | 0 | 2242 | TL | 0.61 | 0 | **2236** | 1168.81 | **0.04** | 0 |
| e500-4 | 0.3 | 2123 | TL | 1.55 | 0 | 2119 | TL | 0.72 | 0 | **2112** | 1522.59 | **0.04** | 0 |
| e500-5 | 0.1 | 2417 | TL | 1.40 | 0 | 2408 | TL | 0.41 | 0 | **2405** | 1461.42 | **0.01** | 3 |
| e500-5 | 0.2 | 2237 | TL | 1.94 | 0 | 2226 | TL | 0.76 | 0 | **2219** | TL | **0.08** | 0 |
| e500-5 | 0.3 | 2103 | TL | 1.78 | 0 | 2090 | TL | 0.52 | 0 | **2087** | TL | **0.08** | 0 |
| e500-6 | 0.1 | 2471 | TL | 1.20 | 0 | 2467 | TL | 0.51 | 0 | **2463** | 1220.00 | **0.04** | 0 |
| e500-6 | 0.2 | 2297 | TL | 2.05 | 0 | 2289 | TL | 1.20 | 0 | **2278** | TL | **0.12** | 0 |
| e500-6 | 0.3 | 2165 | TL | 1.95 | 0 | 2157 | TL | 0.90 | 0 | **2148** | TL | **0.12** | 0 |
| e500-7 | 0.1 | 2439 | TL | 1.24 | 0 | 2432 | TL | 0.40 | 0 | **2431** | 1492.12 | **0.04** | 0 |
| e500-7 | 0.2 | 2245 | TL | 1.36 | 0 | 2236 | TL | 0.33 | 0 | **2232** | 1463.05 | **0.01** | 0 |
| e500-7 | 0.3 | 2118 | TL | 1.50 | 0 | 2113 | TL | 0.74 | 0 | **2104** | TL | **0.08** | 0 |
| e500-8 | 0.1 | 2515 | TL | 1.68 | 0 | 2505 | TL | 0.57 | 0 | **2501** | TL | **0.08** | 0 |
| e500-8 | 0.2 | 2306 | TL | 1.83 | 0 | 2294 | TL | 0.38 | 0 | **2290** | 1411.52 | **0.04** | 0 |
| e500-8 | 0.3 | 2167 | TL | 1.80 | 0 | 2157 | TL | 0.52 | 0 | **2154** | TL | **0.07** | 0 |
| e500-9 | 0.1 | 2460 | TL | 1.00 | 0 | 2452 | TL | 0.24 | 0 | **2450** | 870.22 | **0.01** | 0 |
| e500-9 | 0.2 | 2282 | TL | 1.52 | 0 | 2276 | TL | 0.67 | 0 | **2270** | 1750.00 | **0.00** | 0 |
| e500-9 | 0.3 | 2154 | TL | 1.64 | 0 | 2148 | TL | 0.66 | 0 | **2142** | TL | **0.06** | 0 |
| e500-10 | 0.1 | 2491 | TL | 1.15 | 0 | 2485 | TL | 0.30 | 0 | **2483** | TL | **0.05** | 0 |
| e500-10 | 0.2 | 2317 | TL | 1.81 | 0 | 2302 | TL | 0.48 | 0 | **2298** | TL | **0.06** | 0 |
| e500-10 | 0.3 | 2185 | TL | 1.68 | 0 | 2174 | TL | 0.39 | 0 | **2169** | 1334.59 | **0.05** | 3 |

z*: value of the best found solution.

t(s): the runtime in seconds (with TL indicating that the time limit of 1800 seconds was reached).

g(%): the rootgap (if the instance is solved to optimality), or the optimality gap otherwise, both in percentage.

nodes: number of branch-and-bound nodes.

## 6.4 Primal Heuristic Results

Figure 6.4 illustrates the effectiveness of the primal heuristic discussed in Section 3.3, with the vertical axis representing the percentage of cases and the horizontal axis showing the difference between the best solution value found by the heuristic and the optimal solution value. This analysis considers the C and EUCLIDEAN instances with $|V| \in \{100, 250\}$ for the PROPOSED_P and MIRANDA_ZHRP configurations. We chose to apply the heuristic only at the root node in each iteration of Branch-and-Cut.

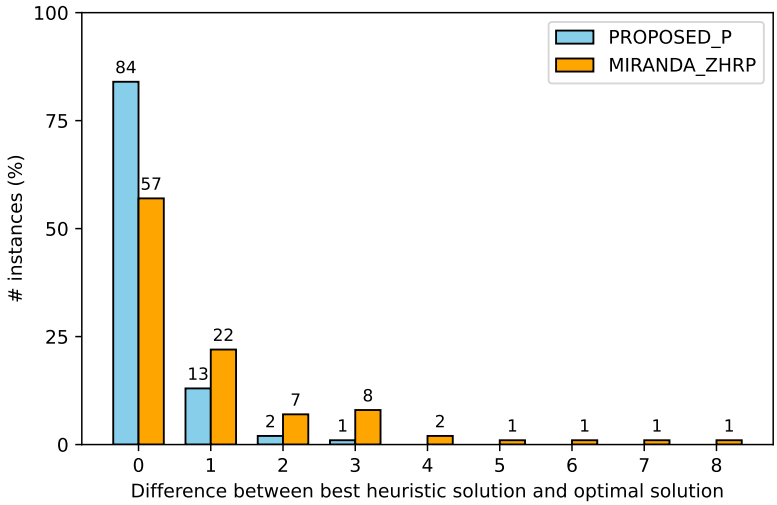



Figure 6.4: Comparison of primal heuristic solution quality.

The heuristic demonstrated high effectiveness in both models, with maximum differences of 3 and 8 for PROPOSED_P and MIRANDA_ZHRP, respectively. Can be noticed, in our model, the optimal solution was found in 84% of cases, while for MIRANDA_ZHRP only 57% of cases reached the optimal solution.

The result presented above can be explained by the fact that our model benefits from a tighter linear relaxation compared to the model in Section 3.1 (see Figure 6.1), which leads to higher-quality solutions being found by the heuristic. A strong linear relaxation is a crucial factor behind the performance of primal heuristics commonly used in optimization solvers, which use the solution from the linear relaxation to guide

the search for a feasible solution. The *Feasibility Pump* (FISCHETTI and SALVAGNIN, 2009), for example, performs iterative rounding based on the linear relaxation solution, ensuring feasibility by respecting the problem's constraints. Therefore, this type of heuristic is likely to perform better in our model than in that of (ALVAREZ-MIRANDA and SINNL, 2017), as discussed in the previous paragraph.

# Chapter 7

# Conclusions and Future Work

## 7.1 Conclusions

In this work, we have studied the Minimum Delay Upgrading Minimum Spanning Tree Problem (MDUMSTP), wich focuses on allocating limited resources to upgrade an existing network and minimize the total delay of the minimum spanning tree after these upgrades. This problem is a specific case of node-based upgrades, where upgrade a node incurs a cost but reduces the weights of all its connected edges. The problem was introduced in (KRUMKE *et al.*, 1999b), where the authors provided an approximation algorithm. In (ALVAREZ-MIRANDA and SINNL, 2017), the paper presents a Branch-and-Cut algorithm and a Lagrangian Relaxation approach.

To solve the MDUMSTP, we propose a new Integer linear Programming (IP) formulation and compared it with the exact model found in (ALVAREZ-MIRANDA and SINNL, 2017). By incorporating the separation and addition of valid inequalities, we enhance the formulation, leading to tighter bounds and improved computational performance. Moreover, we introduce a preprocessing procedure that significantly reduces the size of instances. Numerical experiments show that our Branch-and-Cut algorithm outperforms the Branch-and-Cut algorithm in (ALVAREZ-MIRANDA and SINNL, 2017)

for several test instances and, furthermore, solves to optimality previously unsolved instances. Finally, we show that applying primal heuristics to our model produces viable solutions of better quality than the model in (ALVAREZ-MIRANDA and SINNL, 2017).

## 7.2 Future Work

In order to expand on the current work and possibly improve its results, below we suggest some promising directions that could be explored as future research.

In the instances considered in this study, performing an upgrade on a node never increases the value of its adjacent edges. Generalizing, upgrades never reduce efficiency. However, this assumption does not always hold in real-world scenarios. For example, upgrading a signal repeater might improve its efficiency but cause interference with nearby repeaters, reducing their performance. Our model would solve these types of instances without any modification. However, the model proposed by (ALVAREZ-MIRANDA and SINNL, 2017) would require additional constraints, such as:

$$y^1_{ij} + y^1_{ji} + 2y^0_{ij} + 2y^0_{ji} \leq 2 - x_i - x_j, \qquad \forall (i,j) \in E. \tag{7.1}$$

A challenge introduced by this type of instance is that the preprocessing strategy proposed here would not work as currently defined (non-increasing edge weights), requiring adaptations to handle such cases.

Given the scalability challenges of the problem, a natural alternative is to decompose it into smaller and more manageable subproblems. Benders decomposition could be a promising approach in this regard (BNNOBRS, 1962, CODATO and FISCHETTI, 2006, GENDRON *et al.*, 2016, LAPORTE and LOUVEAUX, 1993, RAHMANIANI *et al.*, 2017). In (ÁLVAREZ-MIRANDA *et al.*, 2016), Benders decomposition has been applied to a node-based upgrade problem.

In addition to solving the MDUMSTP exactly through the branch-and-cut algorithm,

(ALVAREZ-MIRANDA and SINNL, 2017) also addressed it using Lagrangian relaxation, applying the Subgradient Method (HELD *et al.*, 1974) to solve the Lagrangian dual problem. As an alternative, a potentially more robust method for solving the Lagrangian dual problem could be the Revised Volume Algorithm (BAHIENSE *et al.*, 2002), which offers interesting dual convergence properties and a high-quality primal approximation that could be incorporated into our exact model, still enabling variable fixing to further reduce computational effort.

The heuristic used in this work proved to be very efficient in obtaining high-quality feasible solutions. But, as discussed in Section 6.4, its performance is heavily dependent on a LP-relaxation solution which provides a strong lower bound. For large-scale instances, this dependency can lead to high computational time. To address this, meta-heuristics that do not require solving LP-relaxation could be employed.

A point of attention identified in our computational experiments is the separation procedure of Branch-and-Cut algorithm in Sections 4.1 and 4.2. In this process, the max flow algorithm is repeatedly applied to each node, but it often fails to identify violated cuts, especially when the algorithm nears convergence, resulting in unnecessary processing. A more efficient alternative would be to employ algorithms capable of calculating the global minimum cut of a graph in a single step, such as the method proposed by (HAO and ORLIN, 1994). This approach has the potential to identify violated cuts, when existing, in a more direct way, significantly reducing the computational time required to separate the cuts.

Furthermore, a less rigorous stopping criterion for the addition of cuts at the root node could be adopted, halting this process earlier and moving to the branch-and-bound tree enumeration phase. Lastly, our model could be applied to other upgrade-based problems, as presented in Chapter 2.

# References


ALVAREZ-MIRANDA, E., SINNL, M., 2017, "Lagrangian and branch-and-cut approaches for upgrading spanning tree problems", *Computers & Operations Research*, v. 83, pp. 13–27.

ÁLVAREZ-MIRANDA, E., LUIPERSBECK, M., SINNL, M., 2016, "Optimal upgrading schemes for effective shortest paths in networks". In: *Integration of AI and OR Techniques in Constraint Programming: 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29-June 1, 2016, Proceedings 13*, pp. 406–420. Springer.

BAHIENSE, L., MACULAN, N., SAGASTIZÁBAL, C., 2002, "The volume algorithm revisited: relation with bundle methods", *Mathematical Programming*, v. 94, pp. 41–69.

BARBOSA, F., AGRA, A., DE SOUSA, A., 2021, "The minimum cost network upgrade problem with maximum robustness to multiple node failures", *Computers & Operations Research*, v. 136, pp. 105453.

BLANCO, V., MARÍN, A., 2019, "Upgrading nodes in tree-shaped hub location", *Computers & Operations Research*, v. 102, pp. 75–90.

BNNOBRS, J., 1962, "Partitioning procedures for solving mixed-variables programming problems", *Numer. Math*, v. 4, n. 1, pp. 238–252.

BORUVKA, O., 1926, "Contribution to the solution of a problem of economical construction of electrical networks", *Elektronickỳ Obzor*, v. 15, pp. 153–154.

CAPRARA, A., FISCHETTI, M., 1996, "{0, 1/2}-Chvátal-Gomory cuts", *Mathematical Programming*, v. 74, pp. 221–235.

CODATO, G., FISCHETTI, M., 2006, "Combinatorial Benders' cuts for mixed-integer linear programming", *Operations Research*, v. 54, n. 4, pp. 756–766.

CSARDI, G., NEPUSZ, T., 2006, "The igraph software", *Complex syst*, v. 1695, pp. 1–9.

DE ALMEIDA, A. M., MARTINS, P., DE SOUZA, M. C., 2012, "Min-degree constrained minimum spanning tree problem: complexity, properties, and formulations", *International Transactions in Operational Research*, v. 19, n. 3, pp. 323–352.

DIAS, S. S., SIMONETTI, L., "Results on Minimum-Weight Constrained Spanning Forests", *Available at SSRN 5023611*.

DRANGMEISTER, K. U., KRURNKE, S. O., MARATHE, M. V., et al., 1998, "Modifying edges of a network to obtain short subgraphs", *Theoretical Computer Science*, v. 203, n. 1, pp. 91–121.

FISCHETTI, M., SALVAGNIN, D., 2009, "Feasibility pump 2.0", *Mathematical Programming Computation*, v. 1, n. 2, pp. 201–222.

FLOYD, R. W., 1962, "Algorithm 97: shortest path", *Communications of the ACM*, v. 5, n. 6, pp. 345–345.

GENDRON, B., SCUTELLÀ, M. G., GARROPPO, R. G., et al., 2016, "A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks", *European Journal of Operational Research*, v. 255, n. 1, pp. 151–162.

GOEMANS, M. X., 1994, "The Steiner tree polytope and related polyhedra", *Mathematical programming*, v. 63, n. 1-3, pp. 157–182.

GOLDBERG, A. V., TARJAN, R. E., 1988, "A new approach to the maximum-flow problem", *Journal of the ACM (JACM)*, v. 35, n. 4, pp. 921–940.

GOMORY, R. E., 1958, "Outline of an algorithm for integer solutions to linear programs", *Bulletin of the American Mathematical Society*, v. 64, pp. 275–278.

GRAHAM, R. L., HELL, P., 1985, "On the history of the minimum spanning tree problem", *Annals of the History of Computing*, v. 7, n. 1, pp. 43–57.

HAMBRUSCH, S. E., TU, H.-Y., 1997, "Edge weight reduction problems in directed acyclic graphs", *Journal of Algorithms*, v. 24, n. 1, pp. 66–93.

HAO, J., ORLIN, J. B., 1994, "A faster algorithm for finding the minimum cut in a directed graph", *Journal of Algorithms*, v. 17, n. 3, pp. 424–446.

HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., et al., 2020, "Array programming with NumPy", *Nature*, v. 585, n. 7825, pp. 357–362.

HELD, M., WOLFE, P., CROWDER, H. P., 1974, "Validation of subgradient optimization", *Mathematical programming*, v. 6, pp. 62–88.

HO, J.-M., LEE, D., 1989, "Bounded diameter minimum spanning trees and related problems". In: *Proceedings of the fifth annual symposium on Computational geometry*, pp. 276–282.

KOCH, T., MARTIN, A., VOSS, S., 2001, "SteinLib: An updated library on Steiner tree problems in graphs", *Springer*.

KOSTER, A. M., ZYMOLKA, A., KUTSCHKA, M., 2009, "Algorithms to separate-chvátal-gomory cuts", *Algorithmica*, v. 55, n. 2, pp. 375–391.

KRUMKE, S., NOLTEMEIER, H., MARATHE, M., et al., 1996, *Improving Steiner trees of a network under multiple constraints*. Relatório técnico, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

KRUMKE, S. O., MARATHE, M. V., NOLTEMEIER, H., et al., 1998, "Approximation algorithms for certain network improvement problems", *Journal of Combinatorial Optimization*, v. 2, pp. 257–288.

KRUMKE, S. O., NOLTEMEIER, H., WIRTH, H.-C., et al., 1999a, "Improving spanning trees by upgrading nodes", *Theoretical Computer Science*, v. 221, n. 1-2, pp. 139–155.

KRUMKE, S. O., NOLTEMEIER, H., MARATHE, M. V., et al., 1997, "Modifying networks to obtain low cost trees". In: *Graph-Theoretic Concepts in Computer Science: 22nd International Workshop, WG'96 Cadenabbia, Italy, June 12–14, 1996 Proceedings 22*, pp. 293–307. Springer.

KRUMKE, S. O., MARATHE, M. V., NOLTEMEIER, H., et al., 1999b, "Improving minimum cost spanning trees by upgrading nodes", *Journal of Algorithms*, v. 33, n. 1, pp. 92–111.

KRUSKAL, J. B., 1956, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical society*, v. 7, n. 1, pp. 48–50.

LAM, S. K., PITROU, A., SEIBERT, S., 2015, "Numba: A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6.

LAPORTE, G., LOUVEAUX, F. V., 1993, "The integer L-shaped method for stochastic integer programs with complete recourse", *Operations research letters*, v. 13, n. 3, pp. 133–142.

LUCENA, A., RESENDE, M. G., 2004, "Strong lower bounds for the prize collecting Steiner problem in graphs", *Discrete Applied Mathematics*, v. 141, n. 1-3, pp. 277–294.

MARATHE, M. V., RAVI, R., SUNDARAM, R., et al., 1998, "Bicriteria network design problems", *Journal of algorithms*, v. 28, n. 1, pp. 142–171.

MEDYA, S., RANU, S., VACHERY, J., et al., 2018, "Noticeable network delay minimization via node upgrades", *Proceedings of the VLDB Endowment*, v. 11, n. 9, pp. 988–1001.

OBRUČA, A. K., 1968, "Spanning tree manipulation and the travelling salesman problem", *The Computer Journal*, v. 10, n. 4, pp. 374–377.

PAIK, D., SAHNI, S., 1995, "Network upgrading problems", *Networks*, v. 26, n. 1, pp. 45–58.

PRIM, R. C., 1957, "Shortest connection networks and some generalizations", *The Bell System Technical Journal*, v. 36, n. 6, pp. 1389–1401.

RAHMANIANI, R., CRAINIC, T. G., GENDREAU, M., et al., 2017, "The Benders decomposition algorithm: A literature review", *European Journal of Operational Research*, v. 259, n. 3, pp. 801–817.

VAN LEEUWEN, J., 1991, *Handbook of theoretical computer science (vol. A) algorithms and complexity*. Mit Press.

VANROSSUM, G., DRAKE, F. L., 2010, *The python language reference*, v. 561. Python Software Foundation Amsterdam, The Netherlands.