University of Aveiro Department of Mathematics,
2017

Olga Margarida Fajarda Oliveira

Descoberta da topologia de rede

Network topology discovery

Olga Margarida Fajarda Oliveira

Descoberta da topologia de rede

Network topology discovery

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Matemática, realizada sob a orientação científica da Doutora Maria Cristina Saraiva Requejo Agra, Professora Auxiliar do Departamento de Matemática da Universidade de Aveiro e do Doutor Bernard Fortz, Professor do Departamento de Informática da Universidade Livre de Bruxelas.

Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio, através de uma bolsa de doutoramento com referência SFRH/BD/6268/2011

To the loving memory of my father, Flávio Rodrigues de Oliveira, who has always supported me in all my endeavors

o júri

presidente

Professor Doutor José Carlos da Silva Neves
Professor Catedrático da Universidade de Aveiro

Professor Doutor Jorge Orestes Lasbarréres Cerdeira
Professor Catedrático da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Professor Doutor Domingos Moreira Cardoso
Professor Catedrático da Universidade de Aveiro

Professora Doutora Rosa Maria Videira Figueiredo
Professora Associada da Université d'Avignon et des Pays de Vaucluse, França

Professora Doutora Ana Maria Duarte Silva Alves Paias
Professora Auxiliar da Faculdade de Ciências da Universidade de Lisboa

Professora Doutora Maria Cristina Saraiva Requejo Agra
Professora Auxiliar da Universidade de Aveiro (orientadora)

agradecimentos

First I would like to thank my supervisors, Maria Cristina Saraiva Requejo Agra and Bernard Fortz, without whom this work would not have been possible. I thank their dedication, patience, understanding, teachings, suggestions and help.

I would like to thank my daughter, Ema, for her understanding, smiles and tenderness.

I would like to thank my boyfriend, Sílvio Gonçalves, for his understanding, support and incentive, giving me courage to overcome some difficulties.

I want to thank my parents and brother Dani for the support they have always given me despite being far. I also appreciate the support of my uncles, Maria do Carmo and Vitor Ramos and my cousin Vitor Miguel.

I want to thank my friends, André Belo, Ana Jordão, Maria Madalena Fonseca for their affection, friendship and support, which were of great importance throughout this work.

I want to thank CNRS researcher Michael Poss who helped me better understand robust optimization and collaborated in the development of the robust formulations presented in Chapter 7.

I want to thank my friend João Almeida who helped and collaborated on the creation of the software presented in Chapter 8. I also want to thank professor António Manuel Duarte Nogueira who supervised the creation of the software.

I want to thank and acknowledge the funding agency FCT (Fundação para a Ciência e a Tecnologia) who supported this research work with the grand reference SFRH/BD/6268/2011.

Finally, I want to thank everyone who somehow helped me complete this work.

palavras–chave

Problema da reconstrução de árvores de peso mínimo, Realização de árvores, Inferência da topologia da rede, Árvores filogenéticas, Programação linear inteira mista, Feasibility Pump, Local Branching, Otimização robusta.

resumo

A monitorização e avaliação do desempenho de uma rede são essenciais para detetar e resolver falhas no seu funcionamento. De modo a conseguir efetuar essa monitorização, é essencial conhecer a topologia da rede, que muitas vezes é desconhecida. Muitas das técnicas usadas para a descoberta da topologia requerem a cooperação de todos os dispositivos de rede, o que devido a questões e políticas de segurança é quase impossível de acontecer. Torna-se assim necessário utilizar técnicas que recolham, passivamente e sem a cooperação de dispositivos intermédios, informação que permita a inferência da topologia da rede. Isto pode ser feito recorrendo a técnicas de tomografia, que usam medições extremo-a-extremo, tais como o atraso sofrido pelos pacotes.

Nesta tese usamos métodos de programação linear inteira para resolver o problema de inferir uma topologia de rede usando apenas medições extremo-a-extremo. Apresentamos duas formulações compactas de programação linear inteira mista (MILP) para resolver o problema. Resultados computacionais mostraram que à medida que o número de dispositivos terminais cresce, o tempo que as duas formulações MILP compactas necessitam para resolver o problema, também cresce rapidamente. Consequentemente, elaborámos duas heurísticas com base nos métodos Feasibility Pump e Local Branching. Uma vez que as medidas de atraso têm erros associados, desenvolvemos duas abordagens robustas, um para controlar o número máximo de desvios e outra para reduzir o risco de custo alto. Criámos ainda um sistema que mede os atrasos de pacotes entre computadores de uma rede e apresenta a topologia dessa rede.

keywords

Minimum weighted tree reconstruction problem, Tree realization, Routing topology inference, Phylogenetic trees, Mixed integer linear programming, Feasibility Pump, Local Branch, Robust optimization.

abstract

Monitoring and evaluating the performance of a network is essential to detect and resolve network failures. In order to achieve this monitoring level, it is essential to know the topology of the network which is often unknown. Many of the techniques used to discover the topology require the cooperation of all network devices, which is almost impossible due to security and policy issues. It is therefore, necessary to use techniques that collect, passively and without the cooperation of intermediate devices, the necessary information to allow the inference of the network topology. This can be done using tomography techniques, which use end-to-end measurements, such as the packet delays.

In this thesis, we used some integer linear programming theory and methods to solve the problem of inferring a network topology using only end-to-end measurements. We present two compact mixed integer linear programming (MILP) formulations to solve the problem. Computational results showed that as the number of end-devices grows, the time need by the two compact MILP formulations to solve the problem also grows rapidly. Therefore, we elaborate two heuristics based on the Feasibility Pump and Local Branching method. Since the packet delay measurements have some errors associated, we developed two robust approaches, one to control the maximum number of deviations and the other to reduce the risk of high cost. We also created a system that measures the packet delays between computers on a network and displays the topology of that network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Monitoring and evaluating the performance of a network is essential to detect and solve network failures, in order to ensure high quality service levels that are required by some network-based services, like video conferencing and streaming multimedia. In order to achieve this monitoring standard it is essential to know the topology of the network, but due to the rapid, unregulated and decentralized development of communication networks, this topology is often unknown. Thus, efficient techniques are required for discovering the topology of a network.

Nowadays, several techniques can be used to infer a network topology, one can use Internet Control Message Protocol (ICMP) commands such as traceroute and the Simple Network Management Protocol (SNMP). However, these techniques require the cooperation of all internal network devices, which is frequently impossible, due to political and security issues.

Another way to infer the routing network topology is to use tomographic techniques. These techniques only use end-to-end network measurements, such as packet loss measurements or packet delay variance. Considering that packet loss is very low, nowadays, most of the used measurements, are packet delay measurements. The end-to-end network measurements can be obtained using multicast or unicast probing, although those obtained using multicast probing are more efficient and simple to obtain.

Our goal is to identify the topology of an unknown network, using only packet delay measurements between the end-devices of a network, which are obtained without the cooperation of the internal nodes.

This is an application of a more general problem, the Minimum Weighted Tree Reconstruction (MWTR) problem, that consists of finding a minimum length weighted tree connecting a set of terminal nodes in such a way that the length of the path between each pair of terminal nodes is greater than or equal to a given distance between the considered pair of terminal nodes. Beside the identification of the topology of an unknown network, the MWTR problem has applications in several other areas, namely, the inference of phylogenetic trees, that is a well studied problem.

The MWTR problem is a specific version of the distance realization problem (which is a graph realization problem), namely a tree realization problem for a distance matrix. Several authors studied the tree realization problem for a distance matrix and this class of combinatorial problems was proved to be NP-complete.

The inference of phylogenetic trees and the identification of an unknown network using only end-to-end measurements are two similar problems. Therefore tools from the phylogenetic area can be used to solve the problem of identifying a network topology.

1.1 Contribution

To the best of our knowledge, integer linear programming theory and methods were not yet applied to the specific problem of inferring a network topology.

In this work, we use integer linear programming theory and methods to infer a network topology. Some of these methods were also applied in the phylogenetic area. We contribute with two compact mixed integer linear programming (MILP) formulations of the MWTR problem.

As the number of end-devices grows, the time needed by the two compact MILP formulations to solve the problem also grows rapidly and therefore we elaborate two heuristics based on the methods Feasibility Pump and Local Branching.

The packet delay measurements have some errors associated and are therefore uncertain. To handle this uncertainty, we used two robust approaches, one to control the maximum number of deviation and the other to reduce the risk of high cost. From these two approaches we derived three formulations, the Robust-Deviation-Dual formulation, the Robust-Deviation formulation and the Robust-CVaR formulation.

Finally, to implement the several formulations we developed in a real scenario, we elaborate a system that displays the topology of an unknown network, using only packet delay measurements between the end-devices of the network. The system consists of two independent applications that work cooperatively. The first application synchronizes the devices, determines the delays and compiles these delays in a distance matrix. The second application determines the topology of the network using the distance matrix compiled and displays the topology in a graphical way.

1.2 Document structure

Besides the introduction, this report has eight more chapters. In Chapter 2, we present some preliminary definitions and results to better understand the problem, we analyze the graph and tree realization problem in more detail and define formally the MWTR problem.

Chapter 3 introduces the concepts related to the inference of a network topology. We present some of the existing techniques to discover a network topology. In this chapter, we also present the instances we generated using the network-level simulator NS-3.

To run our tests, we use, beside the data from the network simulation, data from the phylogenetics area. So, in Chapter 4, we describe some concepts related to the inference of the phylogenetic tree and present the instances of the phylogenetic area we used.

The two exact formulations, the Path-weight formulation and the Path-edges formulation, are presented in Chapter 5. In this chapter, we also present the computational results obtained by running the two formulations when using the data instances from networking application and phylogenetic application presented in Chapter 3 and Chapter 4, respectively.

Then, in Chapter 6 we present the methods Feasibility Pump and Local Branching. We also present the heuristics we developed applying the ideas of these methods to the Path-edges formulation. This chapter also contains the computational results obtained by running the two heuristics when using the data instances from networking application and phylogenetic application.

Subsequently, in Chapter 7 two robust approaches to solve the problem are studied, one to control the maximum number of deviations and the other to reduce the risk of high cost. In this chapter, we also present the three formulation, the Robust-Deviation-Dual formulation, the Robust-Deviation formulation and the Robust-CVaR formulation, we derived from these two approaches. We also present the computational results obtained by running the three formulations when using the data instances from networking application and phylogenetic application.

In Chapter 8, we present the software we developed and, finally, Chapter 9 presents our conclusions.

Chapter 2

The Minimum Weighted Tree Reconstruction (MWTR) problem

The Minimum Weighted Tree Reconstruction (MWTR) problem, defined in [65], is the problem of reconstructing a weighted tree by knowing only pairwise distances between a set of terminal nodes. This problem has applications in several areas such as psychology [40, 41, 43, 118] to represent cognitive processes or proximity and similarity relations; information security for the detection and recognition of documents duplications [45, 69]; telecommunications, namely in network tomography to discover the logical underlying network [32, 36, 47, 104] and the routing topology of a network [18, 32, 47, 76]; and in computational biology, to reconstruct phylogenetic trees [24, 60, 85]. All the applications have in common the notion of a *graph realization* of a distance matrix and more specific the notion of a tree realization of a distance matrix.

In this chapter, we start by presenting some preliminary definitions and results to better understand the problem. Subsequently, we analyze the graph and tree realization problem in more detail and define formally the MWTR problem.

2.1 Preliminaries

Let $G = (V, E, w)$ be a connected weighted graph, where V is the set of nodes, E is the set of edges and $w : E \to \mathbb{R}_0^+$ a function that assigns a non-negative weight to each edge of G.

2.1.1 Distance matrix

The input data we use to infer the tree are distances between terminal nodes. Therefore, we start by defining the following concepts: distance, distance between two nodes in a graph and distance matrix.

Definition 2.1. [128, 139] Let X be a set. A *distance* is a function d defined on $X \times X$ that satisfies the following four conditions:

- $\forall x, y \in X, \quad d(x, y) \geq 0$ (non-negativity);
- $\forall x, y \in X, \quad d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles);
- $\forall x, y \in X, \quad d(x, y) = d(y, x)$ (symmetry);
- $\forall x, y, z \in X, \quad d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

The function d is also called a *metric* and the couple (X, d) a *metric space*. If the set X is finite then (X, d) is a *finite metric space*.

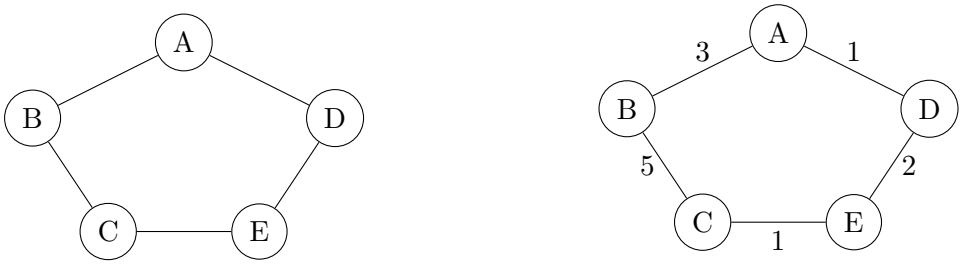
To simplify the notation, we denote the distance $d(x, y)$ as d_{xy}.

Definition 2.2. [128, 139] Let $G = (V, E)$ be a connected graph, where V is the set of nodes and E is the set of edges. The *distance between two nodes,* $i, j \in V$, in G, is the length of the shortest path in G between i and j and is denoted as d_{ij}^G. If the graph G is not weighted the length of the path corresponds to the number of edges in the path and if graph G is weighted the length of the path corresponds to the sum of the edge weight's along the path.

Example 2.1. Figure 2.1 presents two connected graphs: in (a) graph G, a not weighted graph, in (b) graph H, a weighted graph.

In graph G the shortest path between nodes A and C is the path $\{\{AB\},\{BC\}\}$ and $d^G_{AC} = 2$.

In graph H the shortest path between nodes A and C is the path $\{\{AD\},\{DE\},\{EC\}\}$ and $d^H_{AC} = 1 + 2 + 1 = 4$.



(a) Graph G. (b) Graph H.

Figure 2.1: Distances between two nodes in a connected graph.

Definition 2.3. Let $G(V, E)$ be a connected graph, where V is the set of nodes and E is the set of edges. The *diameter of* G is the greatest distance, in terms of number of edges, between any pair of nodes.

Example 2.2. The graph represented in Figure 2.2 has diameter 4. One of the paths with the greatest distance is represented in red.

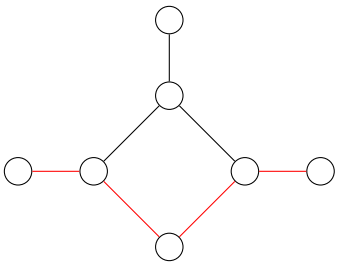


Figure 2.2: Diameter of a graph.

Definition 2.4. [129] An $n \times n$ matrix D with entries d_{ij} is called a *distance matrix* if it satisfies the following conditions:

- $d_{ij} \geq 0, \qquad \forall i, j \in \{1, 2, \ldots, n\};$
- $d_{ii} = 0, \qquad \forall i \in \{1, 2, \ldots, n\};$
- $d_{ij} = d_{ji}, \qquad \forall i, j \in \{1, 2, \ldots, n\};$
- $d_{ij} \leq d_{ik} + d_{kj}, \qquad \forall i, j, k \in \{1, 2, \ldots, n\}.$

Example 2.3. Figure 2.3 presents two matrices, matrix M_1 and matrix M_2. We can verify that matrix M_1 is a distance matrix. Matrix B is a non-distance matrix, once, for example, $d_{GH} > d_{GE} + d_{EH}$ and so the triangle inequality is not verified.

	A	B	C	D
A	0	1	2	4
B	1	0	3	3
C	2	3	0	3
D	4	3	3	0

(a) Matrix M_1.

	E	F	G	H
E	0	1	2	3
F	1	0	4	5
G	2	4	0	6
H	3	5	6	0

(b) Matrix M_2.

Figure 2.3: A distance matrix and a non-distance matrix.

A finite metric space, (X, d) can be described by a distance matrix [129], where the rows and columns of the distance matrix are indexed by the elements of X.

Example 2.4. The distance matrix M_1 of Example 2.3 describes the finite metric space, (X, d), where $X = \{A, B, C, D\}$ and the metric d is given by the entries of the distance matrix M_1. Since the matrix M_1 is a distance matrix its entries verify the four conditions: non-negativity, identity of indiscernibles, symmetry and triangle inequality.

Definition 2.5. The *order* of an $n \times n$ matrix is n.

Definition 2.6. A distance matrix is *additive* if it satisfies the following *four-point condition*:

$$d_{ij} + d_{k\ell} \leq \max\{d_{ik} + d_{j\ell}, d_{i\ell} + d_{jk}\}, \qquad \forall i, j, k, \ell \in V_t. \tag{2.1}$$

The four-point condition generalizes the triangle inequality by taking $k = \ell$.

The four-point condition is equivalent of saying that on any quartet i, j, k, ℓ we have that of the three sums $d_{ij} + d_{k\ell}$, $d_{ik} + d_{j\ell}$ and $d_{i\ell} + d_{jk}$, the largest two are equal.

Example 2.5. The distance matrix M_1 represented in Figure 2.3 is not additive, since $d_{AB} + d_{CD} = 1 + 3 = 4$, $d_{AC} + d_{BD} = 2 + 3 = 5$ and $d_{AD} + d_{BC} = 4 + 3 = 7$ and therefore $d_{AD} + d_{BC} > max\{d_{AB} + d_{CD}, d_{AC} + d_{BD}\}$.

The distance matrix represented in Figure 2.4 is additive: $d_{AB} + d_{CD} = 8 + 11 = 19$, $d_{AC} + d_{BD} = 7 + 14 = 21$ and $d_{AD} + d_{BC} = 12 + 9 = 21$.

	A	B	C	D
A	0	8	7	12
B	8	0	9	14
C	7	9	0	11
D	12	14	11	0

Figure 2.4: An additive distance matrix.

Definition 2.7. [129] Let D be an $n \times n$ matrix. An $m \times m$ *principal submatrix* ($m < n$) of matrix D is a matrix obtained from D by removing $n - m$ rows and the corresponding $n - m$ columns.

Example 2.6. In Figure 2.5 we have the 4×4 matrix M_1 from Example 2.3 in (a) and one of its 3×3 principal submatrix in (b). To obtain this principal submatrix the line and column C were removed.

	A	B	C	D
A	0	1	2	4
B	1	0	3	3
C	2	3	0	3
D	4	3	3	0

(a) Matrix M_1.

	A	B	D
A	0	1	4
B	1	0	3
D	4	3	0

(b) A principal submatrix of matrix M_1.

Figure 2.5: A matrix and on of its 3×3 principal submatrix.

2.1.2 Some properties of trees

Let $T = (V, E, w)$ be a weighted tree (a connected acyclic graph), where V is the set of nodes, E is the set of edges and $w : E \to \mathbb{R}_0^+$ a function that assigns a non-negative weight to each edge of T.

A tree can either be rooted or unrooted. A rooted tree is a tree with a hierarchical structure, with one special node, called the root, at the top and the other nodes branching down from it [141].

The set of nodes, V, can be divided in two subsets, $V = V_t \cup V_a$, the set of terminal nodes, V_t, nodes of degree one and the set of internal or additional nodes, V_a, nodes of degree at least two. If the tree is rooted, the root is an element of V_a. Also the set of edges, E, can be divided in two subsets, the set of terminal edges, E_t, and the set of internal or additional edges, E_a. The terminal edges connect terminal nodes to additional nodes and the additional edges connect two additional nodes.

Example 2.7. In Figure 2.6, we present a weighted tree, $T(V, E, w)$, with $V = \{1, 2, 3, 4, A, B, C, D, E, F, G\}$, $V_t = \{A, B, C, D, E, F, G\}$ and $V_a = \{1, 2, 3, 4\}$. Node 1 is the root of the tree and w_{ij} represents the weight of the edge $\{i, j\} \forall i, j \in V$.

A natural orientation can be assigned to a rooted tree, which goes from the root to the leafs. Considering such an orientation, we can define some concepts, such as parent node,

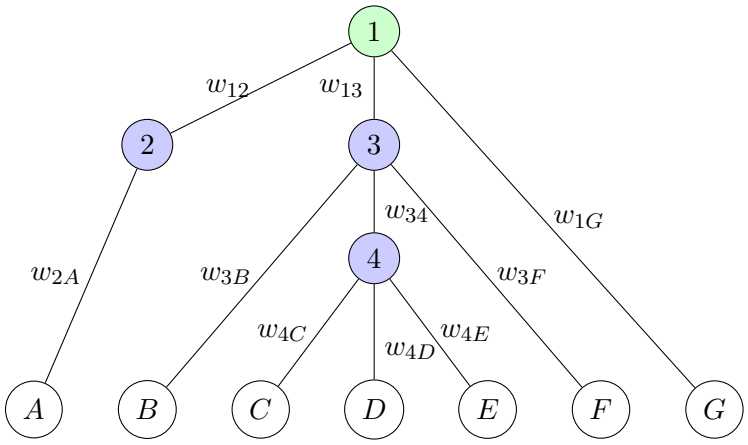


Figure 2.6: A weighted tree.

child node and sibling node.

Definition 2.8. Let $T = (V, E)$ be a rooted tree and $v \in V$ a node of the tree.

The *parent* of the node v is a node adjacent to it in the path to the root.

A *child* of the node v is a node of which v is the parent.

Two nodes are *siblings* if they have the same parent.

Example 2.8. Considering the tree of Figure 2.6 from Example 2.7. Node 3 is the parent of nodes $4, B$ and F, node A is a child of node 2 and nodes C and D are siblings.

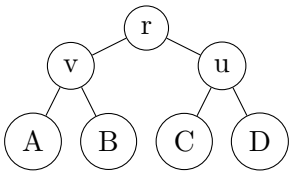
Definition 2.9. A tree $T' = (V', E')$ is a *subtree* of the tree $T = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

A subtree, of a tree $T = (V, E)$, rooted at an internal node $v \in V_a$, of T is denoted by T_v.

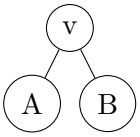
Definition 2.10. Let $T = (V, E)$ be a rooted tree and $v, u \in V$ two internal nodes of the tree. Two subtrees T_v and T_u are *sibling subtrees* if the nodes v and u are siblings.

Example 2.9. In Figure 2.7 we present a tree T in (a) and a subtree, T_v, of tree T in (b).

The subtrees T_v and T_u are sibling subtrees.



(a) Tree T.

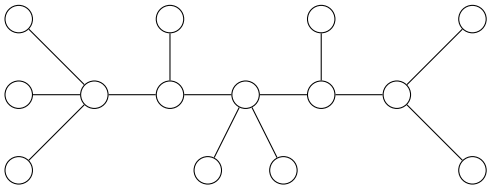


(b) Subtree, T_v, of T.

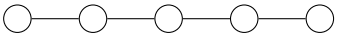
Figure 2.7: A tree T and a subtree, T_v, of T.

Definition 2.11. [67] A *caterpillar* is a tree such that, when all the terminal nodes are removed, becomes a path.

Example 2.10. In Figure 2.8 we have a caterpillar and the correspondent path obtained by removing the terminal nodes.



(a) Caterpillar



(b) Correspondent Path

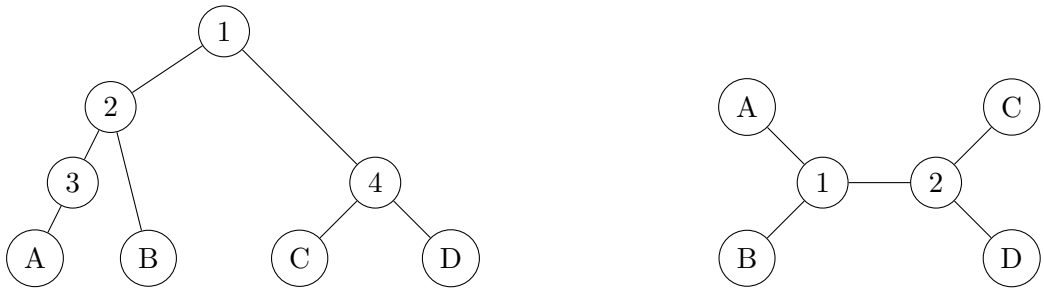
Figure 2.8: A Caterpillar.

Definition 2.12. A *binary tree* is a tree in which every node has at most degree three.

So the internal or additional nodes of a binary tree have degree two or three, the terminal nodes have degree one and, if the binary tree is rooted, the root has degree two.

Example 2.11. In Figure 2.9, we present two binary trees, a rooted and an unrooted binary tree.

In a tree, nodes with degree two can be eliminated by replacing the two edges incident by an only edge. If the tree is weighted the weight of the new edge is the sum of the two eliminated edges [128]. From now on, we only consider trees in which internal nodes have

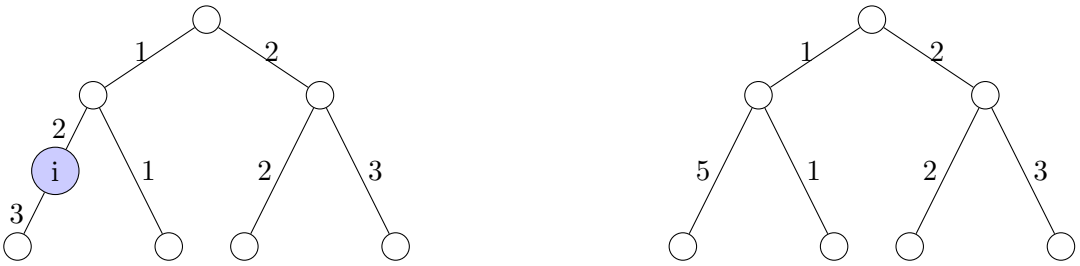


(a) Rooted binary tree. (b) Unrooted binary tree.

Figure 2.9: Binary trees.

degree at least three except the root (if the tree is rooted) which is the only node with degree two. So the internal nodes, except the root, of any binary tree have degree three.

Example 2.12. In Figure 2.10, (a) represents a tree in which node i has degree two and (b) represents the same tree after eliminating node i.



(a) Tree with node i of degree two. (b) Tree without node i.

Figure 2.10: Elimination of internal nodes of degree two.

Any tree can be converted into a binary tree, and vice-versa.

A binary tree can be transformed into a non-binary tree by eliminating nodes and edges in a similar way as we describe the elimination of internal nodes with degree two.

Byun and Yoo [21] present an algorithm to convert a tree into a binary tree by including dummy nodes and edges. If the tree is weighted the dummy edges have weight 0.

Example 2.13. In Figure 2.11, the tree in (a) has a node i with degree four and in (b) a

dummy node and a dummy edge (dashed) were include to transform the tree in a binary tree.

(a) Tree with node i of degree four. (b) Binary tree.

Figure 2.11: Transformation of a tree in a binary tree.

One of the advantages of binary trees is that, by knowing only the number of terminal nodes, we also know the number of internal nodes and the number of the edges of the tree.

In an unrooted binary tree with $|V_t| = n$, the set V_a has $(n-2)$ elements, V has $(2n-2)$ elements, E_a has $(n-3)$ elements and E has $(2n-3)$ elements.

In a rooted binary tree with $|V_t| = n$, the set V_a has $(n-1)$ elements, V has $(2n-1)$ elements, E_a has $(n-2)$ elements and E has $(2n-2)$ elements.

Any rooted tree can be converted into an unrooted tree, and vice-versa.

To eliminate the root of a rooted tree we proceed in the same way we did to eliminate internal nodes of degree two.

To transform an unrooted tree in a rooted tree, the root node can be added at the middle point of the longest path of the tree. If this point is in the edge $\{i, j\}$, remove edge $\{i, j\}$, add the dummy root node r and edges $\{i, r\}$ and $\{j, r\}$, one of the new edges inherits the weight of edge $\{i, j\}$ and the other new edge has weight 0. If this point is the node i, consider the edge $\{i, j\}$ connecting two nodes and proceed as before.

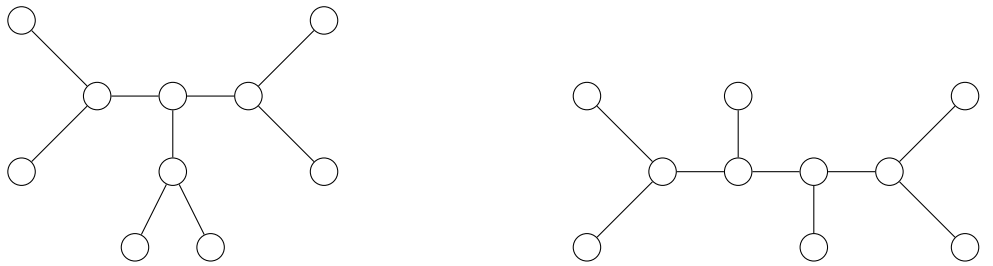
Example 2.14. In Figure 2.12, we transform an unrooted tree (a) in a rooted tree (b).

(a) Unrooted binary tree. (b) Rooted binary tree.

Figure 2.12: Transformation of an unrooted binary tree into a rooted binary tree.

Definition 2.13. Let $T = (V, E)$ be a tree with $|V| = n$. T is a *balanced binary tree* if it is a binary tree with minimal diameter.

Example 2.15. In Figure 2.13, (a) represents a balanced binary tree in which the diameter is equal to four and (b) represents an unbalanced binary tree in which the diameter is equal to five.



(a) Balanced binary tree. (b) Unbalanced binary tree.

Figure 2.13: Balanced and unbalanced binary trees.

Definition 2.14. Let $T = (V, E, w)$ be a weighted tree. The *tree length* of T is the sum of all weights of T, $\sum_{e \in E} w_e$.

Example 2.16. The tree in Figure 2.14 has tree length: $1 + 2 + 5 + 1 + 2 + 3 = 14$.

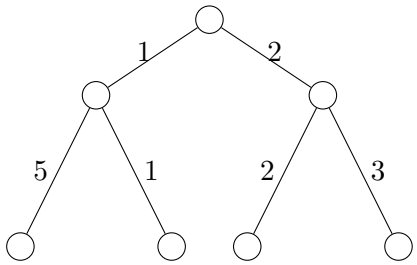


Figure 2.14: Weighted tree with tree length 14.

2.2 Graph realization problem

Hakimi and Yau [78] were the first to refer, in 1965, the graph realization problem of a distance matrix and presented an algorithm for the special case where the realization of the matrix is a tree.

Since then, over the years, several authors studied the characteristics of the distance matrix and its graph and tree realization [77, 87, 95, 126, 127, 129, 139, 140]. In this section, we present a few results about the graph and tree realization.

Definition 2.15. [32, 81] Given an $n \times n$ distance matrix D, representing the distances between a set of n objects, the *graph realization problem* is the problem of determining an edge-weighted connected graph $G = (V, E, w)$ with node set V, $|V| \geq n$, containing a subset $V_t \subseteq V$, with $|V_t| = n$ terminal nodes (each node representing an object), and the value d_{ij} in matrix D satisfying $d_{ij}^G = d_{ij}$, where d_{ij}^G denotes the length of the shortest path in G between terminal node i and terminal node j, $\forall i, j \in V_t$. If such a graph exists, then the distance matrix D has a *realization*. This means a metric given by the distance matrix D must be embedded into a graph.

Definition 2.16. If the graph which realizes D is a tree, we say that D has a *tree-realization* and the problem is known as the *tree realization problem*.

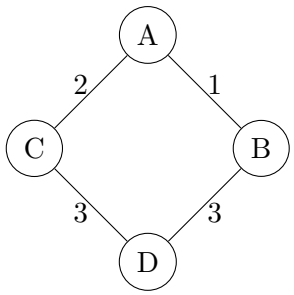
Every distance matrix D has a realization $G = (V, E, w)$. To construct a realization, it is sufficient to connect each pair of nodes $i, j \in V_t$ by an edge of weight d_{ij} and then delete

edges $\{i, j\} \in E$ verifying $d_{ij} = d_{ik} + d_{kj}$ for some $k \in V_t$ [142]. This realization has no auxiliary nodes, in other words, $V_t = V$.

Example 2.17. In Figure 2.15 we present the distance matrix M_1 from Example 2.3 and its graph realization.

	A	B	C	D
A	0	1	2	4
B	1	0	3	3
C	2	3	0	3
D	4	3	3	0

(a) Distance matrix.



(b) Graph realization.

Figure 2.15: A distance matrix and its graph realization.

Associated with this existence problem is the optimization problem that determines a graph such that the sum of all edge weights of G is minimized.

Definition 2.17. [32, 139] The realization $G = (V, E, w)$ having the minimum total length, $\sum_{e \in E} w_e$, among all the realizations of the distance matrix D, is *optimal*.

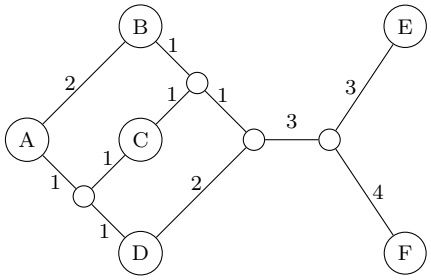
Dress [50] and Imrich et al. [86] proved that every distance matrix has an optimal realization. However, this realization may not be unique, as we can observe in Example 2.18.

Example 2.18. [80] In Figure 2.16 we present in (a) a distance matrix D and in (b) and (c) two optimal realizations of D of weight 20.

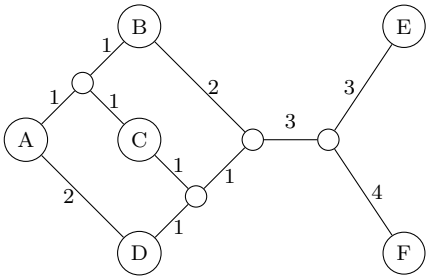
It is worth noting that, although a distance matrix has a graph realization, not all distance matrices describe trees, this is, are tree-realizable. Buneman [20] proved that a distance matrix of order four is tree realizable if and only if it satisfies the four-point condition (2.1).

	A	B	C	D	E	F
A	0	2	2	2	10	11
B	2	0	2	4	8	9
C	2	2	0	2	8	9
D	2	4	2	0	8	9
E	10	8	8	8	0	7
F	11	9	9	9	7	0

(a) Distance matrix D.



(b) Optimal realization of D.



(c) Optimal realization of D.

Figure 2.16: The distance matrix D and two optimal realizations.

Example 2.19. For a tree with 4 terminal nodes, i, j, k, ℓ (e.g. Figure 2.17), it can be checked that the four-point condition (2.1) holds

$$d_{ij} + d_{kl} \leq \max\{d_{ik} + d_{jl}, d_{il} + d_{jk}\} \Leftrightarrow$$

$$w_{i1} + w_{1j} + w_{k2} + w_{2\ell} \leq$$

$$\max\{w_{i1} + w_{12} + w_{2k} + w_{j1} + w_{12} + w_{2\ell},\ w_{i1} + w_{12} + w_{2\ell} + w_{j1} + w_{12} + w_{2k}\}$$

with w_{ij} being the weights associate to each edge in the tree solution.

Simões-Pereira [125] proved that an $n \times n$ distance matrix has a tree-realization if and only if all its principal submatrices of order four are tree-realizable.

Thus an $n \times n$ distance matrix has a tree-realization if and only if it is additive.

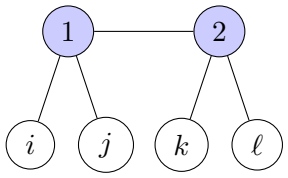


Figure 2.17: A possible topology for a tree with 4 terminal nodes.

Example 2.20. The distance matrix of Example 2.3 has no tree-realizations, once it is not additive as we refered in Example 2.5.

Hakimi and Yau [78] proved the following theorem:

Theorem 2.1. *If a distance matrix has a tree-realization, then this realization is optimal and unique.*

Several authors studied the tree realization problem for a distance matrix and this class of combinatorial problems was proved to be NP-complete [32, 42, 58]. Fiorini and Joret [61] and Catanzaro et al. [26] discuss the NP-hardness of two related optimization problems, the minimum evolution problem (MEP) and the balanced minimum evolution problem (BMEP). These two problems are very well known distance realization problems from the computational biology area and will be defined in Chapter 4.

The tree realization problem has application in several areas, as we stated before. In these areas the distance matrix is obtained using real data, that most of the time contains errors and therefore these distance matrix may not be additive. Consequently, most of the distance matrices used in the application areas have no tree-realization. On the other hand, computing optimal realizations is hard, even for a small number of terminal nodes [32]. Chung et al. [32] formulate several versions of the distance matrix realization problem, mention relevant results, discuss their algorithmic implications, present approximation results and several heuristics. Because the realization problem is hard even to approximate, Chung et al. [32] and Farach et al. [58] introduce a *weak realization* of a distance matrix.

Definition 2.18. An edge-weighted connected graph $G = (V, E, w)$ is a *weak realization* of a $n \times n$ distance matrix $D = (d_{ij})$, with rows and columns indexed by V_t, with $|V_t| = n$,

if the node set of G contains node set V_t, $V_t \subseteq V$, and the distance d_{ij}^G in the graph G between nodes i and j is greater than or equal to d_{ij}, $d_{ij}^G \geq d_{ij}$.

An optimal weak realization of a distance matrix D, i.e., the one that has the minimum total length, must be a tree [32]. Suppose the graph $G = (V_G, E_G, w^G)$, that is not a tree, is a weak realization of the distance matrix D. Once G is not a tree, it has cycles. By eliminating an edge in every cycle, we obtain a tree $T = (V_T, E_T, w^T)$ such that $V_T = V_G$ and $V_t \subseteq V_T$. Regarding the distance, in a cycle if the eliminated edge belongs to the shortest path between nodes i and j then $d_{ij}^T > d_{ij}^G$, if not then $d_{ij}^T = d_{ij}^G$ and therefore $d_{ij}^T \geq d_{ij}^G \geq d_{ij}$. Consequently, T is a weak realization of the distance matrix D and we have $\sum_{e \in E_T} w_e^T < \sum_{e \in E_G} w_e^G$.

Example 2.21. In Example 2.18 we presented two optimal realizations of matrix D with total length 20. This optimal realizations are also weak realizations of D. If we eliminate, in the optimal realization presented in Figure 2.16 (a), two edges we obtain the tree T represent in Figure 2.18 with total length 16. We verify that $d_{ij}^T \geq d_{ij}, \forall i, j \in \{A, B, C, D, E, F\}$. Consequently, T is also a weak realization of D and has total length lower than the total length of the weak realizations presented in Figure 2.16.

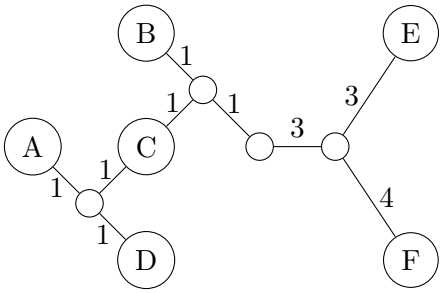


Figure 2.18: A weak realization of the matrix D presented in Example 2.18.

Since the weak realization is a tree, for a set of n terminal nodes, there can be at most $n - 2$ additional nodes without counting the root in case the tree is rooted. This can be proven by induction. The affirmation is trivially true for $n = 0$. We assume that the

affirmation is true for $n \geq 1$. Suppose we have a tree with n terminal nodes and $n-2$ additional nodes, we can add a terminal node by connecting it to an existing additional node and the tree obtained has $n+1$ terminal nodes and $n-2$ additional nodes, or we can connect the terminal node to a new additional node that will connect to an existing additional node, obtaining a tree with $n+1$ terminal nodes and $n-1$ additional nodes. Consequently, a tree with $n+1$ nodes has at most $n-1$ additional nodes.

Even when restricting the topologies to binary trees in which every internal node, except the root, has degree three, there are, for n terminal nodes, in case of rooted trees $(2n-3)!! = \frac{(2n-3)!}{2^{n-2}(n-2)!}$ different possible topologies and in the case of unrooted trees $(2n-5)!! = \frac{(2n-5)!}{2^{n-3}(n-3)!}$ different possible topologies [30, 102]. The Example 2.22 exemplifies the case for $n = 4$.

Example 2.22. Figure 2.19 displays in (A) the 15 possibilities for the topologies of rooted trees with 4 terminal nodes and in (B) the 3 possibilities for the topologies of unrooted trees with 4 terminal nodes.

There are finitely many topologies for a weak realization of D, and for each topology the problem of determining a weak realization can be formulated as a linear programming problem. To the best of our knowledge Cantanzaro et al. [26, 27] are the only authors to present (mixed integer) linear programming models for solving the problem.

2.3 Definition of the MWTR problem

A specific version of a tree realization problem is the Minimum Weighted Tree Reconstruction (MWTR) problem, which is defined as follows:

Definition 2.19. Given a set V_t of n terminal nodes and an $n \times n$ distance matrix D, whose entries represent the distances between the n terminal nodes, find a weighted tree $T = (V, E, w)$ spanning $V = V_t \cup V_a$, where V_a is a subset of $n-2$ additional nodes, that

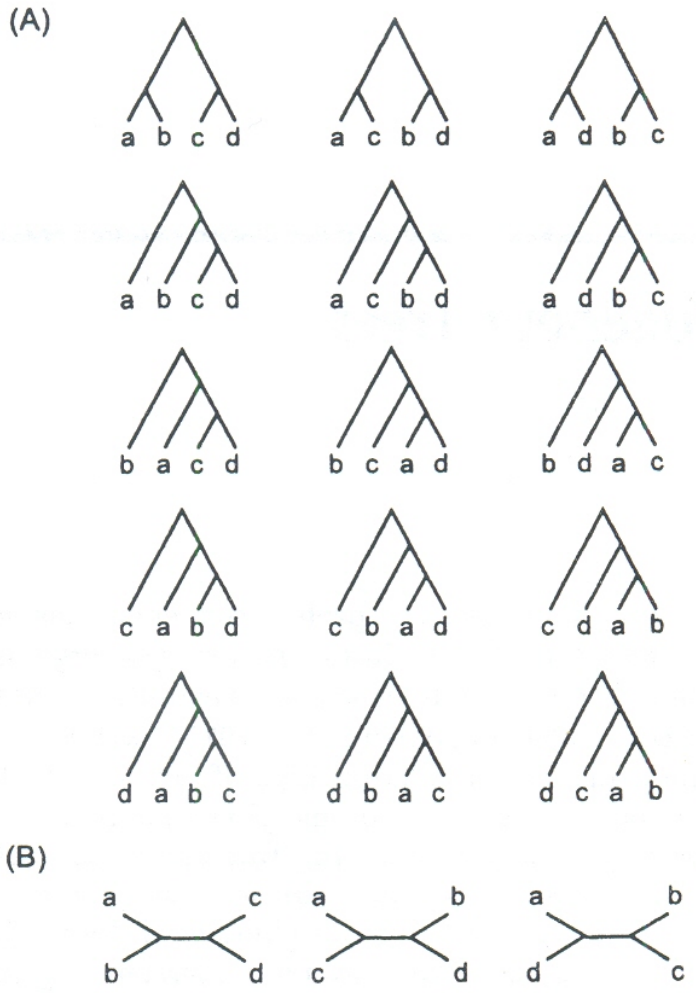


Figure 2.19: Possible topologies of trees with 4 terminal nodes[102].

solves the problem:

$$\min \sum_{e \in E} w_e$$
$$s.t. \sum_{e \in P_{ij}} w_e \geq d_{ij}, \qquad \forall i, j \in V_t : i \neq j$$
$$w_e \geq 0 \qquad \forall e \in E$$

where P_{ij} is the unique path between any two terminal nodes i and $j \in V_t$

The constructed trees can be either rooted or unrooted. We consider unrooted trees and all the contents in this work can be adapted to a rooted tree. Therefore, and for modeling

purposes, we force the tree structure to be an unrooted binary tree, all the additional nodes to have degree exactly three and the terminal nodes to be the leaves. Restricting ourselves to such topologies will help us deal with this combinatorial problem which is a weak realization problem. Enforcing this tree structure is not a restriction to the problem as any tree can be converted into a rooted binary tree, and vice-versa, as we showed in Section 2.1.

To simplify tree length computations, Pauplin [107] developed a method to directly calculate the sum of all weights of an urooted tree (the *tree length*) without having to explicitly determine its edge-weights. According to this method the tree length is given by

$$\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} \tag{2.2}$$

where z_{ij} indicates the number of edges in the path P_{ij} between terminal nodes i and j. When $d_{ij} = \sum_{e \in P_{ij}} w_e$, we have

$$\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} = \sum_{i,j \in V_t} \sum_{e \in P_{ij}} w_e\ 2^{-z_{ij}}.$$

As the weight w_e appears as many times as the number of paths P_{ij} to which the edge e belongs to, it holds

$$\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} = \sum_{e \in E} w_e$$

and for a weak realization we have

$$\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} \leq \sum_{e \in E} w_e$$

and

$$\min \sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} \leq \min \sum_{e \in E} w_e$$

.

Example 2.23 illustrates the fact that $\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} = \sum_{e \in E} w_e$, when $n = 4$.

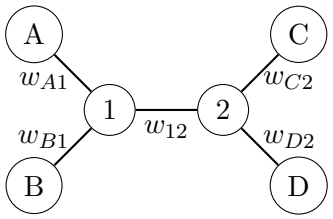


Figure 2.20: Tree with four external nodes.

Example 2.23. Figure 2.20 represents a tree with four terminal nodes, A, B, C and D and two additional nodes 1 and 2.

The distance matrix is symmetric, thus

- $d_{ij} = d_{ji}$ and $z_{ij} = z_{ji}$

The values z_{ij} indicate the number of edges in the path between node i and node j.
We have

- $z_{AB} = z_{CD} = 2$
- $z_{AC} = z_{AD} = z_{BC} = z_{BD} = 3$

The value w_{ij} represents the weight of the edge between node i and node j, thus

- $d_{AB} = w_{1A} + w_{1B}$
- $d_{AC} = w_{1A} + w_{12} + w_{2C}$
- $d_{AD} = w_{1A} + w_{12} + w_{2D}$
- $d_{BC} = w_{1B} + w_{12} + w_{2C}$
- $d_{BD} = w_{1B} + w_{12} + w_{2D}$
- $d_{56} = w_{25} + w_{26}$

And so we obtain:

$$\begin{aligned}\sum_{i,j\in V_{ext},i\neq j} d_{ij}\, 2^{-z_{ij}} &= 2[2^{-2}(d_{AB}+d_{CD})+2^{-3}(d_{AC}+d_{AD}+d_{BC}+d_{BD})]\\ &= 2^{-1}(w_{1A}+w_{1B}+w_{2C}+w_{2D})+\\ &\quad +2^{-2}(2w_{1A}+2w_{1B}+4w_{12}+2w_{2C}+2w_{2D})\\ &= w_{12}+w_{1A}+w_{1B}+w_{2C}+w_{2D}\\ &= \sum_{e\in E} w_e\end{aligned}$$

We can replace the objective function of the MWTR problem, $\sum_{e\in E} w_e$, by the objective function $\sum_{i,j\in V_t} d_{ij}\, 2^{-z_{ij}}$, obtaining an alternative way to infer the weight tree.

Chapter 3

Network topology

In this chapter we present an application of the MWTR in telecommunications: the inference of an underlying network. We describe some existing methods to infer a network topology and describe how we obtained the instances we used to run ours tests.

A network is a number of interconnected devices [1]. The communication and computer networks have evolved very rapidly in a largely unregulated and open environment from a small controlled network serving only a few users to the immense collection of heterogeneous interconnected terminals, routers and other platforms.

Example 3.1. In Figure 3.1 we present an example of a computer network. The Internet is represented as a cloud because its topology is unknown.

Monitoring and assessing the network is essential to achieve high quality service levels. The task of network monitoring gives rise to performance problems identification involving link failures, delays, connectivity, and traffic flow. To do the monitoring, network performance parameters, such as traffic rates, link delays and packet loss rate, are needed. However, the decentralized nature of the communication and computer networks makes assessment of network performance very difficult, once the internal devices do not freely transmit them. One way to overcome this difficulty is to use end-to-end network measurements, such as packet loss measurements or packet delay between a sender and a receiver,

[1] in Oxford English Dictionary

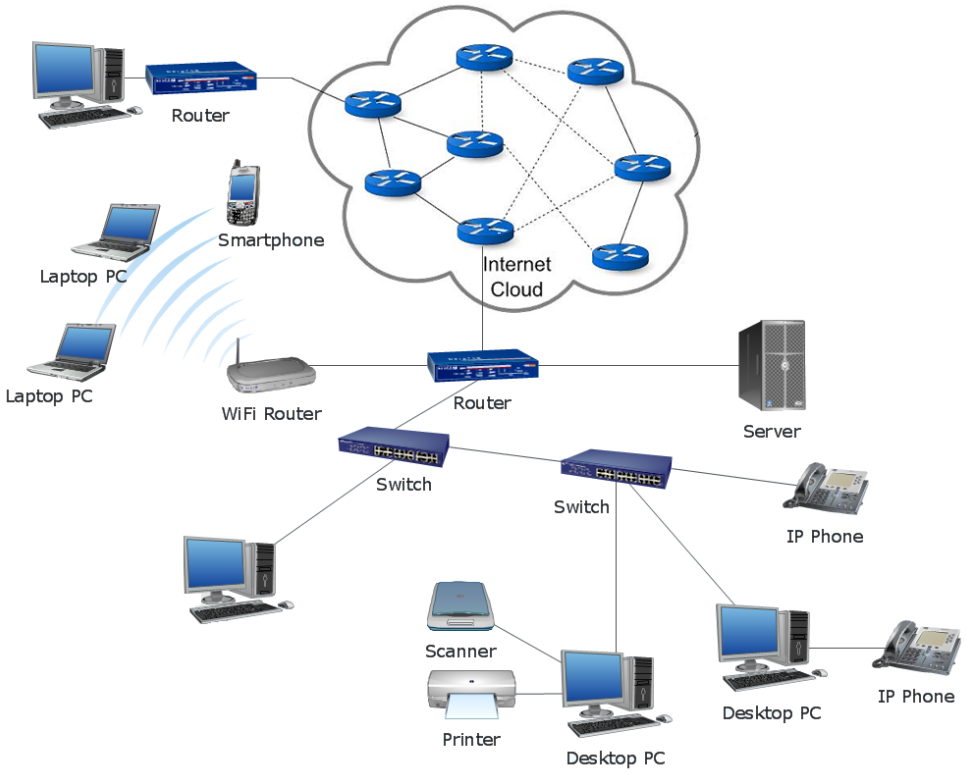


Figure 3.1: A computer network [5].

which have little or no impact on the network load. Packet loss measurements are obtained by counting the packets transmitted and received between nodes. These measurements do not need the clocks of the several end-hosts to be synchronized. Delays measurements are obtained by measuring the difference between the time of the packet sent and the packet reception. The delays on the links have two origins: propagation delays and router processing delays. The time delay between the sent and the reception is the sum of the link delays in the path [35]. These measurements generally require clock synchronization, that is, the clocks of the several end-hosts must be synchronized before making the measurements.

The problem of deducing the internal network characteristics from end-to-end measurements is called *network tomography* and is a field in the statistic area [34]. Vardi [138] was one of the first to study the problem of network performance parameters estimation based on traffic measurements and coined the term "network tomography" due to the similarity between network inference and medical tomography. Coast et al. [35] introduced formally

the field of network tomography and Castro et al. [22] survey this field.

The methods used in network tomography to estimate network performance parameters using end-to-end measurements assume that the underlying topology of the network is known, but most of the time it is unknown. So the first step in network tomography is to infer the underlying network topology.

3.1 Methods to infer a network topology

Nowadays several techniques can be used to infer a network topology. One can use the networking tool *traceroute* [46, 49, 75, 124] that sends several Internet Control Message Protocol (ICMP) [2] packets, with gradually increasing *time-to-live* (TTL). The TTL is decremented each time the packet reaches a router and when the TTL is zero the router drops the packet and sends an ICMP *time-exceeded* message to the source of the packet. When a packet reaches the destination, the destination is supposed to reply with an ICMP *destination unreachable* message with the code *port unreachable*. In order for the destination to sent this type of message, the UDP (User Datagram Protocol) packet must specify a high, unused, port number. The messages sent by the routers have the IP address of the sender and so a list of the routers traversed until reaching the destination can be constructed. To avoid waiting too long for a response, the sender also has a timer which indicates when the sender should send another packet.

Example 3.2. Figure 3.2 illustrates how the networking tool *traceroute* works. The Monitor sends ICMP packets with gradually increasing TTL starting with one until it receives a message from Destination. The TTL of the first packet sent is equal to one. The first router in the path from the Monitor to the destination to receive the packet is router R1. As router R1 receives the packet it decrements the TTL of the packet and drops the packet once the TTL is equal to zero. The router R1 then sends a ICMP *time-exceeded* message (ICMP_TE) to the Monitor. The Monitor then sends a second packet with TTL equal

[2]The protocol ICMP is used by intermediate devices and host to communicate error informations and updates to other devices.

to two. The TTL of this packet is decremented by router R1 which forwards the packet to router R2. Router R2 decrements the TTL, drops the packet and sends a ICMP *time-exceeded* message to the Monitor. The Monitor proceeds this way until a packet reaches Destination which replays with a ICMP *destination unreachable* (ICMP_DU) message.

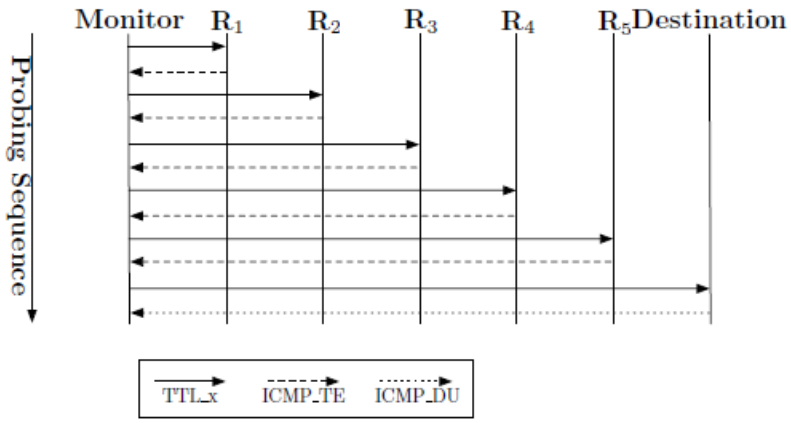


Figure 3.2: A *traceroute* example [46].

Unfortunately, due to political and security issues, not all routers respond to *traceroute* requests and so this tool may not discover all the network.

Another tool used to discover the network topology is the Simple Network Management Protocol (SNMP) [133]. SNMP is a protocol for a management station to exchange information with a number of agents. An agent is a device that has SNMP agent software installed. This software collects information about the device and responds to queries sent from the management station. The management station is a device from which the network administrator can manage the agents in the network, sending query requests to gather information and modifying predefined values.

Several authors developed techniques to discover the network topology based on SNMP [19, 105, 124]. Routers have routing tables with a list of their neighbors. A topology can be discovered by using recursively the information of the routing tables. The management station sends a query request to obtain the information of the routing table, to an

agent. With this information the neighbors of the agent can be identified and the management station can then send query requests to all the neighbors to obtain their routing table informations.

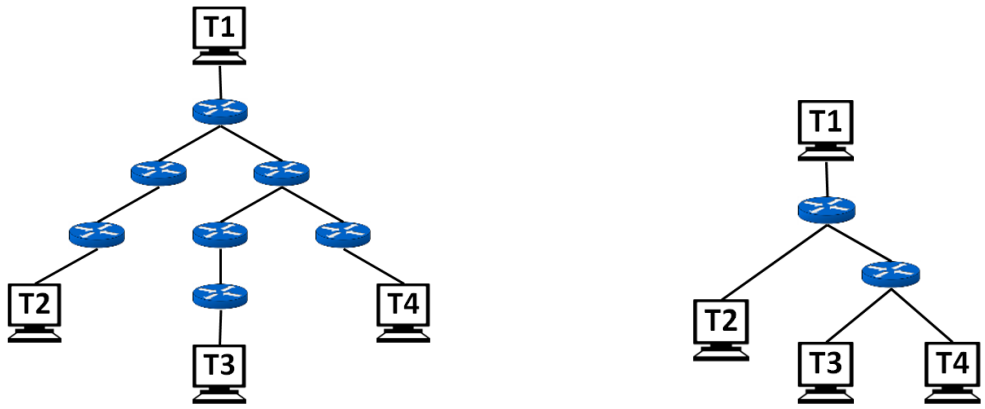
Siamwalla et al. [124] showed that SNMP performs better than other techniques. Unfortunately, SNMP is not implemented in all devices. Furthermore, SNMP can be turned off in the devices in which it is implemented and nowadays due to security issues, most of the network administrator turns it off.

The two methods we just summarized are the most commonly used, but besides these there are others. Haddadi et al. [76], Donnet et al. [48] and Motamedi et al. [100] wrote survey papers where they presented several mechanisms for discovering the internet topology. All the mechanisms they presented require the cooperation of all the internal network devices.

To overcome the fact that many internal network devices do not cooperate, the topology can be inferred using end-to-end traffic measurements. With this limited information the physical topology can not be inferred. The physical topology is the physical structure of the network, that is how the devices are physically interconnected. To avoid failures most of the networks have redundant infrastructures installed. End-to-end traffic measurements only permit to infer the network topology which is traversed by the data sent from one device to another. This network topology is a tree, once it is assumed that the routes traversed by the packets are fixed during the measurements. Furthermore, only the physical devices on the network where traffic branching occurs can be inferred, that is only the logical topology can be inferred. The logical topology is obtained from the physical topology, representing only the physical devices on the network where traffic branching occurs and joining all the connections between these devices by a single logical link, which comprises one or more physical links [52].

Example 3.3. Figure 3.3 displays in (a) a physical routing topology and in (b) the associated logical topology.

A network can be represented by a graph where nodes represent the physical devices



(a) The physical routing topology. (b) The logical topology.

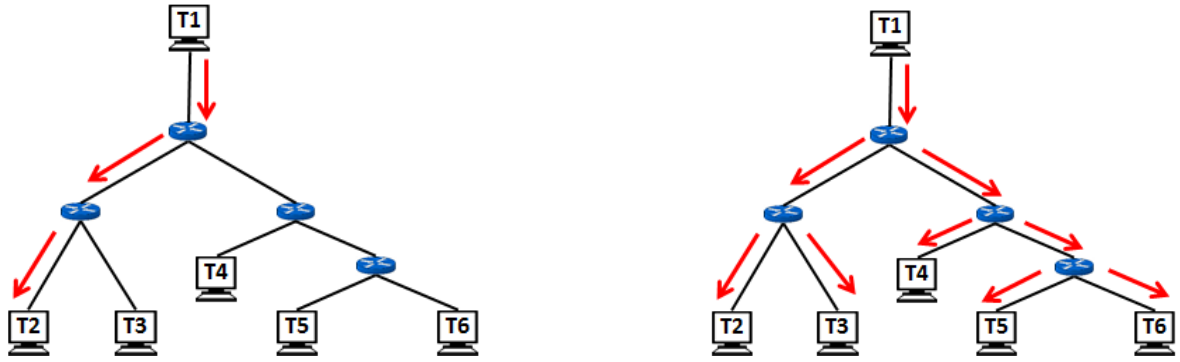
Figure 3.3: A physical routing topology and the associated logical topology.

and the edges the links connecting two devices. So the problem consists in inferring the logical topology, which is a tree, using only measurements at pairs of receivers. This is an application of the MWTR problem presented in Chapter 2. In this application, the terminal nodes represent the receivers, internal nodes represent physical devices where traffic branching occurs, edges represent relationships between pair of nodes and edges weights represent the quantification of some evolving connection property, like the delay or packet losses, for which end-to-end network measurements can be obtained.

The end-to-end network measurements can be obtained using multicast [51, 52, 103, 104, 113] or unicast probing [34, 57, 121], although those obtained using multicast probing are more efficient and simple to obtain. In an unicast transmission a source sends a packet to a single receiver. In multicast, a packet is sent to multiple receivers using a single transmission. To use multicast transmission the source and the receivers must be in the same multicast group. As a packet sent from the source with the multicast group address reaches a branching point, it is replicated and a replica is sent to every branch. So every receiver gets a copy of the packet.

Example 3.4. Figure 3.4 displays in (a) a scheme of a unicast transmission and in (b) a scheme of a multicast transmission. In scheme (a) the source $T1$ sends a packet to

the receiver $T2$. In scheme (b) the source $T1$ sends a multicast packet. The receivers $T2, T3, T4, T5$ and $T6$ belong to the same multicast group as the sender, so every router, along the way from $T1$ to the receivers, replicates the packet and sends a copy to every branch.



(a) Unicast transmission. (b) Multicast transmission.

Figure 3.4: An unicast and a multicast transmission.

In a multicast transmission, the packets reaching a pair of receivers face the same conditions from the source to the closest common branch point between the pair of receivers [51].

Unfortunately, multicast is not implemented in every device and therefore sometimes the end-to-end measurements must be obtained using unicast transmissions. Coates and Nowak [37] proposed a technique to make unicast loss measurements using back-to-back packet pairs. Shih and Hero [121] applied the same technique to make unicast delay measurements. A back-to-back packet pair is composed of two packets destined to two different receivers and sent one after the other by the sender. It has been verified experimentally in real networks [54, 109] that when a back-to-back packet pair is sent across a common link, the packets of the back-to-back packet pair will have practically the same delays through shared links and if the first packet is successfully transmitted the probability of the second to be successfully transmitted is very high. So, like in multicast transmission, the back-to-back packet pair face practically the same conditions from the source to the

closest common branch point between the pair of receivers.

Example 3.5. Figure 3.5 displays an example of a back-to-back packet pair transmission. The terminal node T1 sends a back-to-back packet pair, A and B. The packet A has destination node T5 and packet B has destination node T6. The packet pair share the same path until they arrive at router R1.

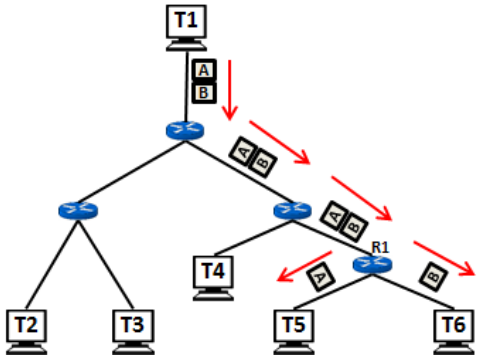


Figure 3.5: Back-to-back packet pair transmission.

Delay measurements generally require clock synchronization. This synchronization can be done using several protocols, such as Network Time Protocol (NTP) [99] or Precision Time Protocol (PTP) [83]. The Precision Time Protocol is described in Section 8.4. Most of these protocols are quite complicated to implement and the accuracy of the synchronization can be small. Therefore, to overcome the clock synchronization limitation Coast et al. [34] developed a delay-based measurement using unicast probes and entitled it the sandwich probes. A sandwich probe is composed of two small packets separated by a larger packet. The two small packets are destined to the same receiver and the larger one to another receiver. The measurements are made at the receiver of the two small packets and consists of the difference between the arrival time of the first and the second small packet. As the second small packet queues behind the large one, the time that separates the first and the second small packet increases in the links shared with the larger packet.

Example 3.6. Figure 3.6 displays an example of a sandwich packet transmission. Terminal node T1 sends a sandwich packet, A,B and C. The packet B has destination node T5 and

packets A and C have destination node T6. The packets share the same path until they arrive at router R1.

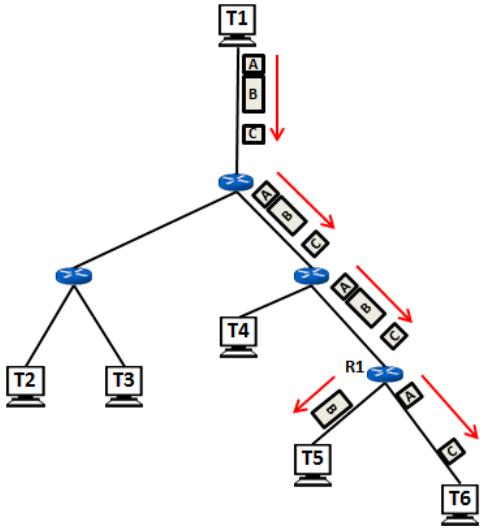


Figure 3.6: Transmission of a sandwich probe.

The inherent randomness of packet delays and packet losses leads to the fact that tomographic methods use statistical methodologies to infer a network topology. The first to use end-to-end packet loss measurements to infer the logical topology of a network were Ratnasamy and McCanne [113]. They observed that when packets are sent using multicast from a sender to many receivers, the receivers sharing a longer path in the routing tree have higher shared loss rates. To reconstruct a logical network topology they first define a selection criteria which indicates how closely terminal nodes are in the routing tree. They then recursively, until only one node is left, aggregate nodes with the highest similarity together and represent them as a single node. The shared losses of the receivers can have two origins. They may be due to losses along the shared path in the routing tree which are called the true shared losses. However, two copies of the same packet can be lost independently along the path that is not shared, these losses are random and are called the false shared losses. Ratnasamy and McCanne used the probability of seeing true shared losses between a pair of receivers as the selection criteria. They estimate the topology using the loss rate of the receivers.

Duffield et al. [51] established the correctness of the method presented by Ratnasamy and McCanne and Duffield et al. [52] presented more general methods that use packet loss measurements or/and delay measurements.

All the tomographic methods start by estimating end-to-end metrics based on the end-to-end measurements and using statistical models. After that, an inference algorithm that uses the pairwise metrics is applied to obtain the topology.

Consider that a sender s sends packets to a set of receivers R and that γ_{ij} is the theoretical metric associate with each pair of receivers $i, j \in R$. The metric must verify the monotonicity property, that is, if the path $P_{s,i}$ shares more links with the path $P_{s,j}$ than with the path $P_{s,k}$ then $\gamma_{ij} > \gamma_{ik}$, where $P_{s,i}, P_{s,j}, P_{s,k}$ represent the path from the source node s to the receiver node i, j and k, respectively. The metrics, $\gamma_{ij},\ \forall i, j \in R$, are estimated from the end-to-end-measurements, $x_{ij}, \forall i, j \in R$. These estimated metrics are statistics obtained from repeated measurements [23].

In what concerns the inference algorithms the two main algorithms used are the grouping procedure [51, 52, 53, 113, 122] and the maximum likelihood procedure [23, 34, 37, 51]. There are others inference algorithms like the Bayesian procedure [51] but these algorithms are not as efficient as the other two and therefore not often used.

The grouping procedure obtains the logical topology tree applying the following four steps [22]:

1. choose a pair of nodes with the highest similarity, using the estimated metrics;
2. group the pair of nodes to form a new node;
3. determine the similarities between the new node and the former existing nodes;
4. repeat the procedure until only one node is left.

The basic idea of the maximum likelihood procedure is to evaluate the statistical likelihood of every possible topology given the measurements, $x_{ij}, \forall i, j \in R$, and select the one which maximizes the likelihood. Let T be the underlying topology, $x = \{x_{ij} : i, j \in R, i \neq j\}$

and $\gamma = \{\gamma_{ij} : i, j \in R, i \neq j\}$. The likelihood is defined to be a quantity proportional to the probability of making the observations (x) given the model, this probability is given by $p(x|\gamma, T)$. The maximum likelihood tree is given by

$$T^*(x) = \arg\max_{T \in F} \quad \sup_{\gamma \in G} p(x|\gamma, T)$$

where F is the set of all possible trees connecting the sender to the receivers and G is the set of all metrics satisfying the monotonicity property [22, 23].

More recently, Ni and Tatikonda [103] used ideas and tools from phylogenetic inference to infer routing topologies. They adapted the neighbor-joining heuristic, a widely used algorithm to reconstruct phylogenetic trees from distance matrices. This algorithm was proposed by Saitou and Nei [116] and posteriorly modified by Studier and Kepler [134]. This method builds an unrooted tree in which the tree length is minimal, using as input a distance matrix with the distances between n terminal nodes. It starts with a tree topology in form of a star, with an internal node and all the n terminal nodes connected to it. Then, iteratively, it searches the nearest neighbor, that is, the pair of nodes that induces a tree with the smaller tree length. This pair is then clustered into a new internal node and the distance between the new node and the other nodes are calculated, obtaining a new distance matrix to be used in the next iteration. The iteration process stops when the number of internal nodes of the tree is $(n - 2)$ [110].

Ni and Tatikonda [103] used multicast end-to-end loss measurements and defined additive metrics as follows.

Definition 3.1. Let $T = (V, E)$ be a tree, where V is the set of nodes, E is the set of edges and let P_{ij} represent the path from node i to node j, $i, j \in V$.

$d : V \times V \to \mathbb{R}^+$ is an *additive metric* if

- $0 < d(e) < \infty, \quad \forall e \in E$;
- $d(i, j) = d(j, i)$;

- $d(i,j) = \begin{cases} \sum_{e \in P_{ij}} d(e) & i \neq j \\ 0 & i = j \end{cases} \quad \forall i, j \in V.$

To obtain the distances, they sent n probes from a source node to the other terminal nodes and for each terminal node i they consider the loss outcome, $X_i^{(t)}$, of the t-th probe. If node i received the t-th probe then $X_i^{(t)} = 1$, otherwise $X_i^{(t)} = 0$.

The estimated distances between terminal nodes were obtained as follows.

$$d(i,j) = \log \frac{\overline{X}_i \overline{X}_j}{\overline{X_i X_j}^2}, \quad \forall i, j \in V_t$$

where $V_t \subset V$ is the set of terminal nodes,

$$\overline{X}_i = \frac{1}{n} \sum_{t=1}^{n} X_i^{(t)},$$

$$\overline{X}_j = \frac{1}{n} \sum_{t=1}^{n} X_j^{(t)},$$

$$\overline{X_i X_j} = \frac{1}{n} \sum_{t=1}^{n} X_i^{(t)} X_j^{(t)}.$$

As stated before the end-to-end traffic measurements used in network tomography are packet loss measurements and packet delay measurements. Considering that nowadays the packet loss is very low, currently most of the measurements made are packet delay measurements.

3.2 The instances

To obtain the instances to run our tests we generated three networks using the network-level simulator NS-3 [3]. We performed three simulations named Simulation7, Simulation15 and Simulation20 with, respectively, 7, 15 and 20 terminal nodes. Figures 3.9, 3.8 and 3.7 display the routing trees used to run the simulation.

For each routing tree we used multicast probing to obtain the delays between the terminal nodes and compiled these delays in a matrix. We named the matrices S7 (an

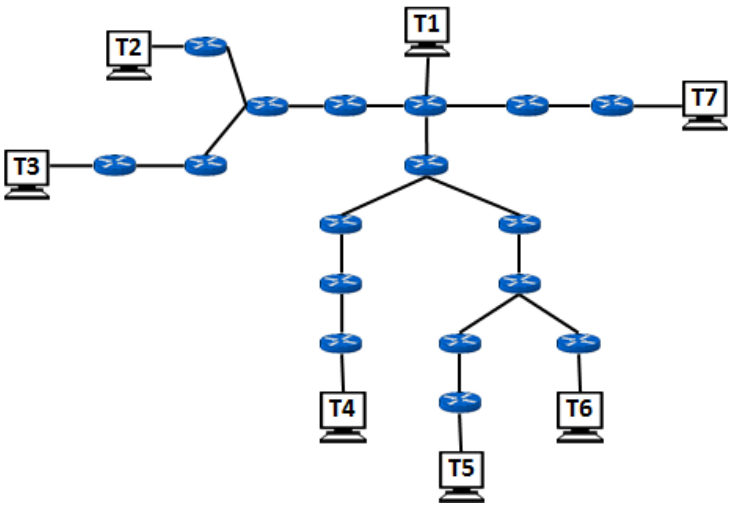


Figure 3.7: Routing tree used to run simulation Simulation7.

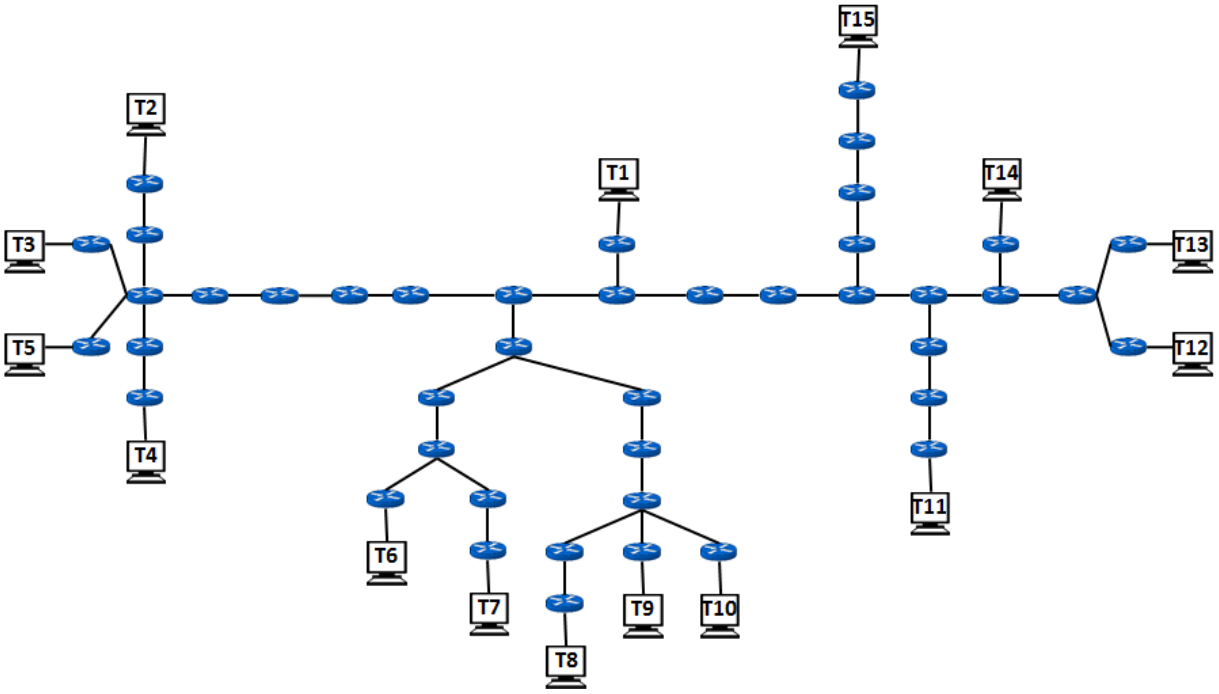


Figure 3.8: Routing tree used to run simulation Simulation15.

7×7 matrix), S15 (an 15×15 matrix) and S20 (an 20×20 matrix), which correspond to simulation Simulation7, Simulation15 and Simulation20, respectively.

From matrix S20 we extracted the principal submatrix that has the first 15 rows.

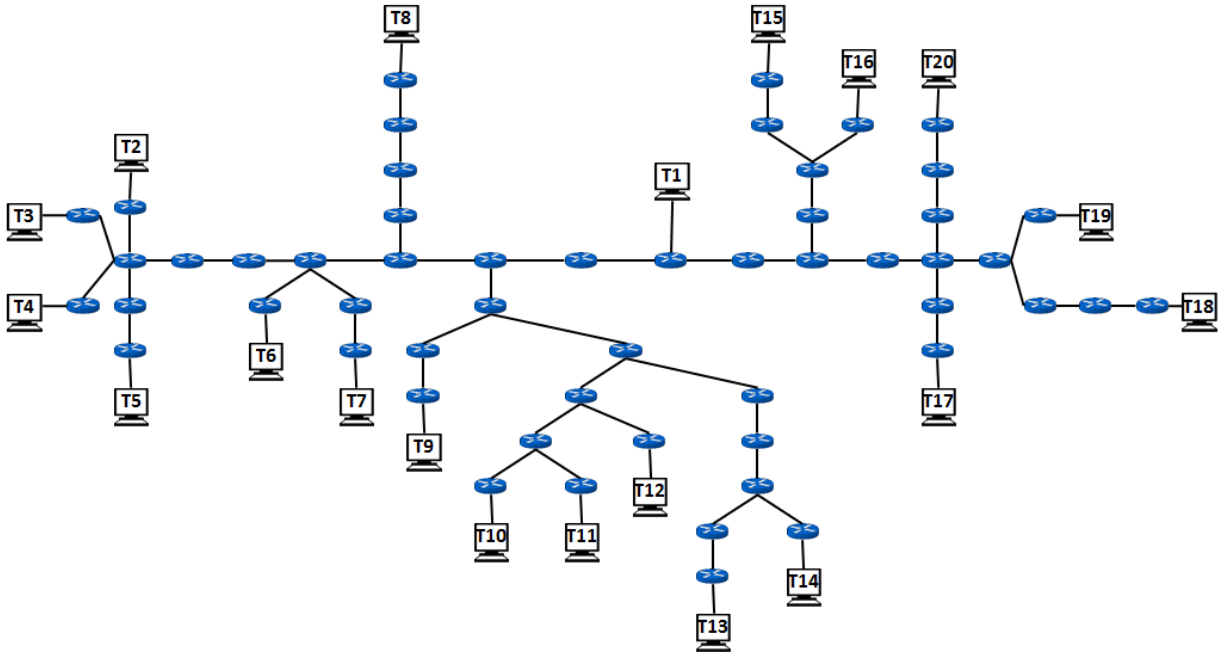


Figure 3.9: Routing tree used to run simulation Simulation20.

For each matrix we vary the number of terminal nodes between 5 and n, where $n = 15$ for matrices S20 and S15 and $n = 7$ for matrix S7. This way we obtain 25 principal submatrices.

Once we only use the distance matrix to infer the routing tree, we can only infer the logical topology. Therefore, we present in Figures 3.10, 3.11 and 3.12 the logical topologies corresponding to the routing topologies displayed in Figures 3.7, 3.8 and 3.9.

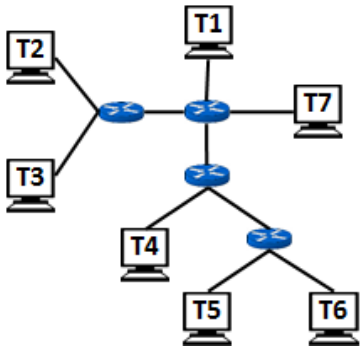


Figure 3.10: Logical topology obtained from the routing tree display in Figure 3.7.

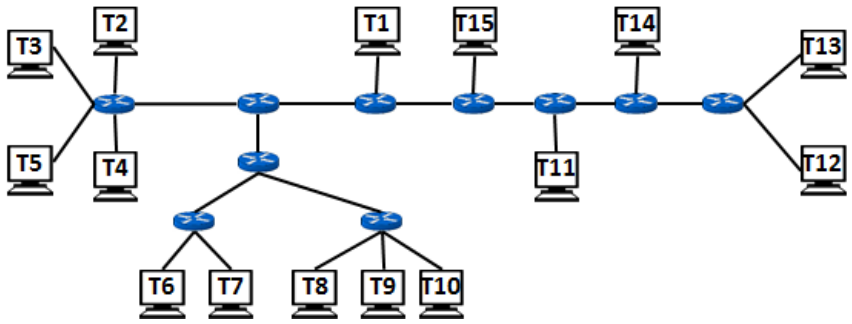


Figure 3.11: Logical topology obtained from the routing tree display in Figure 3.8.

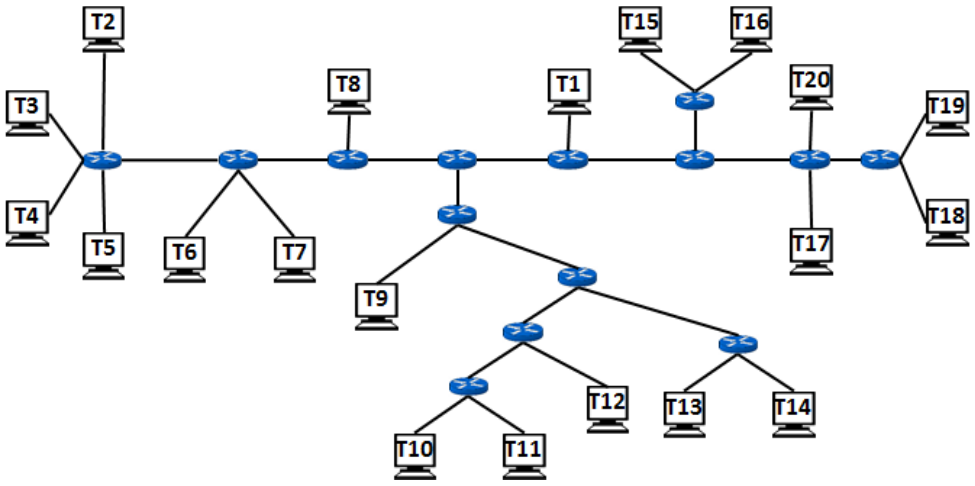


Figure 3.12: Logical topology obtained from the routing tree display in Figure 3.9.

We also generate matrices with random numbers. For each of the 25 matrices, $D = (d_{ij})$, we generate ten random values belonging to $[d_{ij}, d_{ij} + a \times d_{ij}]$, where $a \in \{0.1; 0.15; 0.2; 1\}$. Then we used the mean of the ten numbers to construct a new matrix. For $a = 0.1$ we obtain the matrices A10S7, A10S15 and A10S20, for $a = 0.15$ we obtain the matrices A15S7, A15S15 and A15S20, for $a = 0.2$ we obtain the matrices A20S7, A20S15 and A20S20 and for $a = 1$ we obtain the matrices A100S7, A100S15 and A100S20. Thus, we obtain more 100 matrices.

We also run other simulations where we defined the delays on the intermediate routers to be random. For these simulation we used the routing tree displayed in Figure 3.7 and the routing trees obtain from the trees displayed in Figures 3.8 and 3.9 by considering only

twelve of the terminal nodes. Figures 3.13 and 3.14 display the obtained routing trees and Figures 3.15 and 3.16 display the associate logical topology.

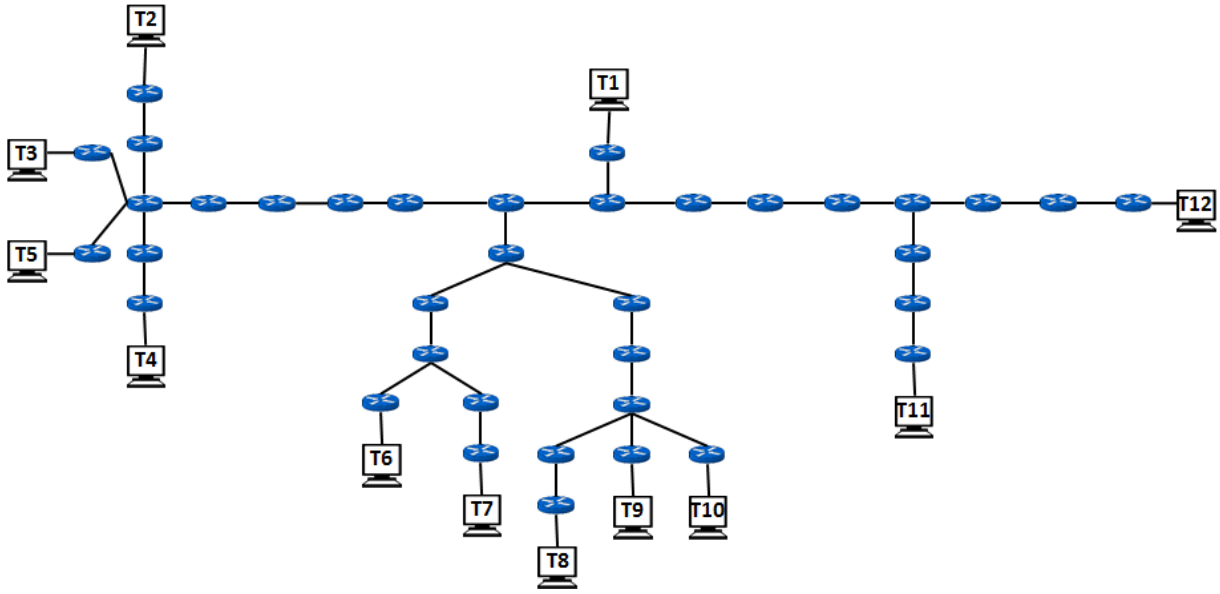


Figure 3.13: Routing tree obtain from the one used in simulation Simulation15 by considering only twelve terminal nodes.

For each routing tree we run ten simulations and used the mean of the delays obtained to construct the matrices. We named these matrices SS7 (an 7×7 matrix), SS15 (an 12×12 matrix) and SS20 (an 12×12 matrix) corresponding to the simulations ran using the routing tree displayed in Figures 3.7, 3.13 and 3.14, respectively. For each matrix we vary the number of terminal nodes between 5 and n, where $n = 12$ for matrices SS20 and SS15 and $n = 7$ for matrix SS7. This way we obtain 19 principal submatrices. So we have a total of 144 instances.

Once the distances used in the several matrices were obtained using random delays on the intermediate routers, these distances are subject to errors. Consequently, not all the matrices verify the triangle inequality and/or the four point condition.

Table 3.1 shows whether or not the matrices verify the triangle inequality. It also indicates how many times the triangle inequality is not verified in the matrix. Columns labeled **Matrix**, indicate the name of the matrix instance, columns labeled n indicate the

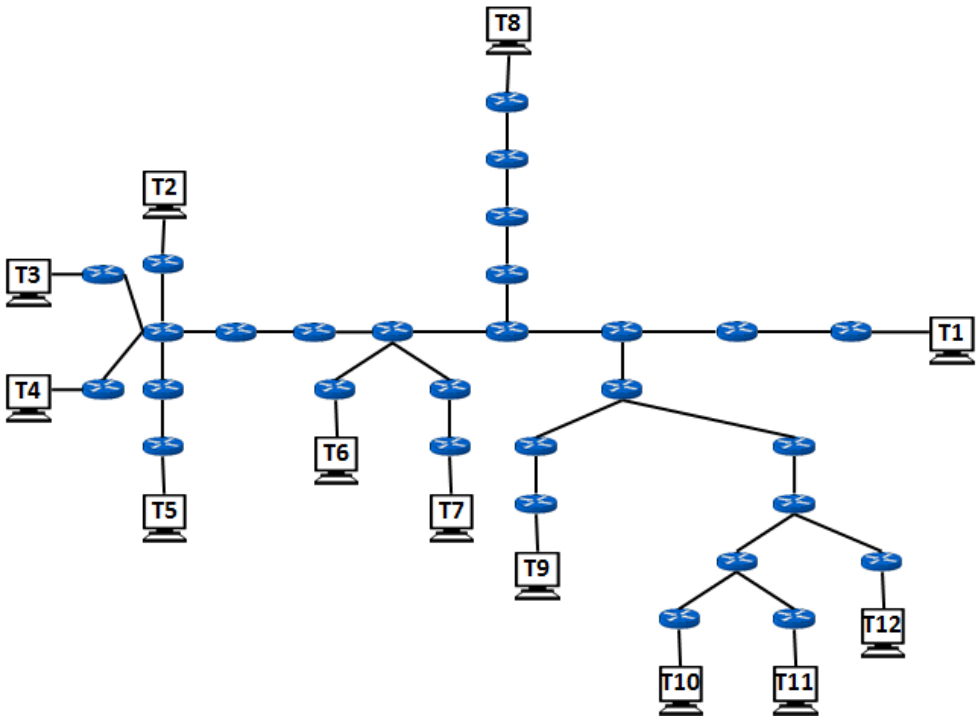


Figure 3.14: Routing tree obtain from the one used in simulation Simulation20 by considering only twelve terminal nodes.

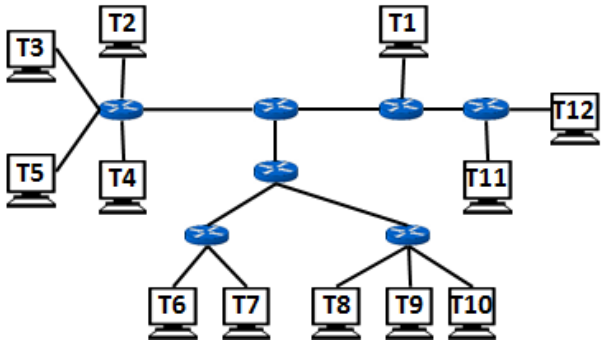


Figure 3.15: Logical topology obtained from the routing tree display in Figure 3.13.

order of the matrix, columns labeled **Verifies** indicate whether or not the matrix verifies the triangle inequality and columns labeled m indicate for how many distances the triangle inequality is not verified.

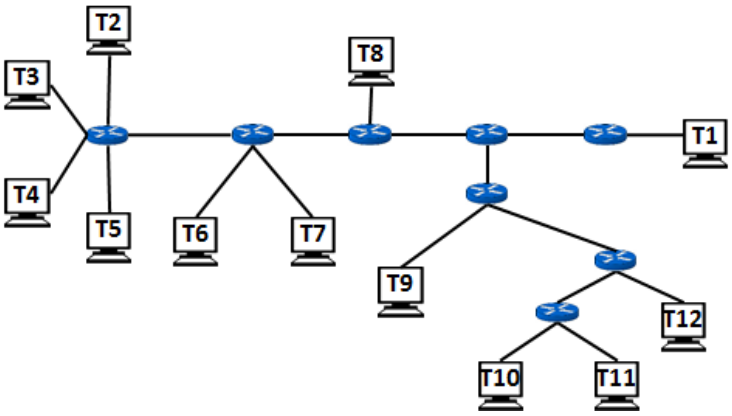


Figure 3.16: Logical topology obtained from the routing tree display in Figure 3.14.

Table 3.1: Verification of the triangle inequality for instances of the networking application.

Matrix	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m
	5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	no	1		5	no	3
S7	6	yes	–	A10S7	6	yes	–	A15S7	6	yes	–	A20S7	6	yes	–	A100S7	6	yes	–	SS7	6	no	3
	7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	no	3
	5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	yes	–
	6	yes	–		6	yes	–		6	yes	–		6	yes	–		6	yes	–		6	yes	–
	7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–
	8	yes	–		8	yes	–		8	yes	–		8	yes	–		8	no	1		8	yes	–
	9	yes	–		9	yes	–		9	yes	–		9	yes	–		9	yes	–		9	yes	–
S15	10	yes	–	A10S15	10	yes	–	A15S15	10	yes	–	A20S15	10	yes	–	A100S15	10	yes	–	SS15	10	yes	–
	11	yes	–		11	yes	–		11	yes	–		11	yes	–		11	yes	–		11	yes	–
	12	yes	–		12	yes	–		12	yes	–		12	yes	–		12	no	1		12	yes	–
	13	yes	–		13	yes	–		13	yes	–		13	yes	–		13	no	2		–	–	–
	14	yes	–		14	yes	–		14	yes	–		14	yes	–		14	no	3		–	–	–
	15	yes	–		15	yes	–		15	yes	–		15	yes	–		15	no	4		–	–	–
	5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	yes	–
	6	yes	–		6	yes	–		6	yes	–		6	yes	–		6	yes	–		6	yes	–
	7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–
	8	yes	–		8	yes	–		8	yes	–		8	yes	–		8	yes	–		8	yes	–
	9	yes	–		9	yes	–		9	yes	–		9	yes	–		9	yes	–		9	yes	–
S20	10	yes	–	A10S20	10	yes	–	A15S20	10	yes	–	A20S20	10	yes	–	A100S20	10	yes	–	SS20	10	yes	–
	11	yes	–		11	yes	–		11	yes	–		11	yes	–		11	yes	–		11	yes	–
	12	yes	–		12	yes	–		12	yes	–		12	yes	–		12	no	2		12	yes	–
	13	yes	–		13	yes	–		13	yes	–		13	yes	–		13	no	2		–	–	–
	14	yes	–		14	yes	–		14	yes	–		14	yes	–		14	no	2		–	–	–
	15	yes	–		15	yes	–		15	no	1		15	no	1		15	no	5		–	–	–

As we can observe almost all the matrices verify the triangle inequality.

Regarding the four-point condition 2.1 only matrices S20 with $n = 5$ and $n = 6$, S15 with $n = 5$ and $n = 6$ and S7 with $n = 5$ are additive.

Chapter 4

Phylogenetic Tree

Another application of the MWTR problem is the reconstruction of phylogenetic trees and therefore ideas and tools from the phylogenetic inference can be used to solve the MWTR problem. In this chapter, we present some concepts of the phylogenetics and some methods on how to obtain the distance matrix. We also present the instances of the phylogenetic area we used to run ours tests.

Phylogenetics is the branch of the biology that studies the evolution history of a species or any hierarchically recognized group [1]. Phylogenetic analysis can be done by comparing palaeontological characteristics (through the study of fossils), morphological characteristics (by studying the outer shape that living beings may take) and/or physiological characteristics (by studying the functions of various organs of living beings) of several species. Nowadays, with advances in molecular biology, phylogenetic trees are constructed by DNA analysis of several organisms, comparing families of nucleic acids or protein sequences [102].

A phylogenetic tree represents the evolutionary relationships of a set of species (also called taxon), where terminal nodes represent the observed species, additional nodes represent common ancestors, edges represent the evolutionary relationships between pairs of nodes and edge weights represent the quantification of this evolutionary relationship [60, 101, 102]. It is worth noting that phylogenetic trees allow understanding of

[1] in Oxford English Dictionary

the evolutionary history of species and can assist in the development of vaccines [68] and the study of biodiversity [89].

Example 4.1. In Figure 4.1 we present an example of a phylogenetic tree.

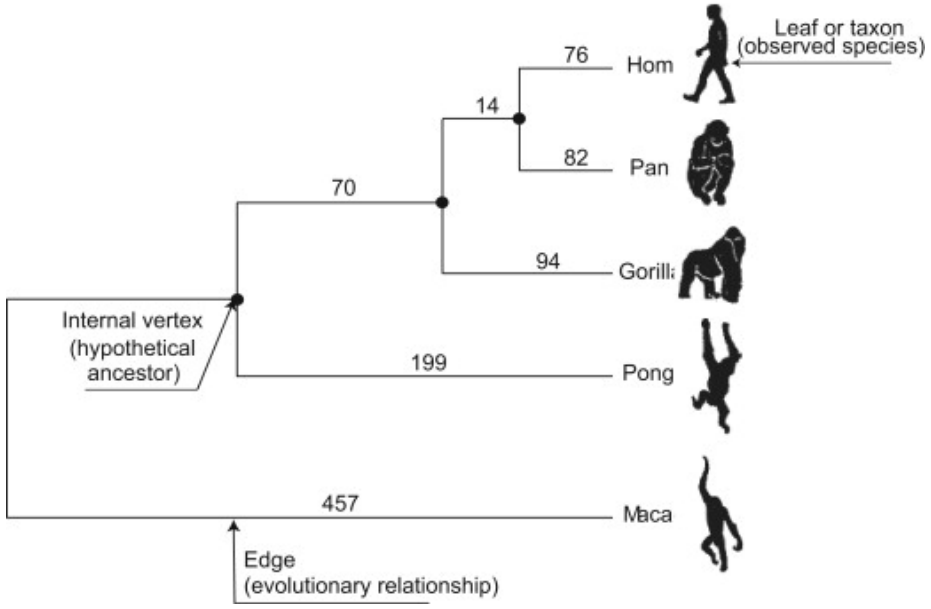


Figure 4.1: A phylogenetic tree [24].

In computational biology, inferring a phylogenetic tree is one of the steps of the phylogenetic reconstruction. As part of such inference is the determination of the tree topology and the determination of the branch lengths, which requires further analysis methods.

According to Prado et. al. [112] the methods to infer phylogenetic trees can be divided in two groups:

- Model-based methods. These methods select, using some principle, the best or a good phylogenetic tree from a set of candidate trees and generally this is done using probabilistic models arising from the maximum likelihood approach.

- Non-model-based methods. These methods use an algorithm to obtain the best or a good phylogenetic tree.

All these methods make specific assumptions about evolution and are based on a set of criteria, that can be express through an objective function.

Non-model-based methods includes parsimony methods and distance methods. Distance methods exploit the existence of a measure of dissimilarity (also called distance) among pairs of species, represented using a distance matrix, and aim at determining the tree topology together with branch lengths.

One of the most used criteria to infer the phylogenetic tree is the minimum evolution principle, which was introduced for the first time by Kidd and Sgaramella-Zonta [90]. In the minimum evolution principle, the best tree is the one that minimizes the sum of all branch lengths [102].

The problem of finding the phylogenetic tree that satisfies the minimum evolution principle is called the Minimum Evolution Problem (MEP) and is defined as follows [26]:

Definition 4.1. Given a set Γ of n taxa and pairwise distances d_{ij}, $\forall i, j \in \Gamma$ $(i \neq j)$ find a phylogenetic tree $T = (V, E, w) \in \mathcal{T}$ that solves the problem:

$$\begin{aligned} \min_{T \in \mathcal{T}} \mathcal{L}(T) &= \sum_{e \in E} w_e \\ s.t. \sum_{e \in P_{ij}} w_e &\geq d_{ij} \qquad \forall i, j \in \Gamma : i \neq j \\ w_e &\geq 0 \qquad \forall e \in E \end{aligned}$$

where $\mathcal{T}$ is the set of all possible phylogenetic trees and P_{ij} is the unique path in T connecting the taxa $i, j \in \Gamma$ $(i \neq j)$.

A variant of the minimum evolution principle, first introduced by Pauplin [107], is the balanced minimum evolution principle. Within this principle, sibling subtrees have equal weight, as opposed to the standard version where all terminal nodes have the same weight and thus the weight of a subtree is equal to the number of its terminal nodes [44]. The problem which uses this principle is the Balanced Minimum Evolution Problem (BMEP) and can be defined as followed [28].

Definition 4.2. Given a set Γ of n taxa and pairwise distances d_{ij}, $\forall i, j \in \Gamma$ $(i \neq j)$ find a phylogenetic tree $T = (V, E, w) \in \mathcal{T}$ that minimizes the length function:

$$\sum_{i,j \in \Gamma, i \neq j} d_{ij} \; 2^{-z_{ij}}$$

where $\mathcal{T}$ is the set of all possible phylogenetic trees and z_{ij} indicates the number of edges in the unique path P_{ij} in T connecting the taxa $i, j \in \Gamma$ $(i \neq j)$.

4.1 Methods to obtain the distances

The quality of the phylogenetic tree obtained by solving the MEP or the BMEP depends on the quality of the evolutionary distances used to infer the tree and therefore also on the way these distances were obtained. In this section, we present some of the methods used to obtain the evolutionary distances.

A DNA structure is composed of four nucleotides, adenine (A), thymine (T), cytosine (C), and guanine (G) [102]. A DNA sequence is a sequence of the letters A, T, C and G.

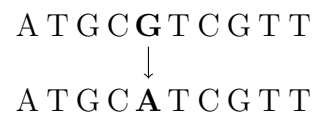
Example 4.2. An example of a DNA sequence is the following sequence:

ATGCGTCGTT

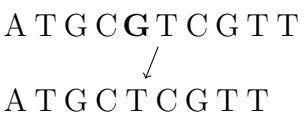
During evolution the genes mutate and therefore the DNA sequence changes. There are four basic types of change: substitution of a nucleotide for another nucleotide, deletion of nucleotides, insertion of nucleotides and inversion of nucleotides [102].

Example 4.3. Figure 4.2 displays the four basic types of changes that can occur in a DNA sequence.

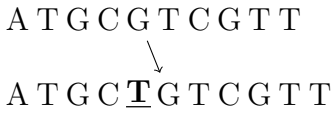
There are two classes of substitutions, the transitions and the transversions. There is a transition when a purine nucleotide (adenine or guanine) is substituted by another purine nucleotide or when a pyrimidine nucleotide (cytosine or thymine) is substituted by another



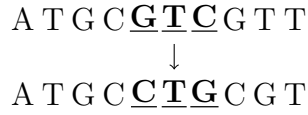
(a) Substitution of a nucleotide.



(b) Deletion of a nucleotide.



(c) Insertion of a nucleotide.



(d) Inversion of nucleotides.

Figure 4.2: The four basic types of changes occurring in a DNA sequence.

pyrimidine nucleotide. We have a transversion when a purine nucleotide is substituted by a pyrimidine nucleotide or vice versa [25].

When two species descend from a common ancestor, the DNA sequence of the two species diverges by nucleotide substitution [102].

The distances between two species (sequences), should estimate the number of changes since the two species diverge from a common ancestor.

The simplest way to determine the distances is to use the Hamming distances, which count the number of sites where the nucleotides differ in two aligned sequences.

Definition 4.3. [70] The Hamming distance, $H(x, y)$, between two sequences $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ is equal to $|\{i : x_i \neq y_i\}|$.

Example 4.4. The Hamming distance of the two aligned sequences presented below is 5.

(X) ATGCGTCGTT
(Y) ATCCGCGATC

Another simple way to calculate the distance between two DNA sequences is through the *p distance* which measures the proportion of divergence [102]. This proportion (p) is obtained by dividing the number of sites where nucleotides differ in two aligned sequences

(n_d) by the number of nucleotides examined (n), that is,

$$p = \frac{n_d}{n}.$$

Example 4.5. The p distance of the two sequences presented in Example 4.4 is

$$p = \frac{5}{10} = 0.5.$$

Sometimes it is useful to determine a matrix of the relative frequencies of nucleotide pairs. Since there are four nucleotides the matrix of the frequencies of nucleotide pairs, when comparing a sequence X with a sequence Y, has sixteen entries and can be represented as:

$$F_{XY} = \begin{bmatrix} \frac{n_{AA}}{n} & \frac{n_{AC}}{n} & \frac{n_{AG}}{n} & \frac{n_{AT}}{n} \\ \frac{n_{CA}}{n} & \frac{n_{CC}}{n} & \frac{n_{CG}}{n} & \frac{n_{CT}}{n} \\ \frac{n_{GA}}{n} & \frac{n_{GC}}{n} & \frac{n_{GG}}{n} & \frac{n_{GT}}{n} \\ \frac{n_{TA}}{n} & \frac{n_{TC}}{n} & \frac{n_{TG}}{n} & \frac{n_{TT}}{n} \end{bmatrix}$$

where n_{ij} is the number of times the nucleotide i of sequence X is aligned with nucleotide j of sequence Y and n is the number of nucleotides examined [82].

Example 4.6. The matrix with the frequencies of nucleotides pair of Example 4.4 is given by:

$$F_{XY} = \begin{bmatrix} \frac{1}{10} & 0 & 0 & 0 \\ 0 & \frac{1}{10} & \frac{1}{10} & 0 \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{5} \end{bmatrix} \tag{4.1}$$

The Hamming distance and the p distance may underestimate the number of changes, once they do not take into account all the substitution, like backwards, parallel and multiple substitutions, that occurred since two DNA sequences diverged from a common ancestor [102]. Example 4.7 shows how the Hamming distance and the p distance only reflect some of the changes occurred since two DNA sequences diverged.

Example 4.7. Figure 4.3 presents two DNA sequences fragments, sequence X and sequence Y, that diverged from a common ancestor, with DNA sequence fragment ACGAACGTAAGC. Figure 4.3 also presents the changes occurred during the evolution. The Hamming distance between these two sequences is 3 and the p distance is 0.25 and we can see that these distances do not reflect all the changes occurred since they diverged.

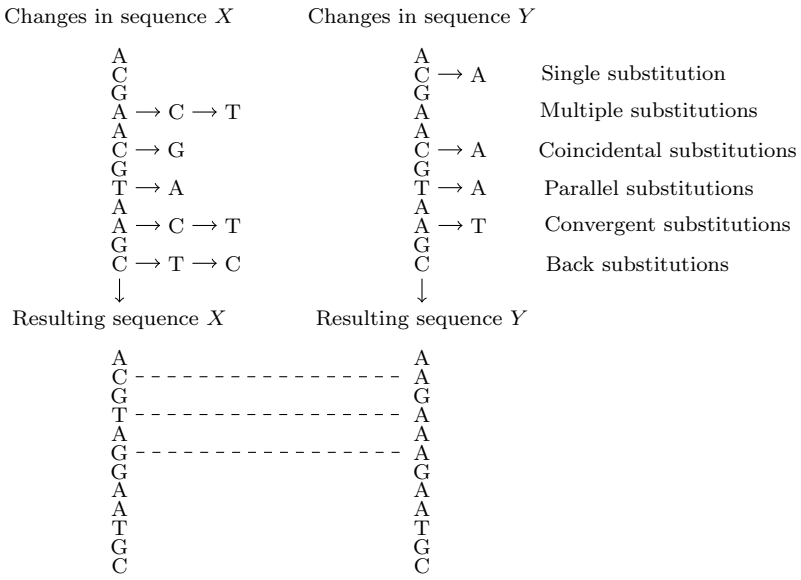


Figure 4.3: Some types of changes that can occur in two DNA sequences fragments that diverged from a common ancestor [101].

To improve the quality of the distances it is necessary to model the nucleotide substitution process and this can be done by time-homogeneous stationary Markov model.

This model makes the following assumptions [93]:

- The substitution of a nucleotide by another is random and independent (Markov property);
- The substitution rates do not change over time (homogeneity);
- The relative frequencies of the nucleotides, $\pi_i, i \in \{A, C, G, T\}$, do not change over time and from one sequence to another (stationarity).

In this model, each site of the DNA sequence is a random variable that can have four possible states: A, C, G or T. Let $P_{ij}(t), \forall i, j \in N = \{A, C, G, T\}$, denote the probability of substituting nucleotide i by nucleotide j, after a certain period of time t. All the probabilities can be compiled in a transition probability matrix, $P(t)$:

$$P(t) = \begin{bmatrix} P_{AA}(t) & P_{AC}(t) & P_{AG}(t) & P_{AT}(t) \\ P_{CA}(t) & P_{CC}(t) & P_{CG}(t) & P_{CT}(t) \\ P_{GA}(t) & P_{GC}(t) & P_{GG}(t) & P_{GT}(t) \\ P_{TA}(t) & P_{TC}(t) & P_{TG}(t) & P_{TT}(t) \end{bmatrix}$$

The probabilities satisfy the following conditions:

- $P_{ij}(t) > 0 \qquad \forall i, j \in N, \forall t > 0$
- $\sum_{j \in N} P_{ij}(t) = 1 \qquad \forall i \in N, \forall t > 0$

It is also assumed that the substitution problem is reversible, that is

$$\pi_i P_{ij}(t) = \pi_j P_{ji}(t), \quad \forall i, j \in N, \forall t > 0.$$

This method uses a rate matrix, Q, which gives the relative rate of change of each nucleotide along the sequence and such that $Q_{ii} = -\sum_{j \in N, j \neq i} Q_{ij}, \ \forall i \in N$. The rows and columns of Q follow the order A, C, G, T. So, the element Q_{12}, for example, indicate the instantaneous rate of change from nucleotide A to nucleotide C [93].

Using this rate matrix, the number of substitutions during time t is given by $d = \mu t$, where $\mu = -\sum_{i \in N} \pi_i Q_{ii}$ is the mean instantaneous substitution rate that indicate the total number of substitution per unite of time.

We also have that the probability that a substitution occur after time t is given by:

$$p = 1 - \sum_{i \in N} \pi_i P_{ii}(t)$$

where π_i is the relative frequency of nucleotide i and $P_{ii}(t)$ is the probability that nucleotide i is not substituted by another nucleotide after the period of time t.

Once $t = \frac{d}{\mu}$, we obtain

$$p = 1 - \sum_{i \in N} \pi_i P_{ii} \left(-\frac{d}{\sum_{i \in N} \pi_i Q_{ii}} \right) \tag{4.2}$$

By estimating p using the p distance of the two observed sequences and solving this equation in order of d, we can construct a *method of moments estimator*[2] of the evolutionary distance between two sequences.

Depending on the definition of the relative frequencies of the nucleotides $\pi_i, i \in N$ and the rate matrix Q, we obtain different ways to obtain the distance, d_{XY}, between a sequence X and a sequence Y.

The simplest model is the Jukes and Cantor's model [88]. In this model it is assumed that $\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$ and that the nucleotide substitutions occur with the same frequency. The rate matrix of this model is given by:

$$Q = \begin{bmatrix} -\frac{3\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & -\frac{3\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & -\frac{3\mu}{4} & \frac{\mu}{4} \\ \frac{\mu}{4} & \frac{\mu}{4} & \frac{\mu}{4} & -\frac{3\mu}{4} \end{bmatrix}$$

The distance, d_{XY}, between sequence X and sequence Y is given by

$$d_{XY} = -\frac{3}{4} \ln \left(1 - \frac{4}{3} p \right)$$

where p is the p distance between sequence X and sequence Y.

This model can only be used if the p distance between two sequences is lower then 0.75.

To overcome the limitation of Jukes and Cantor's model, Felsenstein [59] and Tajima and Nei [135] propose a model where the relative frequencies of the nucleotides may not be equal and are defined according to the group of species studied. However, these frequencies must verify $\pi_A + \pi_C + \pi_G + \pi_T = 1$. The nucleotide substitutions occur with the same

[2] In statistics the method of moments estimator is a method of estimating the population parameters.

frequency. The rate matrix of this model is given by [102]:

$$Q = \begin{bmatrix} 1-\alpha(\pi_C+\pi_G+\pi_T) & \alpha\pi_C & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & 1-\alpha(\pi_A+\pi_G+\pi_T) & \alpha\pi_G & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & 1-\alpha(\pi_A+\pi_C+\pi_T) & \alpha\pi_T \\ \alpha\pi_A & \alpha\pi_C & \alpha\pi_G & 1-\alpha(\pi_A+\pi_C+\pi_G) \end{bmatrix}$$

where α is the rate of the substitutions.

The distance, d_{XY}, between sequence X and sequence Y is given by

$$d_{XY} = -b\ln\left(1-\frac{p}{b}\right)$$

where p is the p distance between sequence X and sequence Y and $b = 1-(\pi_A^2+\pi_C^2+\pi_G^2+\pi_T^2)$ [101].

The two models presented so far assume that all substitutions occur with the same frequency. In practice, however, transitions (substitutions $A \leftrightarrow G$ and $C \leftrightarrow T$) occur more often than transversions (the remaining substitutions). The model presented by Kimura [91] assume that transitions and transversions happen with different frequencies. This model also assume that the relative frequencies of the nucleotides are equal ($\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$).

The rate matrix for this model is given by:

$$Q = \begin{bmatrix} 1-(\alpha+2\beta) & \beta & \alpha & \beta \\ \beta & 1-(\alpha+2\beta) & \beta & \alpha \\ \alpha & \beta & 1-(\alpha+2\beta) & \beta \\ \beta & \alpha & \beta & 1-(\alpha+2\beta) \end{bmatrix}$$

where α represents the rate of the transitions and β represents the rate of the transversions.

The distance, d_{XY}, between sequence X and sequence Y is given by

$$d_{XY} = -\frac{1}{2}\ln(1-2a-b) - \frac{1}{4}\ln(1-2b)$$

where a is the proportion of transitions, $a = \frac{n_AG+n_GA+n_CT+n_TC}{n}$ and b is the proportion of transversions, $b = \frac{n_AC+n_CA+n_AT+n_TA+n_CG+n_GC+n_GT+n_TG}{n}$ [101, 102].

Tavaré [136] presented a more general model, the generalised time reversible (GTR) model. In this model the relative frequencies of the nucleotides may not be equal and the nucleotide substitutions may occur with different frequencies.

The rate matrix of this model is given by:

$$Q = \begin{bmatrix} -\mu(a\pi_C + b\pi_G + c\pi_T) & a\mu\pi_C & b\mu\pi_G & c\mu\pi_T \\ a\mu\pi_A & -\mu(a\pi_A + e\pi_G + f\pi_T) & e\mu\pi_G & f\mu\pi_T \\ b\mu\pi_A & e\mu\pi_C & -\mu(b\pi_A + e\pi_C + g\pi_T) & g\mu\pi_T \\ c\mu\pi_A & f\mu\pi_C & g\mu\pi_G & -\mu(c\pi_A + f\pi_C + g\pi_G) \end{bmatrix}$$

where a, b, c, e, f and g are relative rate parameters that describe the relative rate of each nucleotide substitution. For example, the parameter a indicates the relative rate of the nucleotide A to be substituted by the nucleotide C.

The rate matrix of the GTR model has more free parameters than the other models, therefore solving equation 4.2 is more complex and consequently the formula to obtain the distance d_{XY} is quite complicated. Catanzaro et. al. [29] presented a procedure to estimate the unknown parameters of the rate matrix and determine the distances between two sequences. The instances from the phylogenetic area we used to run our tests were obtained using this procedure.

4.2 The instances

The data instances coming from the phylogenetics application, we used to run our tests, are available from `http://pubsonline.informs.org/doi/\-suppl\-/10.1287/ijoc.1110.0455` [28]. From this set we use three phylogenetic distance matrices, matrix Primate, an 12×12 matrix, matrix M391, an 17×17 matrix and matrix M887, an 18×18 matrix. From matrices M391 and M887 we extracted the principal submatrix that has the first 15 rows. For each matrix we vary the number of terminal nodes (taxa) between 5 and n, where $n = 15$ for matrices M391 and M887 and $n = 12$ for matrix Primate. This way we obtain 30 principal submatrices.

We also generate matrices with random numbers. For each of the 30 matrices, $D = (d_{ij})$,

we generate ten random values belonging to $[d_{ij}, d_{ij} + a \times d_{ij}]$, where $a \in \{0.1; 0.15; 0.2; 1\}$. Then we used the mean of the ten numbers to construct a new matrix. For $a = 0.1$ we obtain the matrices A10M391, A10M887 and A10Pri, for $a = 0.15$ we obtain the matrices A15M391, A15M887 and A15Pri, for $a = 0.2$ we obtain the matrices A20M391, A20M887 and A20Pri and for $a = 1$ we obtain the matrices A100M391, A100M887 and A100Pri. Thus, we obtain more 120 matrices beside the other 30, making a total of 150 instances.

The distances used in matrices M391, M887 and Primate are derived from experimental procedures and therefore these distances are subject to measurement errors. Consequently, not all the matrices verify the triangle inequality and/or the four point condition.

Table 4.1 shows whether or not the matrices verify the triangle inequality. It also indicates how many times the triangle inequality is not verified in the matrix. Columns labeled **Matrix**, indicate the name of the matrix instance, columns labeled n indicate the order of the matrix, columns labeled **Verifies** indicate whether or not the matrix verifies the triangle inequality and columns labeled m indicate for how many distances the triangle inequality is not verified.

Table 4.1: Verification of the triangle inequality for instances of the phylogenetic application.

Matrix	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m	**Matrix**	n	**Verifies**	m
	5	yes	–		5	yes	–		5	yes	–		5	yes	–		5	yes	–
	6	yes	–		6	yes	–		6	yes	–		6	yes	–		6	yes	–
	7	yes	–		7	yes	–		7	yes	–		7	yes	–		7	yes	–
	8	yes	–		8	yes	–		8	yes	–		8	yes	–		8	yes	–
	9	yes	–		9	yes	–		9	yes	–		9	yes	–		9	yes	–
M391	10	yes	–	A10M391	10	yes	–	A15M391	10	yes	–	A20M391	10	yes	–	A100M391	10	yes	–
	11	yes	–		11	yes	–		11	yes	–		11	yes	–		11	yes	–
	12	yes	–		12	yes	–		12	no	1		12	no	1		12	no	1
	13	yes	–		13	no	1		13	no	1		13	no	1		13	no	1
	14	yes	–		14	no	1		14	no	1		14	no	1		14	no	1
	15	yes	–		15	yes	–		15	yes	–		15	yes	–		15	yes	–
	5	no	2		5	no	2		5	no	2		5	no	2		5	no	2
	6	no	2		6	no	2		6	no	2		6	no	2		6	no	3
	7	no	2		7	no	2		7	no	2		7	no	2		7	no	1
	8	no	5		8	no	6		8	no	7		8	no	7		8	no	11
Primate	9	no	7	A10Pri	9	no	7	A15Pri	9	no	6	A20Pri	9	no	7	A100Pri	9	no	19
	10	no	17		10	no	20		10	no	22		10	no	23		10	no	35
	11	no	17		11	no	22		11	no	22		11	no	21		11	no	31
	12	no	17		12	no	23		12	no	25		12	no	25		12	no	44
	5	no	1		5	no	1		5	no	1		5	no	1		5	no	1
	6	no	2		6	no	2		6	no	2		6	no	2		6	no	3
	7	no	3		7	no	3		7	no	3		7	no	3		7	no	4
	8	no	5		8	no	5		8	no	5		8	no	5		8	no	6
	9	no	7		9	no	7		9	no	7		9	no	7		9	no	9
M887	10	no	7	A10M887	10	no	7	A15M887	10	no	7	A20M887	10	no	7	A100M887	10	no	6
	11	no	9		11	no	9		11	no	9		11	no	9		11	no	9
	12	no	15		12	no	15		12	no	17		12	no	17		12	no	17
	13	no	15		13	no	16		13	no	16		13	no	17		13	no	19
	14	no	18		14	no	18		14	no	19		14	no	18		14	no	22
	15	no	21		15	no	22		15	no	23		15	no	22		15	no	23

As we can observe none of the matrix obtain from matrix M887 and matrix Primate verify the triangle inequality and almost all the matrix obtain from matrix M391 verify the triangle inequality.

None of the 150 matrices of the phylogenetic area verify the four-point condition 2.1.

Chapter 5

Exact Formulation

In this chapter, we introduce two compact mixed integer linear programming (MILP) formulations of the MWTR problem, the Path-weight formulation and the Path-edges formulation. We also include valid equalities and inequalities which improve the performance of the second formulation and present the computational results obtained by running the two formulations when using data instances from networking application and phylogenetic application presented in 3.2 and 4.2, respectively.

Both formulations we present use flows to help with the definition of the tree topology, however each with a different underlying reconstructing idea. The Path-weight formulation, produces a tree as a solution and the Path-edges formulation, constructs a balanced tree.

The Path-weight formulation is an extended formulation that produces a tree as a solution to the MWTR problem, using additional flow variables to ensure connectivity. This formulation is inspired from Model 4 in Catanzaro et al. [27]. As reported in [27], the linear relaxation of this model has an objective value of zero. We strengthen the formulation by adding valid inequalities that cut off the zero-objective solutions.

The Path-edges formulation produces a balanced tree solution of the MWTR problem and flows are defined between every pair of terminal nodes. In Catanzaro et al. [28] a balanced tree is also considered. However, we use multicommodity flows to deal with connectivity and path lengths. Instead of minimizing the total sum of all edge weights,

the objective function deals with the minimization of the total tree length.

As stated before in chapter 2, we consider a tree $T = (V, E)$ spanning the set of nodes $V = V_a \cup V_t$. V_a is the set of additional or internal nodes and V_t is the set of terminal or external nodes. The pairwise distances between terminal nodes in V_t are given in distance matrix D. The tree topologies we consider are such that the additional nodes of the tree all have degree three.

Both formulations use the following variables. Binary variables x_{ij}, $i \in V_a$, $j \in V$, $i < j$ indicate whether edge $\{i, j\}$ belongs to the tree solution, while continuous variables $w_{ij} \geq 0$ represent the weight associated to edge $\{i, j\}$.

5.1 Path-weight formulation

We want to obtain a tree T and associated weights w_e, $e \in T$, that provide a weak realization of distance matrix D. Consider the path P_{ij} that connects terminal nodes i and j, $i, j \in V_t$. By definition of a weak realization,

$$\sum_{e \in P_{ij}} w_e \geq d_{ij} \tag{5.1}$$

must hold. To ensure this, consider additional continuous variables u_{ij}, for all $i, j \in V$, $i \neq j$, which indicate the length of the path between nodes i and j, that is $u_{ij} = \sum_{e \in P_{ij}} w_e$.

In order to impose connectivity several approaches can be used. Usual approaches consists either in the inclusion of the subtour elimination inequalities

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, S \subset V, |S| > 1 \tag{5.2}$$

or in the inclusion of the cut-set inequalities

$$\sum_{i \in S, j \in S^c} x_{ij} \geq 1, S \subset V, 0 \in S. \tag{5.3}$$

The linear relaxation of both models provide the same bound, however the number of inequalities increases exponentially with the size of the model. It is well known that in order to ensure connectivity/prevent circuits, instead of using one of the families of inequalities (5.2) or (5.3) with an exponential number of inequalities, one can use compact extended formulations [96]. The most common are derived using either the well-known Miller-Tucker-Zemlin inequalities [73, 98] or using stronger multicommodity flow formulations [74, 96]. In the formulation presented below we use multicommodity flows. Fixing additional node 1 as the root of the flow, we introduce binary flow variables y_{ij}^k for all $k \in V_t$, $i \in V_a$, $j \in V$ with $i < j$, indicating whether edge $\{i, j\}$ is used from i to j in the path from flow root node 1 to terminal node k.

Let $d_{max} := \max\{d_{ij} : i, j \in V_t\}$. The formulation that minimizes the total edges weights and reconstructs an unrooted tree for the MWTR problem is as follows.

Path-weight formulation

$$min \quad \sum_{i \in V_a} \sum_{\substack{j \in V \\ j > i}} w_{ij}$$

subject to

$$\sum_{i \in V_a} \sum_{\substack{j \in V \\ j > i}} x_{ij} = 2n - 3 \tag{5.4}$$

$$\sum_{\substack{j \in V \\ j > i}} x_{ij} + \sum_{\substack{j \in V_a \\ i > j}} x_{ji} = 3, \qquad \forall i \in V_a \tag{5.5}$$

$$\sum_{\substack{i \in V_a \\ j > i}} x_{ij} = 1, \qquad \forall j \in V_t \tag{5.6}$$

$$x_{i,i+1} = 1, \qquad \forall i \in \{1, ..., (\lceil n/2 \rceil - 1)\} \tag{5.7}$$

$$\sum_{\substack{j \in V_a \cup \{k\} \\ j \neq 1}} y_{1j}^k = 1, \qquad \forall k \in V_t \tag{5.8}$$

$$\sum_{i \in V_a} y_{ik}^k = 1, \qquad \forall k \in V_t \tag{5.9}$$

$$\sum_{\substack{i \in V_a \\ i<j}} y_{ij}^k = \sum_{\substack{i \in V_a \cup \{k\} \\ i>j}} y_{ji}^k, \qquad \forall k \in V_t, \forall j \in V_a \setminus \{1\} \tag{5.10}$$

$$y_{ij}^k \leq x_{ij}, \qquad \forall k \in V_t, \forall i \in V_a, \forall j \in V \setminus \{1\}, j > i \tag{5.11}$$

$$u_{ij} \geq d_{ij}, \qquad \forall i, j \in V_t, i < j \tag{5.12}$$

$$u_{ij} \geq w_{ij}, \qquad \forall i \in V_a, \forall j \in V_t, i < j \tag{5.13}$$

$$d_{max} x_{ij} \geq w_{ij}, \qquad \forall i \in V_a, \forall j \in V, i < j \tag{5.14}$$

$$w_{ij} \geq u_{ij} - d_{max}(1 - x_{ij}), \qquad \forall i \in V_a, \forall j \in V, i < j \tag{5.15}$$

$$w_{ij} \geq u_{ik} - u_{jk} - d_{max}(1 - x_{ij}), \qquad \forall i \in V_a, \forall j, k \in V, i < j, j < k \tag{5.16}$$

$$w_{ij} \geq u_{jk} - u_{ik} - d_{max}(1 - x_{ij}), \qquad \forall i \in V_a, \forall j, k \in V, i < j, j < k \tag{5.17}$$

$$w_{ij} \geq u_{ik} - u_{kj} - d_{max}(1 - x_{ij}), \qquad \forall i \in V_a, \forall j, k \in V, i < k, k < j \tag{5.18}$$

$$w_{ij} \geq u_{kj} - u_{ik} - d_{max}(1 - x_{ij}), \qquad \forall i \in V_a, \forall j, k \in V, i < k, k < j \tag{5.19}$$

$$w_{ij} \geq u_{ki} - u_{kj} - d_{max}(1 - x_{ij}), \qquad \forall i, k \in V_a, \forall j \in V, i < j, k < i \tag{5.20}$$

$$w_{ij} \geq u_{kj} - u_{ki} - d_{max}(1 - x_{ij}), \qquad \forall i, k \in V_a, \forall j \in V, i < j, k < i \tag{5.21}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in V_a, \forall j \in V, i < j \tag{5.22}$$

$$y_{ij}^k \in \{0, 1\}, \qquad \forall k \in V_t, \forall i \in V_a, \forall j \in V, i < j \tag{5.23}$$

$$w_{ij} \geq 0, \qquad \forall i \in V_a, \forall j \in V, i < j \tag{5.24}$$

$$u_{ij} \geq 0, \qquad \forall i, j \in V, i < j \tag{5.25}$$

Constraints (5.4)-(5.7) define a spanning tree with all additional nodes with degree three and all terminal nodes with degree one. The cardinality constraints (5.4) ensure that there are $2n - 3$ edges in the solution. Constraints (5.5) ensure that all additional nodes have degree three. Constraints (5.6) ensure that all terminal nodes have degree one. As already mentioned, to reduce the symmetry, we fix a path between the first additional nodes with constraints (5.7). Constraints (5.8)-(5.10) are flow conservation constraints and constraints (5.11) are linking constraints between flow variables y_{ij}^k and topology variables x_{ij}. Constraints (5.12) - (5.21) link edge weight variables w_{ij} and path length variables u_{ij}. Constraints (5.12) ensure a weak realization of the distance matrix. Constraints (5.13)

establish a lower bound for variables u_{ij}, associated to the length of the path between an additional node i and an external node j, to be the value of the corresponding (edge) weight w_{ij}. Constraints (5.14) fix w_{ij} to 0 when edge $\{i, j\}$ does not belong to the tree. Together with (5.13), constraints (5.15) impose that w_{ij} is equal to u_{ij} if edge $\{i, j\}$ belongs to the tree. Constraints (5.16)-(5.21) impose the triangle inequality for any order of the nodes i, j and k. The index specifications included in the constraints (5.16)-(5.21) improve the performance of our formulation as they impose an order on the variables index. Constraints (5.22) to (5.25) are integrality and non-negativity constraints.

Our formulation strengthens the Flow Model proposed in [27] by including lower bounds on the w_{ij} variables.

5.2 Path-edges formulation

The Path-edges formulation uses Pauplin's method to calculate the tree length, thus it minimizes the sum (2.2) presented in Section 2.3:

$$\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}}.$$

Using binary variables p_{ij}^{ℓ}, for all $i, j \in V_t$, $i < j$ and $\ell \in \{2, 3, ..., (n-1)\}$, specifying the number of edges of a path P_{ij} between terminal nodes i and j, the expression $\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}}$ can be linearized. The binary decision variables p_{ij}^{ℓ} indicate whether the path P_{ij} connecting terminal node i to terminal node j has (exactly) ℓ edges. Therefore $\sum_{i,j \in V_t} d_{ij}\ 2^{-z_{ij}} = \sum_{i,j \in V_t} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} p_{ij}^{\ell}$. This relation and linearization process was already used in [28] and variables p_{ij}^{ℓ} have the same interpretation as variables x_{ij}^{ℓ} in [28].

Besides these path variables p_{ij}^{ℓ}, the binary topology variables x_{ij} and the weight variables w_{ij}, we consider flow variables. The binary flow variables $f_{ij}^{k\ell}$, for all $i, j \in V_a \cup \{k, \ell\}, k, \ell \in V_t$, $i \neq j$ and $k < \ell$, indicate whether the flow traverses the edge $\{i, j\}$ belonging to the path connecting terminal node k to terminal node ℓ in the direction from node i to node j.

The formulation that specifies the number of edges of a path between terminal nodes and reconstructs an unrooted tree for the MWTR problem is as follows.

Path-edges formulation

$$min \quad \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \cdot p_{ij}^{\ell}$$

subject to

$$\sum_{i \in V_a} \sum_{\substack{j \in V \\ j>i}} x_{ij} = 2n - 3 \tag{5.26}$$

$$\sum_{i \in V_a} x_{ij} = 1, \qquad \forall j \in V_t \tag{5.27}$$

$$\sum_{\substack{j \in V \\ j>i}} x_{ij} + \sum_{\substack{j \in V_a \\ j<i}} x_{ji} = 3, \qquad \forall i \in V_a \tag{5.28}$$

$$x_{i,i+1} = 1, \qquad \forall i \in V_a,\ i = 1, ..., (\lceil n/2 \rceil - 1) \tag{5.29}$$

$$\sum_{j \in V_t} x_{1j} = 2 \tag{5.30}$$

$$\sum_{j \in V_t} x_{(n-2)j} = 2 \tag{5.31}$$

$$\sum_{j \in V_t} x_{ij} \leq 2, \qquad \forall i \in V_a \tag{5.32}$$

$$\sum_{i \in V_a} f_{ki}^{k\ell} = 1, \qquad \forall k, \ell \in V_t, k < \ell \tag{5.33}$$

$$\sum_{j \in \{\ell\} \cup V_a \setminus \{i\}} f_{ij}^{k\ell} - \sum_{j \in \{k\} \cup V_a \setminus \{i\}} f_{ji}^{k\ell} = 0, \qquad \forall i \in V_a, k, \ell \in V_t, k < \ell \tag{5.34}$$

$$\sum_{i \in V_a} f_{i\ell}^{k\ell} = 1, \qquad \forall k, \ell \in V_t, k < \ell \tag{5.35}$$

$$\sum_{h \in \{\ell\} \cup V_a \setminus \{i\}} f_{jh}^{k\ell} - f_{ij}^{k\ell} \geq 0, \qquad \forall i \in V_a \cup \{k\}, j \in V_a, k, \ell \in V_t, k < \ell \tag{5.36}$$

$$f_{ij}^{k\ell} + f_{ji}^{k\ell} \leq x_{ij}, \qquad \forall i, j \in V, \forall k, \ell \in V_t, i < j, k < \ell \tag{5.37}$$

$$\sum_{\ell=2}^{n-1} p_{ij}^{\ell} = 1, \qquad \forall i, j \in V_t, i < j \tag{5.38}$$

$$2 + \sum_{i \in V_a} \sum_{\substack{j \in V_a \\ j \neq i}} f_{ij}^{k\ell} = \sum_{i=2}^{n-1} i \cdot p_{k\ell}^{i}, \qquad \forall k, \ell \in V_t, k < \ell \tag{5.39}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in V_a, \forall j \in V, i < j \tag{5.40}$$

$$p_{ij}^{\ell} \in \{0, 1\}, \qquad \forall \ell \in \{2, 3, ..., n-1\}, \forall i, j \in V_t, i < j \tag{5.41}$$

$$f_{ij}^{k\ell} \in \{0, 1\}, \qquad \forall i, j \in V_a \cup \{k, \ell\}, \forall k, \ell \in V_t, i \neq j, k < \ell \tag{5.42}$$

Constraint (5.26) is the tree cardinality constraint and establishes that the number of edges in the tree is $2n - 3$. Constraints (5.27) establish that all the terminal nodes have degree one and constraints (5.28) force the additional nodes degree to be three. As above, we fix a path of additional nodes with constraints (5.29). Since the tree is unrooted, we know that there are two additional nodes which are adjacent to two terminal nodes. Therefore, to reduce symmetry, constraints (5.30) and (5.31) enforce those two additional nodes to be node 1 and node $(n-2)$. Constraints (5.32) impose that an additional node is connected to, at most, two terminal nodes. Constraints (5.33)–(5.35) are flow conservation constraints. Constraints (5.36) establish that if the flow sent from terminal node k to terminal node ℓ passes through edge $\{i, j\}$, in the direction from node i to node j, then the flow passes through, at least, one edge between node j and a node different than node i. Constraints (5.37) are linking constraints between variables and impose that there can be no flow in edge $\{i, j\}$ if it does not belong to the tree. Constraints (5.38) impose that variables p_{ij}^{ℓ} assume value one for exactly one ℓ in $\{2, 3, ..., (n-1)\}$ corresponding to the number of edges in the path between terminal nodes i and j. Constraints (5.39) relate variables $f_{ij}^{k\ell}$ with variables p_{ij}^{ℓ}. Using variables $f_{ij}^{k\ell}$ we know exactly the number of edges in the path between terminal nodes k and ℓ. Using variables $p_{k\ell}^{i}$ we also know exactly the number of edges in the path between terminal nodes k and ℓ and that number must be exactly the same. Constraints (5.40), (5.41) and (5.42) are the integrality constraints.

This second formulation uses the same idea as [28] to obtain the total tree edges length by the formulation designated by the authors as the Path-Length-4-Point (PL4). The PL4

formulation is considered by the authors of [28] as the current state-of-the-art algorithm for the BMEP. Therefore, in Section 5.4 we present the PL4 formulation and in Section 5.5 we compare the Path-weight formulation and the Path-edges formulation with the PL4 formulation.

After having reconstructed a unrooted tree with this formulation, the weights have to be assigned to the tree edges. This is accomplished by solving the following simple linear program.

$$
\begin{aligned}
& min \quad \sum_{i \in V_a} \sum_{\substack{j \in V \\ j>i}} w_{ij} \\
& subject\ to \\
& \quad \sum_{i \in V_a} \sum_{\substack{j \in V \\ i<j}} w_{ij}(f_{ij}^{k\ell} + f_{ji}^{k\ell}) \geq d_{k\ell} \qquad \forall k, \ell \in V_t, k < \ell \\
& \quad w_{ij} \geq 0 \qquad \forall i, j \in V, i < j
\end{aligned}
$$

Using the flow variables, the path between each pair of terminal nodes is exactly known. This information is used to associate weights to the edges such that the total sum of the edges weights is minimized and the tree length between every pair of terminal nodes dominates (is greater than) the corresponding distance from the distance matrix D.

5.3 Valid equalities and inequalities

The Path-edges formulation can be improved by considering some valid equalities and inequalities.

In a tree, there is exactly one path between every pair of terminal nodes and the path variables p_{ij}^{ℓ} state there are ℓ, for some unique $\ell \geq 2$, edges in the path between terminal nodes i and j. Therefore $\sum_{\ell=2}^{n-1} \ell p_{ij}^{\ell}$ is the number of edges in the path between terminal nodes i and j. By summing the number of edges of all paths and taking into account that

some edges belong to more than one path we must have

$$2 \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} \sum_{\ell=2}^{n-1} 2^{-\ell} \; \ell \; p_{ij}^{\ell} = 2n - 3, \tag{5.43}$$

as the number of edges in the tree is $2n - 3$. This is already stated in the formulation through constraint (5.26), however this equality reinforces this condition using the path variables.

Huffman codes are optimal path-length sequences whose corresponding rooted binary tree determines a code. When producing optimal Huffman codes the key idea are these rooted binary trees represented as sequences of ascending path-lengths. A sequence of n path-lengths represents a binary tree with n leaves where each leaf represents a symbol in the code. Parker and Ram [106] characterize these path-length sequences by establishing that these binary trees obey the property established by the Kraft equality [1], a special case of the Kraft inequality. These path-lengths can be compared to the distances from the tree realization problem. Therefore the nontrivial property of the path-length sequences in a rooted binary tree characterized with the Kraft equality can be borrowed by the MWTR problem and the following equality can be established:

$$\sum_{\ell=2}^{n-1} \sum_{\substack{j \in V_t \\ j > i}} 2^{-\ell} p_{ij}^{\ell} = \frac{1}{4} \qquad \forall i \in V_t \tag{5.44}$$

The inclusion of the two equalities (5.43) and (5.44) improved the performance of the Path-edges formulation.

Beside these two equalities the following valid inequalities presented in [28] can also be

[1] For all $n \geq 1$, let $\langle \ell_1, \ldots, \ell_n \rangle$ be the sequence of path-lengths in a rooted binary tree. The Kraft equality states that $\sum_{i=1}^{n} 2^{-\ell_i} = 1$

included.

$$\sum_{\substack{j\in V_t\\ i<j}} p_{ij}^{n-1} \leq 2 \sum_{\substack{j\in V_t\\ i<j}} p_{ij}^{\ell} \qquad \forall i \in V_t, \forall \ell \in \{2,3,...,n-2\} \tag{5.45}$$

$$\sum_{i\in V_t} \sum_{\substack{j\in V_t\\ j>i}} p_{ij}^{n-1} \leq 4 \tag{5.46}$$

$$\sum_{\substack{j\in V_t\\ i<j}} \sum_{q=2}^{\ell} 2^{\ell-q} p_{ij}^{q} \leq 2^{\ell-1} - 1 \forall i \in V_t, \forall \ell \in \{2,3,...,\lfloor \frac{n}{2} \rfloor\}, n > 2^{\ell-1}+1 \tag{5.47}$$

Inequalities (5.45) state that if a tree has a path of length $n-1$ then it also has a path of length $n-2, n-3, \ldots, 2$. Inequality (5.46) indicates that a tree has at most four paths of length $n-1$. Inequalities (5.47) are a consequence of the Kraft equality.

5.4 Path-Length-4-Point formulation

In this section, we present the Path-Length-4-Point formulation, defined in [28], which we used to compare the performance of two formulation, Path-weight formulation and Path-edges formulation.

The Path-Length-4-Point formulation uses the binary decision variables p_{ij}^{ℓ}, for all $i, j \in V$, $i \neq j$ and $\ell \in L$, $L = \{1,2,3,...,(n-1)\}$, specifying the number of edges of a path P_{ij} between nodes i and j and the binary decision variables:

$$y_{ijqt} = \begin{cases} 1 \quad \text{if} \quad z_{it} + z_{jq} \geq z_{iq} + z_{jt} \\ \\ 0 \quad \text{otherwise} \end{cases} \qquad \forall i,j,q,t \in V, i \neq j \neq q \neq t,$$

where z_{ij} indicates the number of edges in the path P_{ij} between nodes i and j.

The Path-Length-4-Point formulation is as follows.

Path-Length-4-Point formulation

$$min \quad \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j \neq i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \cdot p_{ij}^{\ell}$$

subject to

$$\sum_{\ell \in L} p_{ij}^{\ell} = 1, \qquad \forall i, j \in V, i \neq j \tag{5.48}$$

$$p_{ji}^{\ell} = p_{ij}^{\ell}, \qquad \forall i, j \in V, i < j, \forall k \in L \tag{5.49}$$

$$\sum_{\substack{j \in V_t \\ j \neq i}} \sum_{\ell=2}^{n-1} 2^{-\ell} p_{ij}^{\ell} = \frac{1}{2}, \qquad \forall i \in V_t \tag{5.50}$$

$$\sum_{\ell=2}^{n-1} \ell 2^{-\ell} \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j \neq i}} p_{ij}^{\ell} = 2n - 3 \tag{5.51}$$

$$\sum_{\ell \in L} \ell \left(p_{ij}^{\ell} + p_{qt}^{\ell} \right) \leq \sum_{\ell \in L} \ell \left(p_{iq}^{\ell} + p_{jt}^{\ell} \right) + (2n-2) y_{ijqt}, \qquad \forall i, j, q, t \in V, i \neq j \neq q \neq t \tag{5.52}$$

$$\sum_{\ell \in L} \ell \left(p_{ij}^{\ell} + p_{qt}^{\ell} \right) \leq \sum_{\ell \in L} \ell \left(p_{it}^{\ell} + p_{jq}^{\ell} \right) + (2n-2)(1 - y_{ijqt}), \quad \forall i, j, q, t \in V, i \neq j \neq q \neq t \tag{5.53}$$

$$p_{ij}^{1} = 0, \qquad \forall i, j \in V_t, i \neq j \tag{5.54}$$

$$\sum_{i \in V} \sum_{\substack{j \in V \\ j > i}} p_{ij}^{1} = 2n - 3 \tag{5.55}$$

$$\sum_{i \in V_a} p_{ij}^{1} = 1, \qquad \forall i \in V_t \tag{5.56}$$

$$\sum_{\substack{j \in V \\ j > i}} p_{ij}^{1} = 3, \qquad \forall i \in V_a \tag{5.57}$$

$$p_{ij}^{1} + p_{ik}^{1} + p_{kj}^{1} \leq 2, \qquad \forall i, j, k \in V_a, i \neq j \neq k \tag{5.58}$$

$$p_{ij}^{\ell} + 1 \geq p_{ik}^{\ell-1} + p_{kj}^{1}, \qquad \forall i,j \in V_t, i \neq j, \forall k \in V_a, \forall \ell \in L \setminus \{1, n-1\} \tag{5.59}$$

$$p_{ij}^{\ell} + p_{ij}^{\ell-2} + 1 \geq p_{ik}^{\ell-1} + p_{kj}^{1}, \qquad \forall i,j,k \in V, i \neq j \neq k, \forall \ell \in L \setminus \{1, 2, n-1\} \tag{5.60}$$

$$p_{ij}^{\ell} \in \{0,1\}, \qquad \forall \ell \in L, \forall i,j \in V \tag{5.61}$$

$$y_{ijqt} \in \{0,1\}, \qquad \forall i,j,q,t \in V \tag{5.62}$$

Constraints (5.48) impose that variables z_{ij} assume exactly on value in L. Constraints (5.49) impose symmetry equalities. Constraints (5.50) impose the Kraft equalities. Constraints (5.51) impose that the number of edges in the tree is $2n-3$.Constraints (5.52) and (5.53) impose the four-point inequalities. Constraints (5.54)–(5.60) describe the structure of the tree. Constraints (5.61) and (5.62) are integrality constraints.

5.5 Computational results

Computational results will assess the quality of the Linear Programming (LP) solutions obtained with each formulation and the best lower and upper bounds achieved. The computational tests were performed on an Intel(R) Core(TM) i7-3770 CPU 3.40 GHz processor and 16Gb of RAM.

The two formulations were implemented using the Mosel language and solved with FICO Xpress 7.8 [1] (Xpress-IVE 1.24.06 64 bit, Xpress-Optimizer 27.01.02 and Xpress-Mosel 3.8.0). Path-edges formulation used together with the valid equalities (5.43) and (5.44) and inequalities (5.45), (5.46), (5.47) presented in Section 5.3 is designated Path-edges^{+} formulation. We compare the performance of the two formulations, Path-weight formulation and Path-edges formulation, and the Path-edges^{+} formulation with the formulation PL4 from [28]. Our implementation of PL4 considers all the valid inequalities presented in [28].

We used the instances presented in Chapter 3 and Chapter 4.

The computational results are summarized in Tables 5.1 - 5.11 in which the first column, labeled **Matrix**, indicates the name of the matrix instance used and the second column,

labeled $|V_t|$, indicates the size of the instance. The third, forth and fifth columns concern the results of the Path-weight formulation, from the sixth to the eleventh columns the results of the Path-edges formulation are presented, from the twelfth to the seventeenth columns are the results of the Path-edges$^+$ formulation and the eighteenth, nineteenth, twentieth and twenty-first columns concern PL4 formulation. The columns labeled **T** show the execution time, in seconds, used to solve the instance and having a maximum runtime of 7200 seconds and for the Path-edges formulation and Path-edges$^+$ formulation, the columns labeled $\mathbf{T_w}$ shows the execution time of the linear program solved to assign the weights. The columns labeled **W** and **DZ** present the optimum value obtained or the best value obtained having a runtime limit of 7200 seconds, where DZ stands for $\sum_{k \in V_t} \sum_{\substack{\ell \in V_t \\ \ell > k}} d_{k\ell} \sum_{i=2}^{n-1} 2^{-i} \cdot p^i_{k\ell}$ and W stands for $\sum_{i \in V_a} \sum_{\substack{j \in V \\ j > i}} w_{ij}$. The columns labeled **GAP** present the LP solution gap which is obtained as follows: $GAP = \dfrac{UB - LP}{UB} \times 100$, where UB represents the best upper bound value obtained (or the optimum value) within the runtime of 7200 seconds and LP represents the value of the corresponding linear programming relaxation. The columns labeled $\mathbf{GAP_{LB}}$ present the lower bound gap and is obtained as follows: $GAP_{LB} = \dfrac{UB - LB}{UB} \times 100$, where UB represents the best upper bound value obtained (or the optimum value) and LB the best lower bound value, both values obtained within the runtime of 7200 seconds.

Table 5.1: Computational results for data from the phylogenetics application.

		Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
Matrix	$\lvert V_t\rvert$	**T**	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
M391	5	0.22	0.0486	0	0.09	0.0473	31.5	0	0	0.0487	0.02	0.0473	0	0	0	0.0487	0.58	0.0473	0	0
	6	0.5	0.0583	0	0.52	0.057	58.8	0	0	0.0592	0.05	0.057	0.3	0	0.01	0.0592	33.24	0.057	0.1	0
	7	2.28	0.0626	0	3.81	0.0607	73.4	0	0.02	0.0634	2	0.0608	0.8	0	0.02	0.0634	328.02	0.0607	0.4	0
	8	73.98	0.0693	0	33.12	0.0672	82.7	0	0.05	0.0698	21.29	0.0672	0.7	0	0.05	0.0698	> 7200	–	–	–
	9	1003.18	0.0948	0	547.06	0.0923	90.1	0	0.16	0.0952	11.06	0.0923	0.6	0	0.16	0.0952	> 7200	–	–	–
	10	> 7200	0.1061	60.8	> 7200	0.1028	94.3	29.1	0.61	0.1073	557.26	0.1027	1	0	1.06	0.1068	> 7200	–	–	–
	11	> 7200	0.1326	83.4	> 7200	0.1276	96.8	49.4	16.57	0.1338	2837.07	0.1276	1.2	0	2.14	0.1338	> 7200	–	–	–
	12	> 7200	0.1386	–	> 7200	0.135	98.2	79.2	35.33	0.1433	> 7200	0.1322	1.2	0.9	5.06	0.1378	> 7200	–	–	–
	13	> 7200	0.1555	–	> 7200	0.1517	99	95.9	55.57	0.1601	> 7200	0.1445	1.8	1.7	58.36	0.151	> 7200	–	–	–
	14	> 7200	0.1607	–	> 7200	0.1631	99.4	99.1	158.7	0.1739	> 7200	0.1505	3.9	3.8	139.34	0.16	> 7200	–	–	–
	15	> 7200	0.1733	–	> 7200	0.1836	99.7	99.7	3506.64	0.1959	> 7200	0.1601	6.3	6.2	912.76	0.1714	> 7200	–	–	–
Primate	5	0.25	0.0843	0	0.06	0.0823	31.9	0	0	0.0843	0	0.0823	0	0	0	0.0843	1.42	0.0823	0	0
	6	0.53	0.1059	0	0.48	0.1023	61.2	0	0.02	0.1059	0.01	0.1023	0	0	0.02	0.1059	67.91	0.1023	0	0
	7	1.84	0.1272	0	2.54	0.1232	72.5	0	0.02	0.1272	0.25	0.1232	0.4	0	0.02	0.1272	507.91	0.1232	0	0
	8	25.94	0.128	0	13.31	0.1230	80.3	0	0.05	0.128	1.59	0.1230	0.4	0	0.05	0.128	> 7200	–	–	–
	9	228.7	0.1296	0	116.13	0.1246	87.1	0	0.16	0.1296	14.18	0.1246	3	0	0.16	0.1296	> 7200	–	–	–
	10	> 7200	0.1297	6.5	5624.82	0.122	92	0	0.3	0.1298	91.68	0.122	2.5	0	1	0.1298	> 7200	–	–	–
	11	> 7200	0.17	72.4	> 7200	0.165	95.9	45.2	15.93	0.1722	598.11	0.1625	2.3	0	28.24	0.1700	> 7200	–	–	–
	12	> 7200	0.2076	96.7	> 7200	0.1986	97.6	70.1	33.9	0.2084	> 7200	0.1959	2.2	1.5	70.01	0.2058	> 7200	–	–	–
M887	5	0.22	0.1033	0	0.09	0.0972	29.6	0	0	0.1033	0.01	0.0972	0	0	0	0.1033	0.13	0.0972	0	0
	6	0.5	0.1157	0	0.41	0.1098	57.6	0	0.01	0.1157	0.61	0.1098	1.9	0	0.02	0.1157	64.79	0.1098	0.4	0
	7	2.12	0.1364	0	3.21	0.1292	71.6	0	0.02	0.1364	1.34	0.1292	1.3	0	0.01	0.1364	608.92	0.1292	0	0
	8	63.95	0.1756	0	24.84	0.1619	82.7	0	0.05	0.1763	17.43	0.1619	1.5	0	0.05	0.1763	> 7200	0.188	14.1	14
	9	742.38	0.1823	0	315.62	0.1685	89.4	0	0.17	0.186	13.56	0.1685	1.7	0	0.16	0.1860	> 7200	–	–	–
	10	> 7200	0.2026	55.7	> 7200	0.1859	93.8	22.2	4.49	0.2059	178.87	0.1859	2.1	0	17.67	0.2059	> 7200	–	–	–
	11	> 7200	0.2113	81.2	> 7200	0.1946	96.5	53.9	16.77	0.2147	2085.04	0.1938	2.6	0	30.09	0.2135	> 7200	–	–	–
	12	> 7200	0.2184	–	> 7200	0.2039	98	79.5	38.42	0.2237	> 7200	0.1968	2.7	2.6	64.63	0.2146	> 7200	–	–	–
	13	> 7200	0.2395	–	> 7200	0.2255	98.9	94	63.48	0.2486	> 7200	0.2154	3.6	3.5	128.42	0.238	> 7200	–	–	–
	14	> 7200	0.2512	–	> 7200	0.2250	99.4	99	138.39	0.2479	> 7200	0.2254	6.1	6	315.12	0.2527	> 7200	–	–	–
	15	> 7200	0.2787	–	> 7200	0.2909	99.7	99.7	411.84	0.3249	> 7200	0.2557	10.7	10.7	589.14	0.2844	> 7200	–	–	–

Table 5.2: Computational results for data from the phylogenetics application generated with $a = 0.1$.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
A10M391	5	0.23	0.0513	0	0.08	0.0498	31.6	0	0	0.0515	0	0.0498	0	0	0.02	0.0515	0.23	0.0498	0	0
	6	0.48	0.0615	0	0.44	0.0599	58.7	0	0.02	0.0622	0.06	0.0599	0.4	0	0.01	0.0622	3.79	0.0599	0.1	0
	7	2.29	0.0661	0	4.41	0.0639	73.4	0	0.02	0.0668	2.42	0.0639	0.8	0	0.02	0.0668	54.13	0.0639	0.4	0
	8	73.06	0.0729	0	32.79	0.0707	82.7	0	0.03	0.0734	14.87	0.0707	0.7	0	0.03	0.0734	> 7200	0.0793	11.2	11.1
	9	1008.93	0.0994	0	531.65	0.0968	90.1	0	0.16	0.0998	9.89	0.0968	0.6	0	0.16	0.0998	> 7200	–	–	–
	10	> 7200	0.112	58	> 7200	0.1082	94.3	32.7	1.01	0.1124	197.67	0.1082	1.1	0	1.04	0.1124	> 7200	–	–	–
	11	> 7200	0.1399	83.4	> 7200	0.1345	96.8	49.3	17.74	0.1410	1197.99	0.1345	1.1	0	2.31	0.141	> 7200	–	–	–
	12	> 7200	0.1465	–	> 7200	0.1418	98.2	79.3	36.21	0.1494	> 7200	0.1394	1.4	1.1	5.38	0.1461	> 7200	–	–	–
	13	> 7200	0.1603	–	> 7200	0.1581	99	94.5	64.58	0.1689	> 7200	0.1519	1.8	1.7	120.17	0.1595	> 7200	–	–	–
	14	> 7200	0.1709	–	> 7200	0.1702	99.4	99.1	138.73	0.1856	> 7200	0.1568	3.1	3	330.19	0.167	> 7200	–	–	–
	15	> 7200	0.1837	–	> 7200	0.195	99.7	99.5	334.09	0.2074	> 7200	0.1719	8.1	8.1	1376.94	0.1838	> 7200	–	–	–
A10Pri	5	0.11	0.0893	0	0.08	0.0864	31.9	0	0.01	0.0893	0	0.0864	0	0	0.01	0.0893	0.17	0.0864	0	0
	6	0.41	0.1115	0	0.36	0.1076	61.3	0	0.02	0.1115	0.05	0.1076	0	0	0	0.1115	62.56	0.1076	0	0
	7	1.86	0.1343	0	3.06	0.1296	72.6	0	0.01	0.1343	0.42	0.1296	0.5	0	0.02	0.1343	43.01	0.1296	0	0
	8	26.94	0.1351	0	13.4	0.1294	80.3	0	0.03	0.1351	1.89	0.1294	0.4	0	0.05	0.1351	> 7200	0.1728	25.1	25.1
	9	227	0.1363	0	117.41	0.1311	87.1	0	0.16	0.1364	12.93	0.1311	3.1	0	0.16	0.1364	> 7200	–	–	–
	10	> 7200	0.1360	5.3	5562.55	0.1283	92	0	0.31	0.1361	138.33	0.1283	2.4	0	9.44	0.1361	> 7200	–	–	–
	11	> 7200	0.1805	74.1	> 7200	0.1709	95.8	44.7	14.85	0.1794	574.74	0.1709	2.4	0	2.26	0.1794	> 7200	–	–	–
	12	> 7200	0.2203	97.1	> 7200	0.2111	97.6	69.4	35.19	0.2203	> 7200	0.2062	2.2	1.7	30.22	0.2162	> 7200	–	–	–
A10M887	5	0.39	0.109	0	0.09	0.1023	29.6	0	0.02	0.109	0.01	0.1023	0	0	0	0.109	0.22	0.1023	0	0
	6	0.41	0.1215	0	0.45	0.1155	57.6	0	0.02	0.1215	0.66	0.1155	1.7	0	0	0.1215	42.17	0.1155	0.3	0
	7	2.12	0.1432	0	3.01	0.1357	71.5	0	0.02	0.1432	1.51	0.1357	1.3	0	0.02	0.1432	1045.01	0.1357	0	0
	8	63.71	0.1841	0	25.02	0.1702	82.7	0	0.03	0.1858	9.45	0.1702	1.5	0	0.03	0.1858	> 7200	0.1935	12.3	12.1
	9	715.67	0.1922	0	335.18	0.1769	89.4	0	0.16	0.1972	9.23	0.1769	1.6	0	0.16	0.1972	> 7200	–	–	–
	10	> 7200	0.2129	57.1	> 7200	0.1962	93.8	26.2	0.59	0.2184	276.4	0.1955	2.1	0	1.06	0.2156	> 7200	–	–	–
	11	> 7200	0.2203	81.4	> 7200	0.2065	96.5	58.6	17.25	0.2274	2253.32	0.2038	2.6	0	27.08	0.2243	> 7200	–	–	–
	12	> 7200	0.2307	–	> 7200	0.212	98	79.4	42.07	0.236	> 7200	0.2066	2.6	2.5	65.77	0.229	> 7200	–	–	–
	13	> 7200	0.2526	–	> 7200	0.2374	98.9	94	60.59	0.2653	> 7200	0.2263	3.6	3.5	57.02	0.2454	> 7200	–	–	–
	14	> 7200	0.2628	–	> 7200	0.2542	99.4	99	146.02	0.2847	> 7200	0.2394	7	7	10.65	0.2668	> 7200	–	–	–
	15	> 7200	0.2888	–	> 7200	0.2974	99.7	99.7	340.47	0.3267	> 7200	0.2589	7.1	7.1	1105.45	0.2857	> 7200	–	–	–

Table 5.3: Computational results for data from the phylogenetics application generated with $a = 0.15$.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
A15M391	5	0.27	0.0526	0	0.09	0.051	31.6	0	0.01	0.0529	0.01	0.051	0	0	0	0.0529	0.25	0.051	0	0
	6	0.41	0.0631	0	0.39	0.0613	58.7	0	0	0.0637	0.08	0.0613	0.4	0	0	0.0637	4.01	0.0613	0.1	0
	7	2.34	0.0679	0	2.67	0.0654	73.4	0	0.02	0.0685	1.37	0.0654	0.8	0	0.02	0.0685	57.22	0.0654	0.4	0
	8	74.05	0.0747	0	33.23	0.0724	82.7	0	0.03	0.0752	16.43	0.0724	0.7	0	0.05	0.0752	> 7200	0.0813	11.2	11.2
	9	1007.39	0.1019	0	481.42	0.099	90.1	0	0.16	0.1021	13.37	0.099	0.6	0	0.16	0.1021	> 7200	–	–	–
	10	> 7200	0.1149	57.5	> 7200	0.111	94.3	28.1	0.61	0.1153	211.55	0.111	1.1	0	28.38	0.1153	> 7200	–	–	–
	11	> 7200	0.1431	83.2	> 7200	0.1387	96.8	53	16.51	0.1442	2032.65	0.138	1.1	0	2.32	0.1433	> 7200	–	–	–
	12	> 7200	0.1509	–	> 7200	0.1443	98.2	79.4	32.79	0.1523	> 7200	0.1428	1.4	1.1	69.62	0.1492	> 7200	–	–	–
	13	> 7200	0.1658	–	> 7200	0.1598	98.9	94.4	63.87	0.1668	> 7200	0.156	2	1.9	65.8	0.1642	> 7200	–	–	–
	14	> 7200	0.1715	–	> 7200	0.1725	99.4	99.1	176.58	0.1851	> 7200	0.1616	3.7	3.6	10	0.1713	> 7200	–	–	–
	15	> 7200	0.182	–	> 7200	0.1834	99.7	99.5	220.44	0.1963	> 7200	0.1681	3.8	3.7	684.61	0.1799	> 7200	–	–	–
A15Pri	5	0.08	0.0917	0	0.08	0.0885	31.9	0	0	0.0917	0.02	0.0885	0	0	0	0.0917	0.17	0.0885	0	0
	6	0.42	0.1143	0	0.37	0.1103	61.3	0	0	0.1143	0.02	0.1103	0	0	0.01	0.1143	52.95	0.1103	0	0
	7	1.81	0.1379	0	2.68	0.1328	72.6	0	0.01	0.1379	0.67	0.1328	0.6	0	0.02	0.1379	647.85	0.1328	0	0
	8	26.77	0.1386	0	12.37	0.1327	80.4	0	0.03	0.1386	2.21	0.1327	0.4	0	0.03	0.1386	> 7200	0.1771	25.1	25.1
	9	218.6	0.1397	0	136.06	0.1343	87.1	0	0.16	0.1397	12.68	0.1343	3.1	0	0.17	0.1397	> 7200	–	–	–
	10	> 7200	0.1392	11.9	5670.35	0.1315	92	0	0.31	0.1393	185.83	0.1315	2.4	0	17.38	0.1393	> 7200	–	–	–
	11	> 7200	0.1852	73.2	> 7200	0.1754	95.8	44.6	16.35	0.1852	322.87	0.1751	2.4	0	2.29	0.1841	> 7200	–	–	–
	12	> 7200	0.2247	97.5	> 7200	0.2143	97.6	65.5	31.51	0.223	> 7200	0.2113	2.2	1.7	70.37	0.2214	> 7200	–	–	–
A15M887	5	0.36	0.1118	0	0.11	0.1049	29.7	0	0	0.1118	0.01	0.1049	0	0	0	0.1118	0.22	0.1049	0	0
	6	0.58	0.1245	0	0.37	0.1183	57.6	0	0.02	0.1245	0.47	0.1183	1.6	0	0.01	0.1245	20.97	0.1183	0.2	0
	7	2.12	0.1466	0	3.28	0.1389	71.5	0	0.02	0.1466	1.86	0.1389	1.3	0	0.02	0.1466	694.06	0.1389	0	0
	8	64.37	0.1884	0	24.88	0.1744	82.7	0	0.05	0.1905	11.61	0.1744	1.5	0	0.05	0.1905	> 7200	0.1983	12.3	12.2
	9	727.04	0.1968	0	329.47	0.1811	89.4	0	0.16	0.202	9.7	0.1811	1.6	0	0.16	0.2020	> 7200	–	–	–
	10	> 7200	0.2181	55.8	> 7200	0.2003	93.8	27.5	0.9	0.2205	128.84	0.2003	2.1	0	16.38	0.2205	> 7200	–	–	–
	11	> 7200	0.2259	80.8	> 7200	0.2096	96.5	54.1	16.88	0.2309	1881.22	0.2088	2.7	0	2.17	0.2297	> 7200	–	–	–
	12	> 7200	0.2322	–	> 7200	0.2158	98	79	33.71	0.2376	> 7200	–	–	–	–	–	> 7200	–	–	–
	13	> 7200	0.2595	–	> 7200	0.2424	98.9	93.9	56.46	0.2688	> 7200	0.2329	4.2	4	5.15	0.2527	> 7200	–	–	–
	14	> 7200	0.2715	–	> 7200	0.2544	99.4	99	121.27	0.2798	> 7200	0.2442	6.7	6.6	587.70	0.2714	> 7200	–	–	–
	15	> 7200	0.3011	–	> 7200	0.2922	99.7	99.5	284.01	0.3172	> 7200	0.2782	11.4	11.3	637.39	0.3050	> 7200	–	–	–

Table 5.4: Computational results for data from the phylogenetics application generated with $a = 0.20$.

		Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
Matrix	$\lvert V_t \rvert$	**T**	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
A20M391	5	0.25	0.0539	0	0.11	0.0522	31.6	0	0	0.0543	0	0.0522	0	0	0	0.0543	0.23	0.0522	0	0
	6	0.22	0.0646	0	0.64	0.0627	58.7	0	0.02	0.0652	0.06	0.0627	0.5	0	0.02	0.0652	4.35	0.0627	0.1	0
	7	2.33	0.0696	0	4.46	0.067	73.5	0	0.03	0.0702	1.4	0.067	0.9	0	0.02	0.0702	1069.77	0.067	0.5	0
	8	77.27	0.0764	0	33.09	0.0741	82.7	0	0.05	0.0770	11.59	0.0741	0.7	0	0.03	0.0770	> 7200	0.0833	11.3	11.2
	9	1025.87	0.1043	0	528.61	0.1013	90.1	0	0.16	0.1044	10.39	0.1013	0.6	0	0.16	0.1044	> 7200	–	–	–
	10	> 7200	0.1177	59.8	> 7200	0.1138	94.3	29.9	3.95	0.1181	274.01	0.1138	1.1	0	1.09	0.1181	> 7200	–	–	–
	11	> 7200	0.1464	82.5	> 7200	0.1425	96.8	54.3	18.91	0.1475	1793.47	0.1414	1.1	0	2.17	0.1467	> 7200	–	–	–
	12	> 7200	0.1545	–	> 7200	0.1488	98.2	79.2	32.79	0.1578	> 7200	0.1463	1.5	1	44.32	0.1515	> 7200	–	–	–
	13	> 7200	0.1724	–	> 7200	0.1639	98.9	94.5	56.86	0.1773	> 7200	0.1596	1.9	1.9	5.19	0.1686	> 7200	–	–	–
	14	> 7200	0.1782	–	> 7200	0.1785	99.4	99.1	154.18	0.1891	> 7200	0.1636	2.7	2.6	525.08	0.1716	> 7200	–	–	–
	15	> 7200	0.1821	–	> 7200	0.1947	99.7	99.7	247.92	0.2078	> 7200	0.1777	6.8	6.7	644.3	0.1883	> 7200	–	–	–
A20Pri	5	0.25	0.0942	0	0.08	0.0906	31.8	0	0	0.0942	0.01	0.0906	0	0	0	0.0942	0.17	0.0906	0	0
	6	0.70	0.1171	0	0.33	0.113	61.3	0	0.02	0.1171	0.05	0.113	0	0	0	0.1171	148.59	0.113	0	0
	7	1.73	0.1414	0	3.15	0.136	72.6	0	0.02	0.1414	0.89	0.136	0.7	0	0.03	0.1414	600.69	0.136	0	0
	8	28.49	0.1422	0	14.15	0.1359	80.4	0	0.05	0.1422	0.92	0.1359	0.4	0	0.03	0.1422	> 7200	0.1813	25	25
	9	233.03	0.143	0	117.27	0.1375	87.1	0	0.16	0.1431	9.08	0.1375	3.2	0	0.16	0.1431	> 7200	–	–	–
	10	> 7200	0.1425	12	5623.51	0.1346	92	0	0.31	0.1427	119.22	0.1346	2.3	0	1.03	0.1427	> 7200	–	–	–
	11	> 7200	0.1899	72.3	> 7200	0.1809	95.8	44.9	16.77	0.1905	569.43	0.1793	2.5	0	2.36	0.1889	> 7200	–	–	–
	12	> 7200	0.2362	97.2	> 7200	0.2207	97.6	70.4	35.65	0.2307	> 7200	0.2164	2.2	1.5	5.18	0.2265	> 7200	–	–	–
A20M887	5	0.14	0.1147	0	0.09	0.1075	29.7	0	0	0.1147	0.02	0.1075	0	0	0	0.1147	0.23	0.1075	0	0
	6	0.36	0.1274	0	0.45	0.1211	57.6	0	0.02	0.1274	0.75	0.1211	1.6	0	0	0.1274	92.9	0.1211	0.1	0
	7	2.14	0.15	0	2.98	0.1421	71.5	0	0.02	0.15	1.64	0.1421	1.3	0	0.02	0.15	647.6	0.1421	0	0
	8	64.99	0.1926	0	22.98	0.1785	82.7	0	0.05	0.1953	11.89	0.1785	1.5	0	0.05	0.1953	> 7200	0.2032	12.4	12.3
	9	762.47	0.2014	0	303.09	0.1853	89.4	0	0.16	0.2067	9.02	0.1853	1.5	0	0.17	0.2067	> 7200	–	–	–
	10	> 7200	0.2233	58.9	> 7200	0.2051	93.8	25.7	0.58	0.2253	120.14	0.2051	2	0	1.03	0.2253	> 7200	–	–	–
	11	> 7200	0.2316	80.9	> 7200	0.2138	96.5	58.4	17.41	0.2352	2079.15	0.2138	2.7	0	2.15	0.2352	> 7200	–	–	–
	12	> 7200	0.2410	–	> 7200	0.2208	98	78.9	33.31	0.2381	> 7200	–	–	–	–	–	> 7200	–	–	–
	13	> 7200	0.2698	–	> 7200	0.2401	98.9	93.1	75.33	0.2669	> 7200	0.2375	3.8	3.7	68.08	0.2639	> 7200	–	–	–
	14	> 7200	0.2762	–	> 7200	0.2593	99.4	99	135.56	0.2842	> 7200	0.2482	6	6	369.75	0.2728	> 7200	–	–	–
	15	> 7200	0.3073	–	> 7200	0.3218	99.7	99.7	312.66	0.3576	> 7200	0.2774	8.9	8.9	796.1	0.311	> 7200	–	–	–

Table 5.5: Computational results for data from the phylogenetics application generated with $a = 1$.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
A100M391	5	0.19	0.0748	0	0.09	0.0721	31.7	0	0	0.0748	0	0.0721	0	0	0	0.0748	0.52	0.0721	0.1	0
	6	0.44	0.0872	0	0.34	0.0847	58.1	0	0.02	0.0882	0.08	0.0847	0.4	0	0	0.0882	186.86	0.0847	0	0
	7	2.25	0.0953	0	3.93	0.0920	73.5	0	0.02	0.0968	1.65	0.0920	1.2	0	0.02	0.0968	2006.88	0.092	0.7	0
	8	73.8	0.104	0	31.89	0.1007	82.5	0	0.03	0.1044	9.64	0.1007	0.7	0	0.05	0.1044	> 7200	0.1152	12.6	12.6
	9	988.37	0.1425	0	489.92	0.1373	90	0	0.16	0.143	13.6	0.1373	1.1	0	0.16	0.143	> 7200	–	–	–
	10	> 7200	0.1631	59.8	> 7200	0.1579	94.4	32.4	4.60	0.1631	279.43	0.1579	1.3	0	17.94	0.1631	> 7200	–	–	–
	11	> 7200	0.2043	83.3	> 7200	0.1973	96.8	54.3	17.03	0.2055	3594	0.1958	1.6	0	29.16	0.2052	> 7200	–	–	–
	12	> 7200	0.2124	99.3	> 7200	0.2048	98.2	79.3	39.64	0.2178	> 7200	0.2004	1.9	1.6	32.31	0.2097	> 7200	–	–	–
	13	> 7200	0.2312	–	> 7200	0.2296	99	94.4	64.71	0.2519	> 7200	0.2194	2.5	2.5	5.16	0.2358	> 7200	–	–	–
	14	> 7200	0.2448	–	> 7200	0.2429	99.4	99.1	159.35	0.2642	> 7200	0.2235	3	2.9	630.09	0.2369	> 7200	–	–	–
	15	> 7200	0.2653	–	> 7200	0.2818	99.7	99.7	256.48	0.3037	> 7200	0.2409	5.8	5.8	427.81	0.2616	> 7200	–	–	–
A100Pri	5	0.11	0.1338	0	0.08	0.1236	31.8	0	0	0.1338	0	0.1236	0	0	0	0.1338	0.19	0.1236	0	0
	6	0.56	0.1618	0	0.37	0.1558	61.7	0	0	0.1618	0.05	0.1558	0.5	0	0.02	0.1618	4.59	0.1558	0.2	0
	7	1.83	0.1966	0	3.04	0.1869	72.8	0	0.02	0.1966	2.18	0.1869	1.4	0	0.02	0.1966	35.86	0.1869	0.6	0
	8	32.95	0.1991	0	14.52	0.1877	80.5	0	0.05	0.1991	1.93	0.1877	0.5	0	0.05	0.1991	> 7200	0.2493	24.7	24.7
	9	243.56	0.202	0	136.78	0.1889	87.1	0	0.16	0.2035	9.77	0.1889	3.5	0	0.16	0.2035	> 7200	–	–	–
	10	> 7200	0.1962	3.7	6007.57	0.1848	92	0	0.31	0.1966	94.63	0.1848	2.2	0	1	0.1966	> 7200	–	–	–
	11	> 7200	0.264	73.1	> 7200	0.246	95.8	41.8	14.98	0.2655	1008.31	0.246	3.2	0	33.77	0.2655	> 7200	–	–	–
	12	> 7200	0.3356	97.9	> 7200	0.3076	97.6	70.8	32.99	0.3303	> 7200	0.2983	2.7	2.2	72.01	0.3166	> 7200	–	–	–
A100M887	5	0.34	0.1591	0	0.09	0.1487	30.1	0	0	0.1601	0	0.1487	0	0	0	0.1601	0.25	0.1487	0	0
	6	0.47	0.1756	0	0.56	0.1664	57.7	0	0.02	0.1756	0.08	0.1664	0.9	0	0.02	0.1756	85.92	0.1664	0	0
	7	2.18	0.2065	0	1.89	0.1934	71.5	0	0.01	0.2065	2.45	0.1934	1.7	0	0.02	0.2065	206.11	0.1934	0	0
	8	62.2	0.2646	0	22.6	0.2451	82.7	0	0.03	0.2714	2.75	0.2451	1.7	0	0.05	0.2714	> 7200	0.2805	13.1	12.9
	9	687.12	0.2762	0	298.44	0.2525	89.3	0	0.17	0.2833	8.08	0.2525	1.3	0	0.16	0.2833	> 7200	–	–	–
	10	> 7200	0.3064	56	> 7200	0.2822	93.8	27.6	0.9	0.3064	144.63	0.2822	2	0	0.98	0.3064	> 7200	–	–	–
	11	> 7200	0.3231	80.6	> 7200	0.2967	96.5	53.9	16.85	0.3273	3532.92	0.2935	3.4	0	2.28	0.3177	> 7200	–	–	–
	12	> 7200	0.3322	–	> 7200	0.3023	98	79.1	37.72	0.333	> 7200	0.2953	2.8	2.6	31.15	0.3265	> 7200	–	–	–
	13	> 7200	0.3589	–	> 7200	0.3376	98.9	93.6	53.68	0.3756	> 7200	0.3247	4.3	4.2	5.12	0.3604	> 7200	–	–	–
	14	> 7200	0.3807	–	> 7200	0.3651	99.4	99	125.17	0.403	> 7200	0.3340	4.9	4.8	495.15	0.3759	> 7200	–	–	–
	15	> 7200	0.4204	–	> 7200	0.4325	99.7	99.5	363.45	0.4881	> 7200	0.3909	11.2	11.1	18.35	0.446	> 7200	–	–	–

Table 5.6: Computational results for data from the networking application.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
	5	0.22	0.4043	0	0.08	0.4043	28.1	0	0.02	0.4043	0.02	0.4043	0	0	0	0.4043	0.16	0.4043	0	0
S7	6	0.28	0.4348	0	0.39	0.4347	49.9	0	0	0.4348	0.05	0.4347	0	0	0.02	0.4348	23.85	0.4347	0	0
	7	1.87	0.5055	0	3.21	0.5054	67.9	0	0.02	0.5055	0.05	0.5054	0	0	0.02	0.5055	4181.93	0.5054	0	0
	5	0.30	0.1893	0	0.09	0.1893	34.4	0	0	0.1893	0	0.1893	0	0	0.02	0.1893	0.12	0.1893	0	0
	6	0.26	0.2756	0	0.62	0.2756	57.3	0	0	0.2756	0.02	0.2756	0	0	0.01	0.2756	118.09	0.2756	0	0
	7	1.29	0.3013	0	3.1	0.3013	66.5	0	0.02	0.3013	0.11	0.3013	1.6	0	0.02	0.3013	8.06	0.3013	0	0
	8	31.69	0.3778	0	10.8	0.3778	79.3	0	0.03	0.3778	7.29	0.3778	2.7	0	0.05	0.3778	> 7200	–	–	–
	9	309.52	0.3932	0	75.18	0.3932	86.5	0	0.16	0.3932	8.74	0.3932	2.6	0	0.16	0.3932	> 7200	–	–	–
S15	10	> 7200	0.4286	33.4	3926.42	0.4286	91.8	0	0.31	0.4286	119.56	0.4286	3.2	0	1.05	0.4286	> 7200	–	–	–
	11	> 7200	0.626	60.3	> 7200	0.626	95.9	43.7	16.79	0.6260	1317.44	0.626	2.2	0	2.15	0.626	> 7200	–	–	–
	12	> 7200	0.722	94.7	> 7200	0.7271	97.6	66.3	34.13	0.7322	> 7200	0.722	1.9	1.7	64.26	0.7220	> 7200	–	–	–
	13	> 7200	0.8085	–	> 7200	0.7777	98.5	91.3	57.03	0.8079	> 7200	0.7474	7.1	1.7	43.4	0.7474	> 7200	–	–	–
	14	> 7200	0.8845	–	> 7200	0.9288	99.2	98.7	124.66	1.0278	> 7200	0.7779	8	6.4	10.27	0.7831	> 7200	–	–	–
	15	> 7200	0.9804	–	> 7200	1.0731	99.6	99.3	303.17	1.1738	> 7200	0.9302	10.2	9.6	1669.59	0.9813	> 7200	–	–	–
	5	0.2	0.299	0	0.08	0.299	35.6	0	0.01	0.299	0	0.299	0	0	0.01	0.299	0.11	0.299	0	0
	6	0.19	0.3394	0	0.34	0.3395	57.4	0	0.02	0.3395	0.01	0.3395	0	0	0.02	0.3395	0.84	0.3394	0	0
	7	1.83	0.3751	0	2.4	0.3751	70.7	0	0.02	0.3751	0.3	0.3751	3.7	0	0.02	0.3751	8.97	0.3751	3.2	0
	8	46.72	0.4564	0	22.14	0.4564	81.8	0	0.03	0.4564	7.41	0.4564	3.5	0	0.03	0.4564	> 7200	–	–	–
	9	581.09	0.5174	0	255.67	0.5174	88.4	0	0.16	0.5174	15.85	0.5174	3.3	0	0.16	0.5174	> 7200	–	–	–
S20	10	> 7200	0.5736	48.8	> 7200	0.5736	93	16.5	0.67	0.5736	172.71	0.5736	2.9	0	1.08	0.5736	> 7200	–	–	–
	11	> 7200	0.609	70	> 7200	0.609	95.7	46.9	20.31	0.609	2039.97	0.609	2.7	0	25.44	0.609	> 7200	–	–	–
	12	> 7200	0.6347	98.4	> 7200	0.6424	97.6	74.4	36.38	0.6553	> 7200	0.6245	2.8	2.2	68	0.6245	> 7200	–	–	–
	13	> 7200	0.7515	–	> 7200	0.7605	98.7	93.7	74.07	0.7721	> 7200	0.7361	3.5	2.9	125.74	0.7413	> 7200	–	–	–
	14	> 7200	0.8481	–	> 7200	0.9238	99.3	98.9	148.47	0.9803	> 7200	0.7941	3.5	3	514.9	0.7967	> 7200	–	–	–
	15	> 7200	1.0057	–	> 7200	0.9898	99.6	99.4	298.8	1.0254	> 7200	0.9353	7.8	7.3	642.75	0.9746	> 7200	–	–	–

Table 5.7: Computational results for data from the networking application generated with $a = 0.1$.

		Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
Matrix	$\lvert V_t\rvert$	**T**	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
	5	0.58	0.4266	0	0.08	0.4247	28.1	0	0	0.4266	0.02	0.4247	0	0	0	0.4266	0.24	0.4247	0	0
A10S7	6	0.33	0.4579	0	0.36	0.4566	49.8	0	0	0.4579	0.03	0.4566	0	0	0.02	0.4579	3.19	0.4566	0	0
	7	1.72	0.5328	0	3.46	0.5311	67.9	0	0.03	0.5328	0.13	0.5311	0	0	0.02	0.5328	720.66	0.5311	0	0
	5	0.59	0.1992	0	0.08	0.1985	34.3	0	0.01	0.1995	0	0.1985	0	0	0	0.1995	0.23	0.1985	0	0
	6	0.41	0.2903	0	0.61	0.2895	57.4	0	0.02	0.2906	0.52	0.2895	0.1	0	0	0.2906	80.13	0.2895	0	0
	7	1.42	0.3172	0	2.57	0.3165	66.5	0	0.02	0.3173	0.69	0.3165	1.5	0	0.02	0.3173	631.13	0.3165	0	0
	8	33.31	0.3987	0	10.23	0.3973	79.3	0	0.03	0.3987	5.74	0.3973	2.7	0	0.03	0.3987	> 7200	0.5132	24.1	24
	9	322.69	0.4146	0	84.46	0.4129	86.5	0	0.16	0.415	9.86	0.4129	2.6	0	0.16	0.415	> 7200	–	–	–
A10S15	10	> 7200	0.4526	36.1	3927.17	0.4505	91.8	0	0.3	0.4528	170.62	0.4505	3.1	0	0.3	0.4528	> 7200	–	–	–
	11	> 7200	0.6608	66.8	> 7200	0.657	95.9	43.2	18.44	0.6612	895.27	0.657	2.2	0	16.64	0.6612	> 7200	–	–	–
	12	> 7200	0.7641	95.5	> 7200	0.7637	97.6	69.1	33.01	0.7751	> 7200	0.7587	1.9	1.3	32.79	0.7634	> 7200	–	–	–
	13	> 7200	0.8666	–	> 7200	0.8328	98.5	91.5	57.28	0.8764	> 7200	0.7854	7	1.7	111.45	0.7891	> 7200	–	–	–
	14	> 7200	0.9359	–	> 7200	0.9286	99.2	98.7	184.38	0.9999	> 7200	0.8169	8	6.3	456.53	0.8276	> 7200	–	–	–
	15	> 7200	1.0797	–	> 7200	1.1801	99.6	99.4	359.82	1.2921	> 7200	1.0344	15.1	14.5	17.27	1.1003	> 7200	–	–	–
	5	0.36	0.315	0	0.08	0.3135	35.4	0	0	0.3152	0	0.3135	0	0	0	0.3152	0.21	0.3135	0	0
	6	0.45	0.3581	0	0.36	0.3567	57.4	0	0.02	0.3583	0.06	0.3567	0.1	0	0.01	0.3583	31.8	0.3567	0	0
	7	1.92	0.3951	0	2.92	0.3939	70.7	0	0.02	0.3951	0.86	0.3939	3.6	0	0.02	0.3951	63.87	0.3939	3.1	0
	8	50.17	0.4805	0	24.15	0.4792	81.8	0	0.03	0.4807	11.54	0.4792	3.4	0	0.05	0.4807	> 7200	0.5742	18.8	18.7
	9	642.67	0.5452	0	246.28	0.5428	88.4	0	0.16	0.5454	12.9	0.5428	3.3	0	0.16	0.5454	> 7200	–	–	–
A10S20	10	> 7200	0.6048	51.2	> 7200	0.6026	93	16.6	0.69	0.6048	206.08	0.6026	2.8	0	1	0.6048	> 7200	–	–	–
	11	> 7200	0.643	71.1	> 7200	0.6443	95.7	47.2	16	0.6532	3055.13	0.6392	2.6	0	77.49	0.6429	> 7200	–	–	–
	12	> 7200	0.6729	98.9	> 7200	0.6615	97.5	72.6	30.69	0.6706	> 7200	0.656	2.7	2	74.49	0.6596	> 7200	–	–	–
	13	> 7200	0.8031	–	> 7200	0.7913	98.7	93	68.02	0.8062	> 7200	0.7726	3.4	3	228.51	0.7821	> 7200	–	–	–
	14	> 7200	0.8843	–	> 7200	0.9435	99.3	98.9	113.71	1.008	> 7200	0.8547	5.7	5.2	281.02	0.883	> 7200	–	–	–
	15	> 7200	1.0276	–	> 7200	1.2949	99.7	99.7	196.14	1.3923	> 7200	0.9342	3.2	2.8	1590.72	0.9413	> 7200	–	–	–

Table 5.8: Computational results for data from the networking application generated with $a = 0.15$.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
A15S7	5	0.22	0.4377	0	0.09	0.4349	28	0	0	0.4377	0	0.4349	0	0	0.02	0.4377	0.22	0.4349	0	0
	6	0.27	0.4694	0	0.38	0.4676	49.8	0	0	0.4694	0.05	0.4676	0	0	0.02	0.4694	100.34	0.4676	0	0
	7	1.86	0.5465	0	3.54	0.544	68	0	0.02	0.5465	0.09	0.544	0	0	0.01	0.5465	656.82	0.544	0	0
A15S15	5	0.08	0.2042	0	0.09	0.2031	34.2	0	0	0.2046	0.01	0.2031	0	0	0	0.2046	0.23	0.2031	0	0
	6	0.44	0.2977	0	0.44	0.2964	57.4	0	0.02	0.2981	0.23	0.2964	0.2	0	0.01	0.2981	335.35	0.2964	0	0
	7	1.54	0.3252	0	3.18	0.3241	66.5	0	0.02	0.3252	0.72	0.3241	1.5	0	0.02	0.3252	145.05	0.3241	0	0
	8	32.93	0.4092	0	10.76	0.4071	79.3	0	0.03	0.4092	3.79	0.4071	2.7	0	0.05	0.4092	> 7200	0.5256	24.1	23.9
	9	311.89	0.4254	0	73.73	0.4228	86.5	0	0.16	0.426	10.37	0.4228	2.6	0	0.26	0.426	> 7200	–	–	–
	10	> 7200	0.4646	35.4	3651.69	0.4615	91.8	0	0.31	0.4646	119.43	0.4615	3.1	0	15.63	0.4646	> 7200	–	–	–
	11	> 7200	0.6781	68	> 7200	0.6729	95.9	43.5	18.21	0.6786	902.2	0.6725	2.1	0	68.92	0.6788	> 7200	–	–	–
	12	> 7200	0.7859	95.1	> 7200	0.7823	97.6	69.1	35.71	0.7962	> 7200	0.7771	1.9	1.5	74.97	0.7841	> 7200	–	–	–
	13	> 7200	0.8869	–	> 7200	0.8401	98.5	91.8	48.56	0.8768	> 7200	0.8046	7	1.6	109.11	0.8105	> 7200	–	–	–
	14	> 7200	0.9621	–	> 7200	0.9349	99.2	98.7	116.66	1.026	> 7200	0.8696	11.5	9.9	10.25	0.9138	> 7200	–	–	–
	15	> 7200	1.1408	–	> 7200	1.6528	99.7	99.7	295.4	1.799	> 7200	1.0131	11.2	10.7	946.84	1.0738	> 7200	–	–	–
A15S20	5	0.38	0.3229	0	0.09	0.3208	35.3	0	0	0.3233	0.01	0.3208	0	0	0	0.3233	0.24	0.3208	0	0
	6	0.52	0.3675	0	0.37	0.3653	57.4	0	0.02	0.3678	0.16	0.3653	0.1	0	0	0.3678	57.90	0.3653	0	0
	7	1.67	0.405	0	2.89	0.4033	70.7	0	0.02	0.405	0.75	0.4033	3.5	0	0.02	0.405	57.84	0.4033	3.1	0
	8	50.28	0.4925	0	23.76	0.4907	81.8	0	0.03	0.4928	7.96	0.4907	3.3	0	0.03	0.4928	> 7200	0.5883	18.8	18.7
	9	701.44	0.5592	0	245.78	0.5555	88.4	0	0.16	0.5593	11.75	0.5555	3.2	0	0.16	0.5593	> 7200	–	–	–
	10	> 7200	0.6203	49.9	> 7200	0.6176	93	16.3	0.67	0.6212	179.48	0.6171	2.7	0	16.89	0.6204	> 7200	–	–	–
	11	> 7200	0.6602	70.3	> 7200	0.6545	95.7	47.1	18.58	0.6596	1804.07	0.6542	2.6	0	2.25	0.6598	> 7200	–	–	–
	12	> 7200	0.6908	98.3	> 7200	0.685	97.6	73.9	35.65	0.7037	> 7200	0.6718	2.7	2	70.18	0.6771	> 7200	–	–	–
	13	> 7200	0.8125	–	> 7200	0.8049	98.7	93.4	70.78	0.8235	> 7200	0.7853	2.7	2.2	116.16	0.7917	> 7200	–	–	–
	14	> 7200	0.908	–	> 7200	0.9443	99.3	98.9	237.18	0.9860	> 7200	0.8714	5.3	4.8	304.57	0.8884	> 7200	–	–	–
	15	> 7200	1.076	–	> 7200	1.1127	99.6	99.4	276.03	1.1873	> 7200	0.9891	6.5	6.1	964.21	1.01	> 7200	–	–	–

Table 5.9: Computational results for data from the networking application generated with $a = 0.2$.

		Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
Matrix	$\lvert V_t \rvert$	**T**	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
	5	0.22	0.4488	0	0.08	0.4451	28	0	0	0.4488	0.02	0.4451	0	0	0	0.4488	0.22	0.4451	0	0
A20S7	6	0.27	0.481	0	0.53	0.4785	49.7	0	0	0.481	0.05	0.4785	0	0	0	0.481	1.72	0.4785	0	0
	7	1.75	0.5602	0	2.26	0.5568	68	0	0.02	0.5602	0.14	0.5568	0	0	0.01	0.5602	509.79	0.5568	0	0
	5	0.2	0.2091	0	0.09	0.2078	34.1	0	0	0.2098	0.01	0.2078	0	0	0	0.2098	0.29	0.2078	0	0
	6	0.34	0.3051	0	0.39	0.3034	57.4	0	0.02	0.3055	0.53	0.3034	0.2	0	0.02	0.3055	17.98	0.3034	0	0
	7	1.45	0.3332	0	1.47	0.3317	66.5	0	0.02	0.3334	0.87	0.3317	1.5	0	0.02	0.3334	82.80	0.3317	0	0
	8	33.09	0.4197	0	11.26	0.4168	79.3	0	0.03	0.4197	7.71	0.4168	2.7	0	0.03	0.4197	> 7200	0.5381	24.1	23.9
	9	347.63	0.4361	0	77.88	0.4327	86.4	0	0.16	0.4369	8.52	0.4327	2.6	0	0.16	0.4369	> 7200	–	–	–
A20S15	10	> 7200	0.4766	35.7	3806.05	0.4725	91.8	0	0.31	0.4766	158.62	0.4725	3.1	0	1	0.4766	> 7200	–	–	–
	11	> 7200	0.6956	66.9	> 7200	0.6886	95.9	43.7	14.68	0.6964	909.73	0.688	2.1	0	2.32	0.6964	> 7200	–	–	–
	12	> 7200	0.8048	95.7	> 7200	0.808	97.7	69.3	33.53	0.8308	> 7200	0.7955	1.9	1.7	75.97	0.8048	> 7200	–	–	–
	13	> 7200	0.8704	–	> 7200	0.8883	98.6	92.4	56.44	0.9130	> 7200	0.8235	7	1.7	148.98	0.8307	> 7200	–	–	–
	14	> 7200	0.9661	–	> 7200	0.9922	99.2	98.7	135.45	1.0770	> 7200	0.9416	16.3	15.4	380.08	1.0304	> 7200	–	–	–
	15	> 7200	1.1575	–	> 7200	1.2393	99.6	99.4	310.91	1.3841	> 7200	1.1320	18.7	18.2	777.36	1.2272	> 7200	–	–	–
	5	0.33	0.3309	0	0.09	0.3280	35.2	0	0	0.3315	0	0.3280	0	0	0.02	0.3315	2.45	0.3280	0	0
	6	0.28	0.3768	0	0.36	0.3739	57.4	0	0	0.3772	0.08	0.3739	0.2	0	0.02	0.3772	24.02	0.3739	0	0
	7	1.83	0.415	0	3.07	0.4127	70.7	0	0.03	0.415	0.87	0.4127	3.4	0	0.03	0.415	332.28	0.4127	3	0
	8	49.11	0.5046	0	22.62	0.5021	81.7	0	0.03	0.5049	13	0.5021	3.2	0	0.05	0.5049	> 7200	0.6025	18.8	18.7
	9	663.49	0.5731	0	267.99	0.5681	88.4	0	0.16	0.5733	11.45	0.5681	3.2	0	0.16	0.5733	> 7200	–	–	–
A20S20	10	> 7200	0.6359	51.2	> 7200	0.6316	93	16.7	0.67	0.6359	178.03	0.6316	2.7	0	9.36	0.6359	> 7200	–	–	–
	11	> 7200	0.6769	70.3	> 7200	0.6705	95.7	46.5	16.75	0.6787	944.07	0.6693	2.5	0	2.34	0.6768	> 7200	–	–	–
	12	> 7200	0.7094	98.8	> 7200	0.703	97.6	74.3	34.54	0.7294	> 7200	0.6875	2.7	2	68.39	0.6947	> 7200	–	–	–
	13	> 7200	0.8393	–	> 7200	0.8405	98.7	84.5	64.21	0.8805	> 7200	0.8034	2.7	2.2	124.25	0.812	> 7200	–	–	–
	14	> 7200	0.9205	–	> 7200	1.0017	99.3	98.9	125.91	1.036	> 7200	0.879	3.8	3.4	306.21	0.8989	> 7200	–	–	–
	15	> 7200	1.1288	–	> 7200	1.3825	99.7	99.7	230.49	1.4636	> 7200	0.9817	3.7	3.3	873.09	0.9988	> 7200	–	–	–

Table 5.10: Computational results for data from the networking application generated with $a = 1$.

		Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
Matrix	$\lvert V_t \rvert$	**T**	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
	5	0.22	0.6269	0	0.11	0.6083	27.6	0	0	0.6269	0	0.6083	0	0	0	0.6269	0.23	0.6083	0	0
A100S7	6	0.37	0.6662	0	0.36	0.6535	49.5	0	0	0.6662	0.03	0.6535	0	0	0.01	0.6662	6.51	0.6535	0	0
	7	1.64	0.7791	0	2.39	0.7625	68.1	0	0.02	0.7791	0.16	0.7625	0.2	0	0.02	0.7791	598.04	0.7625	0	0
	5	0.08	0.2884	0	0.08	0.2815	33.2	0	0	0.2916	0.01	0.2815	0	0	0	0.2916	0.23	0.2815	0	0
	6	0.41	0.4231	0	0.41	0.4147	57.6	0	0.01	0.4255	0.42	0.4147	0.8	0	0.02	0.4255	14.48	0.4147	0	0
	7	1.54	0.4609	0	0.97	0.4536	66.6	0	0.02	0.4609	0.61	0.4536	1.3	0	0.02	0.4609	225.33	0.4536	0.1	0
	8	34.24	0.5873	0	11.36	0.573	79.4	0	0.03	0.5873	8.56	0.573	2.6	0	0.05	0.5873	> 7200	0.737	23.8	23.5
	9	325.43	0.6076	0	68.69	0.5906	86.4	0	0.16	0.6116	9.63	0.5906	2.8	0	0.16	0.6116	> 7200	–	–	–
A100S15	10	> 7200	0.6685	38.2	3826.94	0.6478	91.8	0	0.31	0.6685	142.88	0.6478	30	0	16.5	0.6685	> 7200	–	–	–
	11	> 7200	0.9938	65.4	> 7200	0.9373	95.9	42.2	20.59	0.9783	700.94	0.9359	2.1	0	50.56	0.9782	> 7200	–	–	–
	12	> 7200	1.1447	97.5	> 7200	1.0958	97.6	68.9	36.04	1.1666	> 7200	1.0898	2.4	20	76.41	1.1363	> 7200	–	–	–
	13	> 7200	1.2814	–	> 7200	1.1623	98.5	91.6	54.23	1.2403	> 7200	1.1357	7.4	2.3	5.12	1.186	> 7200	–	–	–
	14	> 7200	1.3932	–	> 7200	1.4504	99.2	98.8	109.26	1.6959	> 7200	1.2284	12.5	11.6	10.26	1.3554	> 7200	–	–	–
	15	> 7200	1.6011	–	> 7200	1.7994	99.6	99.4	326.06	2.0011	> 7200	1.4737	14.5	13.9	1378.98	1.6641	> 7200	–	–	–
	5	0.37	0.4583	0	0.08	0.4441	34.3	0	0.02	0.4611	0.02	0.4441	0	0	0	0.4611	0.22	0.4441	0	0
	6	0.5	0.5263	0	0.37	0.5117	57.5	0	0.02	0.5281	0.28	0.5117	0.7	0	0	0.5281	81.25	0.5117	0	0
	7	1.89	0.5742	0	3.5	0.563	70.7	0	0.02	0.5742	0.94	0.563	2.6	0	0.02	0.5742	492.87	0.563	2.3	0
	8	46.96	0.6973	0	21.92	0.6849	81.7	0	0.05	0.699	8.53	0.6849	2.6	0	0.05	0.699	> 7200	0.829	19.1	18.8
	9	679.91	0.796	0	231.26	0.7713	88.2	0	0.16	0.7972	12.36	0.7713	3.2	0	0.16	0.7972	> 7200	–	–	–
A100S20	10	> 7200	0.8852	51.3	> 7200	0.8635	92.9	14.8	1.75	0.8853	137.65	0.8635	2.2	0	9.31	0.8853	> 7200	–	–	–
	11	> 7200	0.9477	73.3	> 7200	0.9152	95.7	46.2	18.1	0.9593	1042.36	0.9104	2.4	0	66.85	0.9477	> 7200	–	–	–
	12	> 7200	1.0137	99.2	> 7200	0.9618	97.6	73.1	38.7	1.0323	> 7200	0.9399	2.6	20	5.02	0.9756	> 7200	–	–	–
	13	> 7200	1.1752	–	> 7200	1.1787	98.7	890	64.07	1.2614	> 7200	1.0919	2.6	2.2	127.52	1.1311	> 7200	–	–	–
	14	> 7200	1.3391	–	> 7200	1.3367	99.3	98.9	159.28	1.422	> 7200	1.216	4.8	4.4	497.98	1.2801	> 7200	–	–	–
	15	> 7200	1.5517	–	> 7200	1.5042	99.6	99.4	272.17	1.6982	> 7200	1.3386	4.9	4.7	771.36	1.4284	> 7200	–	–	–

Table 5.11: Computational results for data from the networking application generated with random delays.

Matrix	$\|V_t\|$	Path-weight formulation			Path-edges formulation						Path-edges$^+$ formulation						PL4			
		T	**W**	**GAP$_{LB}$**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**	**T$_w$**	**W**	**T**	**DZ**	**GAP**	**GAP$_{LB}$**
	5	0.27	0.3961	0	0.09	0.3929	28.3	0	0	0.3961	0	0.3929	0	0	0.02	0.3961	0.58	0.3929	0	0
SS7	6	0.45	0.4237	0	0.36	0.4202	49.4	0	0.02	0.4237	0.05	0.4202	0	0	0.02	0.4237	1.25	0.4202	0	0
	7	2.06	0.4984	0	3.48	0.4913	67.6	0	0.02	0.4984	0.09	0.4913	0	0	0.02	0.4984	458.92	0.4913	0	0
	5	0.09	0.2081	0	0.11	0.2075	34.6	0	0.02	0.2081	0.02	0.2075	0	0	0	0.2081	0.23	0.2075	0	0
	6	0.34	0.2944	0	0.55	0.2936	57.1	0	0	0.2945	0.14	0.2936	0	0	0.01	0.2945	102.29	0.2936	0	0
	7	1.51	0.3205	0	1.5	0.3193	66.9	0	0.02	0.3205	0.37	0.3193	1.4	0	0.02	0.3205	70.86	0.3193	0	0
	8	33.6	0.3997	0	11.47	0.3982	79.7	0	0.05	0.3997	4.51	0.3982	2.4	0	0.03	0.3997	> 7200	0.5099	23.3	23.1
	9	336.65	0.4154	0	89.7	0.4135	86.7	0	0.16	0.4154	8.5	0.4135	2.4	0	0.17	0.4154	> 7200	–	–	–
SS15	10	> 7200	0.4517	38.9	4519.7	0.4494	92	0	0.3	0.4517	103.65	0.4494	2.9	0	1.03	0.4517	> 7200	–	–	–
	11	> 7200	0.6528	69.6	> 7200	0.6494	96	40.6	17.39	0.6528	762.98	0.6492	2.1	0	2.28	0.6518	> 7200	–	–	–
	12	> 7200	0.7582	97	> 7200	0.7551	97.7	69.5	34.43	0.7671	> 7200	0.745	1.8	1.5	38.34	0.7478	> 7200	–	–	–
	5	0.31	0.3024	0	0.08	0.3019	35.3	0	0.02	0.3026	0.01	0.3019	0	0	0	0.3026	0.23	0.3019	0	0
	6	0.44	0.3426	0	0.39	0.342	57.3	0	0	0.3427	0.19	0.342	0.1	0	0.02	0.3427	104.18	0.342	0	0
	7	1.89	0.3786	0	1.86	0.3779	70.6	0	0.02	0.3789	0.55	0.3779	3.6	0	0.02	0.3789	368.51	0.3779	3.1	0
	8	51.28	0.4602	0	24.1	0.4592	81.8	0	0.03	0.461	7.67	0.4592	3.5	0	0.03	0.461	> 7200	0.5483	18.6	18.5
	9	648.77	0.5217	0	151.99	0.5202	88.4	0	0.16	0.522	12.21	0.5202	3.2	0	0.16	0.522	> 7200	–	–	–
SS20	10	> 7200	0.5781	50.8	> 7200	0.5764	93	16.6	0.7	0.5784	280.8	0.5764	2.8	0	1.01	0.5784	> 7200	–	–	–
	11	> 7200	0.614	71.1	> 7200	0.6116	95.7	47	17.86	0.6136	1108.13	0.6114	2.7	0	15.88	0.6136	> 7200	–	–	–
	12	> 7200	0.6392	98.2	> 7200	0.6319	97.5	73.6	35.76	0.6388	> 7200	0.6264	2.8	2.1	68.23	0.6287	> 7200	–	–	–

The optimum solution within the time limit imposed is obtained in the following cases:

- by the Path-weight formulation for all instances with $n < 10$ terminal nodes;
- by the Path-edges formulation for all instances with $n < 10$ terminal nodes and for instances obtained from matrices Primate and S15 with $n = 10$;
- by the Path-edges$^+$ formulation for all instances with $n < 12$ terminal nodes;
- by the PL4 formulation for all instances with $n < 8$ terminal nodes.

To better check the improvements achieved we display in Table 5.12 the number of instances, among the total of 294 instances, for which the optimum solution was obtained, within the time limit imposed, when the model indicated is used.

Path-weight	Path-edges	Path-edges$^+$	PL4
153	164	207	99
52%	56%	70%	34%

Table 5.12: Number of instances solved within the time limit imposed.

The Path-edges$^+$ formulation solves the instances substantially faster than the other formulations. In Figure 5.1, we compare the computational times in the form of a profile graph that displays the number of solved instances in a given time by each formulation.

Comparing, in Tables 5.1 - 5.11, the columns labeled with W of the Path-weight formulation and of the Path-edges formulation we notice that for some instances the values obtained are different. This is due to the fact that the obtained trees are different. The tree we obtain when minimizing $\sum_{k \in V_t} \sum_{\substack{\ell \in V_t \\ \ell > k}} d_{k\ell} \sum_{i=2}^{n-1} 2^{-i} \cdot p^i_{k\ell}$ is such that, when assigning its edges weights the equality $\sum_{e \in P_{ij}} w_e = d_{ij}$ does not hold. In that case we obtain a tree that

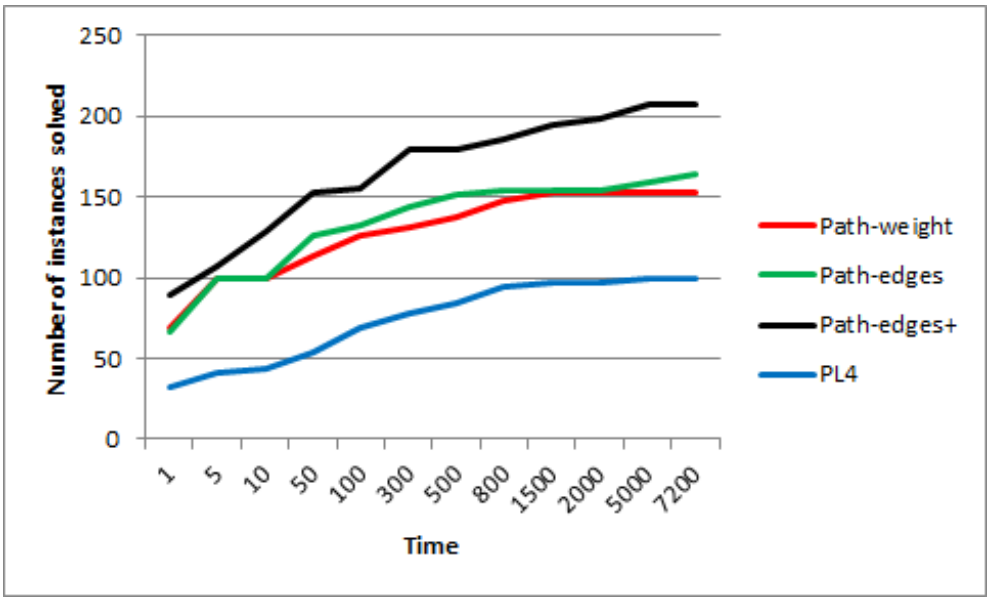


Figure 5.1: Performance profile.

minimizes $\sum_{k\in V_t}\sum_{\substack{\ell\in V_t\\ \ell>k}} d_{k\ell}\sum_{i=2}^{n-1} 2^{-i}\cdot p^i_{k\ell}$ but does not minimizes $\sum_{i\in V_a}\sum_{\substack{j\in V\\ j>i}} w_{ij}$ (see column labeled DZ), as in this case it holds $\sum_{k\in V_t}\sum_{\substack{\ell\in V_t\\ \ell>k}} d_{k\ell}\sum_{i=2}^{n-1} 2^{-i}\cdot p^i_{k\ell} < \sum_{i\in V_a}\sum_{\substack{j\in V\\ j>i}} w_{ij}$.

When the optimum solution can not be found within the runtime limit imposed, the Path-weight formulation obtains, on average, fourteen feasible solutions, the Path-edges formulation obtains, on average, ten feasible solutions, the Path-edges$^+$ formulation obtains, on average, six feasible solutions and the PL4 formulation is unable to obtain feasible solutions within the runtime limit imposed for $n > 8$ and for $n = 8$ obtains, on average, one feasible solution. The difference between the number of feasible solutions of the Path-weight formulation and of the Path-edges formulation is due to the fact that the only integer (binary) variables of the Path-weight formulation are those which identify the tree, whereas the Path-edges formulation only has integer (binary) variables.

For instances with the same number of terminal nodes, Tables 5.13, 5.14 and 5.15 display the average time, the average GAP and the average GAP_{LB}, respectively, and their corresponding standard deviation values. Table 5.13 presents the values for instances

with $n \leq 9$, since for $n > 9$ only the Path-edges$^+$ obtains the optimum solution within the runtime limit imposed. Table 5.15 displays the GAP_{LB} values for the formulations Path-weight, Path-edges and Path-edge$^+$ and for instances with $n \geq 10$, once for $n < 10$ the GAP_{LB} is equal to zero (the optimum solution is achieved). This table does not present the GAP_{LB} values of PL4, since the formulation PL4 does not obtain feasible solutions for $n \geq 10$.

Table 5.13: Average and standard deviation (SD) values for the computational time of the Path-weight, Path-edges, Path-edges$^+$ and PL4 formulations.

n		Path-weight	Path-edges	Path-edges$^+$	PL4
5	average	0.26	0.09	0.01	0.35
	SD	0.12	0.01	0.01	0.44
6	average	0.41	0.43	0.18	60.02
	SD	0.11	0.09	0.22	68.49
7	average	1.88	2.91	0.92	550.51
	SD	0.28	0.79	0.72	770.13
8	average	45.11	20.62	8.49	–
	SD	17.62	8	5.28	–
9	average	581.13	246.4	11.06	–
	SD	281.78	157.57	2.07	–

To evaluate the impact of the valid equalities (5.43) and (5.44), we implemented the Path-edges formulation together with only valid equalities, which we designated Path-edges^{+2} formulation. The computational results of the Path-edges^{+2} formulation are summarized in Tables 5.16 - 5.19.

Table 5.14: Average and standard deviation (SD) values for the GAP of the Path-edges, Path-edges$^+$ and PL4 formulations.

n		Path-edges	Path-edges$^+$	PL4
5	average	31.8	0	0
	SD	2.57	0	0.02
6	average	56.81	0.39	0.05
	SD	3.68	0.55	0.1
7	average	70.27	1.34	0.64
	SD	2.52	1.12	1.15
8	average	81.32	1.8	18.43
	SD	1.34	1.13	5.51
9	average	88.21	1.36	–
	SD	2.3	0.99	–
10	average	92.93	2.3	–
	SD	0.98	0.68	–
11	average	96.13	2.27	–
	SD	0.42	0.59	–
12	average	97.78	2.4	–
	SD	0.25	0.89	–
13	average	98.77	4.01	–
	SD	0.18	1.98	–
14	average	99.34	6.33	–
	SD	0.1	3.55	–
15	average	99.67	8.8	–
	SD	0.04	4.11	–

Table 5.15: Average and standard deviation (SD) values for the GAP_{LB} of the Path-weight, Path-edges and Path-edges$^+$ formulations.

n		Path-weight	Path-edges	Path-edges$^+$
10	average	42.21	14.03	0
	SD	18.73	12.82	0
11	average	74.41	48.09	0
	SD	6.64	5.15	0
12	average	97.4	73.8	1.76
	SD	1.4	4.74	0.47
13	average	–	92.73	2.51
	SD	–	2.48	0.87
14	average	–	98.93	5.84
	SD	–	0.16	3.22
15	average	–	99.54	8.53
	SD	–	0.16	4.02

Table 5.16: Computational results of the Path-edges^{+2} formulation for data from the phylogenetic application.

Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W
M391	5	0.01	0.0473	0	0	0.0487	A10M391	5	0.02	0.0498	0	0	0.0515	A15M391	5	0	0.051	0	0	0.0529
	6	0.06	0.057	0.4	0	0.0592		6	0.08	0.0599	0.4	0	0.0622		6	0.09	0.0613	0.4	0	0.0637
	7	1.45	0.0607	0.8	0	0.0634		7	1.73	0.0639	0.8	0	0.0668		7	1.97	0.0654	0.8	0	0.0685
	8	6.94	0.0672	0.8	0	0.0698		8	7.36	0.0707	0.7	0	0.0734		8	5.76	0.0724	0.7	0	0.0752
	9	22.32	0.0923	0.6	0	0.0952		9	25.54	0.0968	0.6	0	0.0998		9	18.95	0.099	0.6	0	0.1021
	10	451.93	0.1027	1	0	0.1068		10	151.55	0.1082	1.1	0	0.1124		10	348.96	0.111	1.1	0	0.1153
	11	1969.07	0.1276	1.2	0	0.1338		11	4358.52	0.1345	1.1	0	0.141		11	3941.46	0.138	1.1	0	0.1433
	12	>7200	0.1322	1.2	0.9	0.1378		12	>7200	0.1393	1.3	1	0.1454		12	>7200	0.1428	1.4	1	0.1492
	13	>7200	0.145	2.1	2	0.1536		13	>7200	0.1524	2.1	2	0.1613		13	>7200	0.1562	2.1	2.1	0.1647
	14	>7200	0.1538	6	5.9	0.1639		14	>7200	0.1549	1.9	1.8	0.1623		14	>7200	0.1616	3.8	3.7	0.1715
	15	>7200	0.1591	5.7	5.6	0.1697		15	>7200	0.1679	6	5.9	0.1783		15	>7200	0.1753	7.8	7.7	0.1856
Primate	5	0.02	0.0823	0	0	0.0843	A10Pri	5	0.02	0.0864	0	0	0.0893	A15Pri	5	0	0.0885	0	0	0.0917
	6	0.06	0.1023	0	0	0.1059		6	0.05	0.1076	0	0	0.1115		6	0.03	0.1103	0	0	0.1143
	7	0.58	0.1232	0.5	0	0.1272		7	0.86	0.1296	0.7	0	0.1343		7	1.64	0.1328	0.8	0	0.1379
	8	1.94	0.1230	0.8	0	0.128		8	1.7	0.1294	0.7	0	0.1351		8	1.98	0.1327	0.7	0	0.1386
	9	14.16	0.1246	3.1	0	0.1296		9	18.03	0.1311	3.2	0	0.1364		9	16.91	0.1343	3.2	0	0.1397
	10	212.92	0.1220	2.6	0	0.1298		10	189.71	0.1283	2.5	0	0.1361		10	209.59	0.1315	2.5	0	0.1393
	11	1951.88	0.1625	2.6	0	0.17		11	1401.69	0.1709	2.7	0	0.1794		11	1990.42	0.1751	2.7	0	0.1841
	12	>7200	0.1959	2.5	1.6	0.2058		12	>7200	0.2062	2.5	1.7	0.2162		12	>7200	0.2113	2.5	1.6	0.2214
M887	5	0.01	0.0972	0	0	0.1033	A10M887	5	0	0.1023	0	0	0.109	A15M887	5	0.01	0.1049	0	0	0.1118
	6	0.48	0.1098	2	0	0.1157		6	0.64	0.1155	1.8	0	0.1215		6	0.72	0.1183	1.7	0	0.1245
	7	1.22	0.1292	1.3	0	0.1364		7	1.25	0.1357	1.4	0	0.1432		7	1.81	0.1389	1.4	0	0.1466
	8	6.65	0.1619	1.6	0	0.1763		8	4.99	0.1702	1.6	0	0.1858		8	5.83	0.1744	1.6	0	0.1905
	9	24.04	0.1685	1.7	0	0.186		9	20.33	0.1769	1.6	0	0.1972		9	20.64	0.1811	1.6	0	0.202
	10	200.27	0.1859	2.1	0	0.2059		10	282.11	0.1955	2.1	0	0.2156		10	173.6	0.2003	2.1	0	0.2205
	11	2170.18	0.1938	2.6	0	0.2135		11	2502.73	0.2038	2.6	0	0.2243		11	1914.09	0.2088	2.7	0	0.2297
	12	>7200	0.1975	3.1	2.9	0.2191		12	>7200	0.2069	2.8	2.6	0.2298		12	>7200	0.2116	2.6	2.4	0.2345
	13	>7200	0.2166	4.2	4.1	0.2387		13	>7200	0.2257	3.4	3.3	0.2498		13	>7200	0.2322	3.9	3.8	0.2523
	14	>7200	0.2245	5.7	5.6	0.2527		14	>7200	0.2422	8.1	8.1	0.2693		14	>7200	0.2471	7.8	7.7	0.2723
	15	>7200	0.2502	8.8	8.7	0.2853		15	>7200	0.2653	9.4	9.3	0.2873		15	>7200	0.281	12.3	12.2	0.3148

Table 5.17: Computational results of the Path-edges^{+2} formulation for data from the phylogenetic application (continuation).

Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W
A20M391	5	0	0.0522	0	0	0.0543	A100M391	5	0	0.0721	0	0	0.0748
	6	0.5	0.0627	0.5	0	0.0652		6	0.2	0.0847	0.4	0	0.0882
	7	1.98	0.067	0.9	0	0.0702		7	2.04	0.092	1.2	0	0.0968
	8	7.75	0.0741	0.8	0	0.077		8	4.65	0.1007	0.7	0	0.1044
	9	18.25	0.1013	0.6	0	0.1044		9	28.05	0.1373	1.1	0	0.143
	10	337.57	0.1138	1.1	0	0.1181		10	446.89	0.1579	1.4	0	0.1631
	11	3458.14	0.1414	1.1	0	0.1467		11	4300.85	0.1958	1.7	0	0.2052
	12	> 7200	0.1464	1.6	1.1	0.1544		12	> 7200	0.2004	1.9	1.6	0.2097
	13	> 7200	0.1606	2.6	2.5	0.1693		13	> 7200	0.2182	2	1.9	0.2336
	14	> 7200	0.1642	3.1	2.9	0.1733		14	> 7200	0.2291	5.4	5.3	0.2508
	15	> 7200	0.1743	5	4.9	0.1849		15	> 7200	0.247	8.1	8.1	0.2654
A20Pri	5	0	0.0906	0	0	0.0942	A100Pri	5	0.01	0.1236	0	0	0.1338
	6	0.02	0.113	0	0	0.1171		6	0.08	0.1558	0.7	0	0.1618
	7	1.54	0.136	0.9	0	0.1414		7	2.32	0.1869	3	0	0.1966
	8	2.32	0.1359	0.7	0	0.1422		8	1.98	0.1877	1	0	0.1991
	9	19.19	0.1375	3.3	0	0.1431		9	23.43	0.1889	3.6	0	0.2035
	10	247.37	0.1346	2.5	0	0.1427		10	185.83	0.1848	2.7	0	0.1966
	11	1289.95	0.1793	2.7	0	0.1889		11	1263.93	0.246	3.4	0	0.2655
	12	> 7200	0.2164	2.5	1.7	0.2265		12	> 7200	0.2983	2.9	2.2	0.3166
A20M887	5	0.01	0.1075	0	0	0.1147	A100M887	5	0.01	0.1487	0	0	0.1601
	6	0.5	0.1211	1.7	0	0.1274		6	0.42	0.1664	0.9	0	0.1756
	7	1.4	0.1421	1.4	0	0.15		7	1.5	0.1934	1.7	0	0.2065
	8	6.27	0.1785	1.6	0	0.1953		8	6.55	0.2451	1.8	0	0.2714
	9	21.67	0.1853	1.5	0	0.2067		9	17.91	0.2525	1.4	0	0.2833
	10	287.68	0.2051	2	0	0.2253		10	243.16	0.2822	2.1	0	0.3064
	11	2979.29	0.2138	2.7	0	0.2352		11	> 7200	0.2935	3.5	1.6	0.3177
	12	> 7200	0.2138	2.6	2.4	0.2352		12	> 7200	0.2953	2.8	2.7	0.3265
	13	> 7200	0.2369	3.6	3.5	0.2621		13	> 7200	0.3241	4.1	4	0.3497
	14	> 7200	0.2492	6.4	6.3	0.2778		14	> 7200	0.3407	6.7	6.7	0.3842
	15	> 7200	0.274	7.8	7.8	0.3035		15	> 7200	0.3867	10.2	10.1	0.436

Table 5.18: Computational results of the Path-edges^{+2} formulation for data from the networking application.

Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W
S7	5	0.02	0.4043	0	0	0.4043	A10S7	5	0	0.4247	0	0	0.4266	A15S7	5	0	0.4349	0	0	0.4377
	6	0.05	0.4347	0	0	0.4348		6	0.03	0.4566	0	0	0.4579		6	0.05	0.4676	0	0	0.4694
	7	0.06	0.5054	0	0	0.5055		7	0.25	0.5311	0	0	0.5328		7	0.17	0.544	0	0	0.5465
S15	5	0.02	0.1893	0	0	0.1893	A10S15	5	0	0.1985	0	0	0.1995	A15S15	5	0.02	0.2031	0	0	0.2046
	6	0.03	0.2756	0	0	0.2756		6	0.59	0.2895	0.1	0	0.2906		6	0.5	0.2964	0.2	0	0.2981
	7	1.42	0.3013	3.4	0	0.3013		7	1.19	0.3165	3.4	0	0.3173		7	1.22	0.3241	3.3	0	0.3254
	8	5.02	0.3778	2.7	0	0.3778		8	6.8	0.3973	2.7	0	0.3987		8	7.41	0.4071	2.7	0	0.4102
	9	11.98	0.3932	2.6	0	0.3932		9	19.23	0.4129	2.6	0	0.415		9	15.73	0.4228	2.7	0	0.426
	10	136.94	0.4286	3.2	0	0.4286		10	141.93	0.4505	3.1	0	0.4528		10	213.14	0.4615	3.1	0	0.4646
	11	1112.59	0.626	2.3	0	0.626		11	1303.74	0.657	2.2	0	0.6612		11	1019.12	0.6725	2.2	0	0.6788
	12	>7200	0.722	2	1.7	0.722		12	>7200	0.7587	1.9	1.7	0.7634		12	>7200	0.7771	1.9	1.4	0.7841
	13	>7200	0.7474	7.7	1.7	0.7474		13	>7200	0.7854	7.6	1.7	0.7891		13	>7200	0.8099	8.2	2.3	0.8215
	14	>7200	0.8033	10.9	9.4	0.8236		14	>7200	0.86	12.6	11.1	0.8799		14	>7200	0.8918	13.7	12.2	0.9243
	15	>7200	0.9551	12.6	11.9	1.0110		15	>7200	0.9347	6.1	5.4	0.9514		15	>7200	0.9342	3.8	3	0.9416
S20	5	0	0.299	0	0	0.299	A10S20	5	0.02	0.3135	0	0	0.3152	A15S20	5	0.02	0.3208	0	0	0.3233
	6	0.06	0.3395	0	0	0.3395		6	0.27	0.3567	0.1	0	0.3583		6	0.59	0.3653	0.2	0	0.3678
	7	0.62	0.3751	4.1	0	0.3751		7	1.61	0.3939	3.9	0	0.3951		7	1.67	0.4033	3.8	0	0.405
	8	6.77	0.4564	3.9	0	0.4564		8	16.02	0.4792	3.7	0	0.4807		8	12.4	0.4907	3.7	0	0.4928
	9	37.97	0.5174	3.4	0	0.5174		9	22.96	0.5428	3.4	0	0.5454		9	32.46	0.5555	3.3	0	0.5593
	10	313.72	0.5736	3.1	0	0.5736		10	359.05	0.6026	3	0	0.6048		10	333.06	0.6171	2.9	0	0.6204
	11	2078.38	0.609	2.9	0	0.609		11	1201.34	0.6392	2.8	0	0.6429		11	1685.49	0.6542	2.8	0	0.6598
	12	>7200	0.6245	2.9	2.2	0.6245		12	>7200	0.656	2.9	2.1	0.6596		12	>7200	0.6719	2.9	2	0.6776
	13	>7200	0.731	2.9	2.2	0.731		13	>7200	0.7673	2.9	2.2	0.7715		13	>7200	0.7853	2.9	2.1	0.7917
	14	>7200	0.7915	3.3	2.7	0.7967		14	>7200	0.8463	4.9	4.4	0.8642		14	>7200	0.8534	3.4	2.9	0.8661
	15	>7200	0.9128	5.5	5.1	0.9243		15	>7200	0.9426	4.2	3.7	0.9635		15	>7200	1.0149	9	8.5	1.0422

Table 5.19: Computational results of the Path-edges^{+2} formulation for data from the networking application (continuation).

Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W	Matrix	n	T	DZ	GAP	GAP_{LB}	W
A20S7	5	0	0.4451	0	0	0.4488	A100S7	5	0	0.6083	0	0	0.6269	SS7	5	0.02	0.3929	0	0	0.3961
	6	0.01	0.4785	0	0	0.481		6	0.05	0.6535	0	0	0.6662		6	0.05	0.4202	0	0	0.4237
	7	0.19	0.5568	0	0	0.5602		7	0.23	0.7625	0.2	0	0.7791		7	0.17	0.4913	0	0	0.4984
A20S15	5	0.01	0.2078	0	0	0.2098	A100S15	5	0	0.2815	0	0	0.2916	SS15	5	0.02	0.2075	0	0	0.2081
	6	0.52	0.3034	0.3	0	0.3055		6	0.48	0.4147	1	0	0.4255		6	0.19	0.2936	0	0	0.2945
	7	1.12	0.3317	3.3	0	0.3334		7	1.03	0.4536	3	0	0.4609		7	0.55	0.3193	3.1	0	0.3205
	8	5.74	0.4168	2.7	0	0.4197		8	11.47	0.573	2.7	0	0.5873		8	7.11	0.3982	2.5	0	0.3997
	9	14.26	0.4327	2.7	0	0.4369		9	14.68	0.5906	2.8	0	0.6116		9	13.17	0.4135	2.4	0	0.4154
	10	136.7	0.4725	3.1	0	0.4766		10	236.7	0.6478	3	0	0.6685		10	132.23	0.4494	2.9	0	0.4517
	11	890.68	0.688	2.2	0	0.6964		11	963.66	0.9359	2.1	0	0.9782		11	1296.24	0.6492	2.1	0	0.6518
	12	>7200	0.7955	1.9	1.8	0.8048		12	>7200	1.0898	2.3	2.1	1.1363		12	>7200	0.745	1.8	1.5	0.7478
	13	>7200	0.8239	7.5	1.7	0.8321		13	>7200	1.1619	10	5.6	1.2325		–	–	–	–	–	–
	14	>7200	0.8727	9.7	8.8	0.9065		14	>7200	1.2732	15.7	14.7	1.4027		–	–	–	–	–	–
	15	>7200	1.0848	15.2	14.5	1.1581		15	>7200	1.4408	12.5	11.9	1.5967		–	–	–	–	–	–
A20S20	5	0.01	0.328	0	0	0.3315	A100S20	5	0	0.4441	0	0	0.4611	SS20	5	0.02	0.3019	0	0	0.3026
	6	0.48	0.3739	0.3	0	0.3772		6	0.47	0.5117	0.9	0	0.5281		6	0.53	0.3420	0.1	0	0.3434
	7	1.42	0.4127	3.7	0	0.415		7	1.67	0.563	2.9	0	0.5742		7	1.01	0.3779	3.9	0	0.3789
	8	13.09	0.5021	3.6	0	0.5049		8	7.25	0.6849	2.9	0	0.699		8	11.53	0.4592	3.8	0	0.461
	9	22.36	0.5681	3.3	0	0.5733		9	23.2	0.7713	3.2	0	0.7972		9	24.7	0.5202	3.3	0	0.522
	10	267.77	0.6316	2.9	0	0.6359		10	279.89	0.8635	2.4	0	0.8853		10	435.10	0.5764	3	0	0.5784
	11	965.1	0.6693	2.7	0	0.6768		11	1001.66	0.9104	2.6	0	0.9477		11	1014.05	0.6114	2.8	0	0.6136
	12	>7200	0.6875	2.8	2.1	0.6947		12	>7200	0.9409	2.9	2.1	0.9791		12	>7200	0.6264	2.9	2.1	0.6287
	13	>7200	0.8034	2.9	2.2	0.812		13	>7200	1.0939	3	2.3	1.1378		–	–	–	–	–	–
	14	>7200	0.8663	2.6	2	0.8779		14	>7200	1.2091	4.3	3.9	1.2561		–	–	–	–	–	–
	15	>7200	1.0072	6.3	5.7	1.0253		15	>7200	1.4068	9.7	9.3	1.5288		–	–	–	–	–	–

As for the Path-edges^{+}, the optimum solution within the time limit imposed is obtained by the Path-edges^{+2} formulation for all instances with $n < 12$ terminal nodes except for the matrix A100M887 for which the formulation only obtains the optimum solution for instances with $n < 11$ terminal nodes. In Table 5.20 we present the average time and the corresponding standard deviation value for the Path-edges, Path-edges^{+2} and Path-edges^{+} formulations. We present the average values of the Path-edges and Path-edges^{+} to facilitate the comparison.

Table 5.20: Average and standard deviation (SD) values for the computational time of the Path-edges, Path-edges^{+2}, Path-edges^{+} and PL4 formulations.

n		Path-edges	Path-edges^{+2}	Path-edges^{+}
5	average	0.09	0.01	0.01
	SD	0.01	0.01	0.01
6	average	0.43	0.27	0.18
	SD	0.09	0.24	0.22
7	average	2.91	1.18	0.92
	SD	0.79	0.63	0.72
8	average	20.62	6.79	8.49
	SD	8	3.58	5.28
9	average	246.4	20.82	11.06
	SD	157.57	5.84	2.07
10	average	–	257.61	184
	SD	–	95.62	93.44
11	average	–	2119.42	1549.73
	SD	–	1452.84	920.16

As we can see, the Path-edges^{+2} formulation is substantially faster, on average, than the Path-edges formulation but not as fast, on average, as the Path-edges^{+} formulation.

Tables 5.21 and 5.22 display the average GAP and the average GAP_{LB} of formulations

Path-edges, Path-edges^{+2} and Path-edges^{+} and the corresponding standard deviation values. We only present the GAP_{LB} for instances with $n > 10$, since for instances with $n \leq 10$ the GAP_{LB} of formulations Path-edges^{+2} and Path-edges^{+} are zero.

Table 5.21: Average and standard deviation (SD) values for the GAP of the Path-edges, Path-edges^{+2} and Path-edges^{+}.

n		Path-edges	Path-edges^{+2}	Path-edges^{+}
5	average	31.8	0	0
	SD	2.57	0.01	0
6	average	56.81	0.43	0.39
	SD	3.68	0.59	0.55
7	average	70.27	1.8	1.34
	SD	2.52	1.43	1.12
8	average	81.32	1.98	1.8
	SD	1.34	1.16	1.13
9	average	88.21	2.35	1.36
	SD	2.3	1.03	0.99
10	average	92.93	2.4	2.3
	SD	0.98	0.71	0.68
11	average	96.13	2.37	2.27
	SD	0.42	0.64	0.59
12	average	97.78	2.35	2.4
	SD	0.25	0.56	0.89
13	average	98.77	4.27	4.01
	SD	0.18	2.44	1.98
14	average	99.34	6.81	6.33
	SD	0.1	3.91	3.55
15	average	99.67	8.3	8.8
	SD	0.04	3.16	4.11

Table 5.22: Average and standard deviation (SD) values for the GAP_{LB} of the Path-edges, Path-edges^{+2} and Path-edges^{+} formulations.

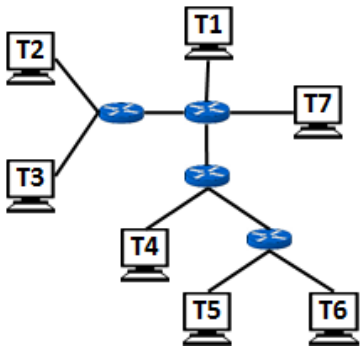
n		Path-edges	Path-edges^{+2}	Path-edges^{+}
11	average	48.09	0.06	0
	SD	5.15	0.3	0
12	average	73.8	1.87	1.76
	SD	4.74	0.53	0.47
13	average	92.73	2.66	2.51
	SD	2.48	1.03	0.87
14	average	98.93	6.3	5.84
	SD	0.16	3.56	3.22
15	average	99.54	7.96	8.53
	SD	0.16	3.11	4.02

As we can see the average GAP and the average GAP_{LB} of formulation Path-edges^{+2} are considerably lower than the correspondent values of the Path-edges formulation and are, on average, slightly higher than the values of Path-edges^{+} formulation, except for instances with $n = 15$ and the average GAP also for instances with $n = 12$. We may conclude that by including the equalities (5.43) and (5.44) the Path-edges formulation considerably improves and that the inclusion of the inequalities (5.45), (5.46) and (5.47) only slightly improves the formulation.

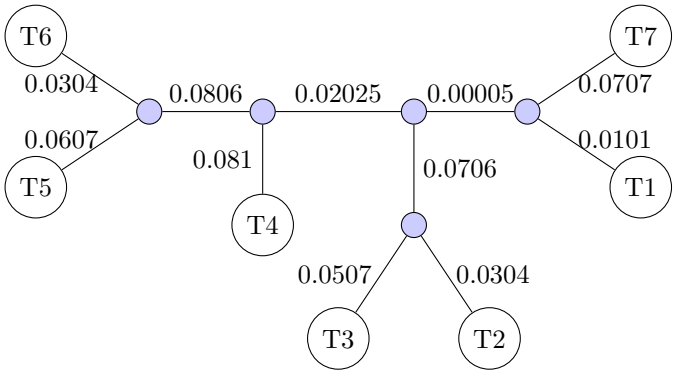
The matrices of the networking application were obtained using simulations and therefore the routing tree is known. We compared the trees obtained by the different formulations with the original trees used to run the simulations and presented in Chapter 3. To illustrate which kind of trees are obtained by using the different formulation, we present the resulting trees of matrix S7 with $n = 7$ and SS15 with $n = 6$.

Figure 5.2 displays the original logical routing tree used to run the simulation to obtain the values of matrix S7 with $n = 7$, which was already presented in Figure 3.10. Figure 5.2

also displays the logical routing tree obtained by running the five formulations, which in this case was the same for the five formulations.



(a) The original logical routing tree.



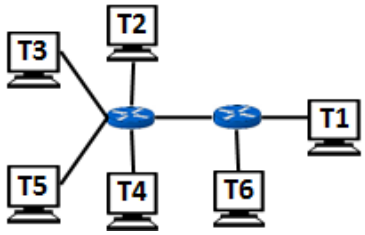
(b) The obtained logical routing tree by running the formulations.

Figure 5.2: The original and the obtained routing tree using matrix S7 with $n = 7$.

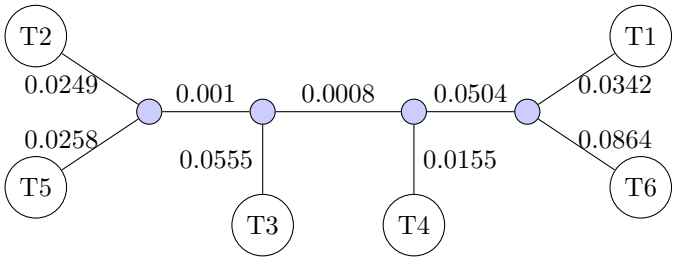
The original routing tree has an internal node (a router) with degree four. The five formulations infer a binary tree and so as we refereed in Chapter 2 a dummy node and edge must be included to convert the tree to a binary one. By observing the logical routing tree obtained we can see that one of the edge weight has a value near to zero (0.00005). If we eliminate this edge and one of the incident nodes, we obtain the original logical routing tree.

Figure 5.3 displays the original logical routing tree used to run the simulation to obtain the values of matrix SS15 with $n = 6$, which is obtained by only considering the terminal

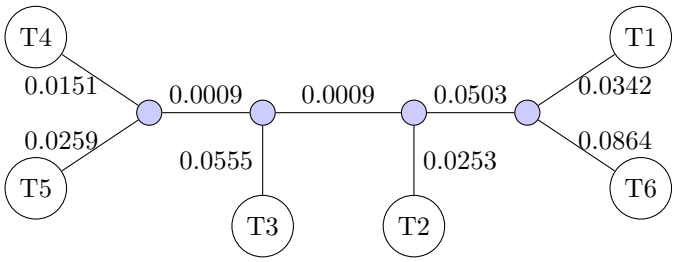
nodes T1 - T6 of Figure 3.11. Figure 5.3 also displays the logical routing tree obtained by running the five formulations. In this case the Path-edges, Path-edges^{+}, Path-edges^{+2} and PL4 formulations obtained the same tree but the Path-weight formulation obtained a slightly different tree.



(a) The original logical routing tree.



(b) The obtained logical routing tree by running the Path-edges, Path-edges^{+}, Path-edges^{+2} and PL4 formulations.



(c) The obtained logical routing tree by running the Path-weight formulation.

Figure 5.3: The original and the obtained routing tree using matrix SS15 with $n = 6$.

The difference between the two logical routing trees obtained is an exchange of terminal nodes T2 and T4. The weights of the edges are also slightly different. These differences are due to the fact that the original routing tree has an internal node (a router) with degree

five. So, in this case, two dummy nodes and two dummy edges must be included to convert the tree to a binary one. Both logical routing trees have two edges whose weights have values near to zero (0.001 and 0.0008 in the tree displayed in (b) and 0.0009 in the tree displayed in (c)). If we eliminate this edges and two incident nodes, we obtain the original logical routing tree.

Finally, we did not found any connection between the fact that a matrix verifies the triangle inequality (Tables 3.1 and 4.1) and the results obtained by the several formulations.

Chapter 6

Feasibility Pump and Local Branching

The computational time the formulations presented in the previous chapter take to find the optimal solution increases significantly as the number of terminal nodes increases. To overcome this limitation we implement two heuristics, the first one, based on the *Feasibility Pump's* ideas [62], finds a feasible solution and the second one, based on the *Local Branching's* ideas [63], improves this feasible solution.

The *Feasibility Pump* is a very efficient heuristic for finding feasible solutions of mixed integer problems. This heuristic generates a sequence of roundings to obtain an integer feasible solution. The *Local Branching* is an exact method to search rapidly for better solutions in the neighborhood of a feasible solution and, as an exact method, is still able to find the optimum solution. This exact method turns in an heuristic if we set a criteria, like a time limit to stop the method before it has exhaustively search all the neighborhoods.

In this chapter, we present the *Feasibility Pump* heuristic and the heuristic we developed applying *Feasibility Pump's* ideas to the Path-edges$^+$ formulation. Then we present the *Local Branching* and the heuristic we developed applying the ideas of the *Local Branching* to the Path-edges$^+$ formulation. We also present the computational results obtained by running the two heuristics using the data instances presented in Section 3.2 and Section 4.2.

6.1 Feasibility Pump

Finding a feasible solution of a Mixed Integer Programming (MIP) problem can be very hard and involve a large computational effort. Fischetti et al. [62] presented a very successful heuristic, that they called the *Feasibility Pump* (FP) to find a feasible solution of generic MIP problems. Despite the fact that Fischetti et al. [62] present the *Feasibility Pump* heuristic for generic MIP problems, they focused essentially on Mixed Binary Programming (MBP) problems. Several authors [8, 14, 64] improved the *Feasibility Pump* heuristic, mostly for generic MIP problems.

Consider the variable index set $N = \{1, \ldots, n\}$ and $\mathcal{I} \subseteq N$.

A generic MIP problem is given by:

$$\begin{aligned} & min \quad c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \mathbb{Z} \qquad \forall i \in \mathcal{I} \end{aligned} \tag{6.1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The integer variables are only variables x_i with $i \in \mathcal{I}$, the index set of the integer variables.

We assume, without loss of generality, that the variables of (6.1) are bounded, that is $l \leq x \leq u,\ l,\ u \in \mathbb{R}^n$.

To facilitate the reading and using a little abuse of notation, we say that a point x is integer if x_i is integer for all $i \in \mathcal{I}$ regardless of the value of the other components that are not obliged to be integer.

A generic MBP problem is given by:

$$\begin{aligned} & min \quad c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0, 1\} \qquad \forall i \in \mathcal{I} \end{aligned} \tag{6.2}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.

As for the MIP problem, in the case of a MBP problem we say that a point x is binary if x_i is binary for all $i \in \mathcal{I}$ regardless of the value of the other components that are not obliged to be binary.

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be the polyhedron associate with the Linear Programing (LP) relaxation of (6.1).

In this section we use the following definitions for rounding and L_1-distance.

Definition 6.1. [15, 62] Let $N = \{1, \ldots, n\}$, $\mathcal{I} \subseteq N$, $x, y \in \mathbb{R}^n$. The *rounding*, $\widetilde{x}$, of the vector x with respect to $\mathcal{I}$, denoted $[x]^{\mathcal{I}}$, is obtained in the following way:

$$\widetilde{x}_i = [x]^{\mathcal{I}}_i = \begin{cases} [x_i] & \text{if} \quad i \in \mathcal{I} \\ x_i & \text{if} \quad i \notin \mathcal{I} \end{cases}$$

where $[\cdot]$ represents scalar rounding to the nearest integer. Notice that only the values of the variables x_i with $i \in \mathcal{I}$ are rounded, the other variables are continuous variables.

Definition 6.2. [15, 62] Let $N = \{1, \ldots, n\}$ and $\mathcal{I} \subseteq N$. The L_1-*distance* of two vectors $x, y \in \mathbb{R}^n$ with respect to $\mathcal{I}$ is given by:

$$\Delta^{\mathcal{I}}(x, y) := \sum_{i \in \mathcal{I}} |x_i - y_i|$$

Notice that the continuous variables x_i, y_i with $i \notin \mathcal{I}$ do not contribute to the distance function $\Delta^{\mathcal{I}}(x, y)$.

The basic idea of the FP heuristic is to sequentially construct two sets of points until a feasible solution of the problem is found. In the first set, the points x^*_{LP} are obtained by solving an LP relaxation ($x^*_{LP} \in P$) and therefore these points satisfy the linear constraints but may not satisfy the integrality constraints. In the second set, the points $\widetilde{x}$ are obtained by rounding the points x^*_{LP} and therefore these points satisfy the integrality constraints but may not satisfy the linear constraints.

The FP heuristic starts by solving the LP relaxation of the MIP problem. If the LP solution, x^*_{LP}, is not integer it is rounded, obtaining a new vector $\widetilde{x} = [x^*_{LP}]^{\mathcal{I}}$. If $\widetilde{x}$ is not feasible, a new point in the LP polyhedron must be obtained. This new point should be the

closest to $\widetilde{x}$ and therefore is obtained by solving a new LP which minimizes the distance $\Delta^{\mathcal{I}}(x, \widetilde{x})$ [8]:

$$\begin{aligned} min \quad & \Delta^{\mathcal{I}}(x,\widetilde{x}) = \sum_{\substack{i\in\mathcal{I}\\ \widetilde{x}_i=l_i}} (x_i - l_i) + \sum_{\substack{i\in\mathcal{I}\\ \widetilde{x}_i=u_i}} (u_i - x_i) + \sum_{\substack{i\in\mathcal{I}\\ l_i<\widetilde{x}_i<u_i}} d_i \\ \text{subject to} \quad & Ax \leq b \\ & d_i \geq x_i - \widetilde{x}_i \qquad \forall i \in \mathcal{I} \\ & d_i \geq \widetilde{x}_i - x_i \qquad \forall i \in \mathcal{I} \\ & l \leq x \leq u \\ & x_i \in \mathbb{Z} \qquad\quad \forall i \in \mathcal{I} \end{aligned} \tag{6.3}$$

The additional variables d_i, $i \in \mathcal{I}$ are introduced to model the nonlinear function $d_i = |x_i - \widetilde{x}_i|$.

If the variables x_i for all $i \in \mathcal{I}$ are binary, the additional variables d_i are not needed, once $0 \leq x_i \leq 1$ for all $i \in \mathcal{I}$ and the new LP to be solved is:

$$\begin{aligned} min \quad & \Delta^{\mathcal{I}}(x,\widetilde{x}) = \sum_{\substack{i\in\mathcal{I}\\ \widetilde{x}_i=0}} x_i + \sum_{\substack{i\in\mathcal{I}\\ \widetilde{x}=1}} (1 - x_i) \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0,1\} \qquad \forall i \in \mathcal{I} \end{aligned} \tag{6.4}$$

Figure 6.1 displays an illustration of the *Feasibility Pump* procedure. The procedure starts with the point $\overline{x}^0$ which is the solution of the LP relaxation of the MIP problem (6.1). Since this point $\overline{x}^0$ is not integer it is rounded and the point $\widetilde{x}^1 = [\overline{x}^0]$ is obtained. The next point is obtained by solving the LP relaxation of (6.3) using $\widetilde{x}^1$. This new point $\overline{x}^1$ is, once again, not integer and therefore, it must be rounded and a new point $\widetilde{x}^2 = [\overline{x}^1]$ is obtained. The integer point $\widetilde{x}^2$ is feasible but the procedure only stops when a LP relaxation solution is integer. So the next step is to solve the LP relaxation of (6.3) using $\widetilde{x}^2$. The solution is $\overline{x}^2$ which is integer and therefore the procedure ends and $\overline{x}^2$ is a feasible solution of the MIP problem (6.1).

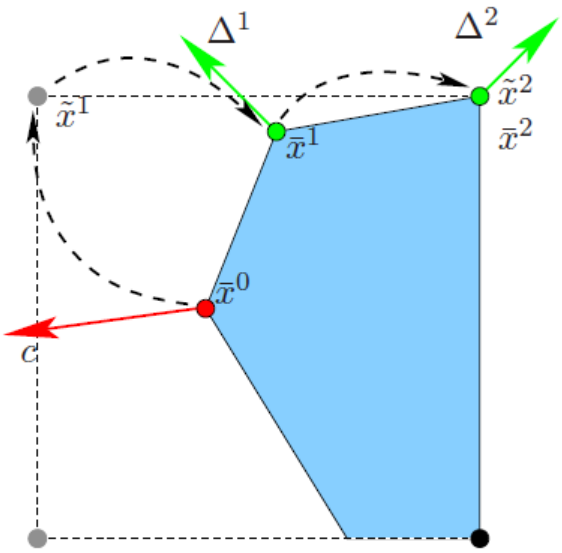


Figure 6.1: Illustration of the *Feasibility Pump* procedure [15].

The FP heuristic presents two problems: stalling issue and cycle issue. The stalling issue occurs when in iteration n the rounding of a new LP solution, $\widetilde{x}^n$ is equal to the integer solution obtained in the previous iteration, that is, $\widetilde{x}^n = \widetilde{x}^{n-1}$. One way to solve this issue, is to change the value of some, randomly chosen, components of $\widetilde{x}$, obtaining, a new $\widetilde{x}$. The number of components which change the value is fixed and is one of the inputs of the FP heuristic. The cycle issue occur when the same sequence of points (LP solutions and rounding points) are obtained over and over again. A cycle is detected by comparing the current $\widetilde{x}$ obtained with the $\widetilde{x}$ found in all previous iterations. If the current $\widetilde{x}$ is equal to a $\widetilde{x}$ found in a previous iteration then the FP heuristic will obtain in the next iteration the same LP solution x^*_{LP} as in the prior iteration. This issue can be solved by applying a so-called *restart*, a perturbation mechanism, every time a cycle is detected. One perturbation mechanism can be to modify some, randomly chosen, components of the current $\widetilde{x}$, obtaining, this way, a new $\widetilde{x}$. The stalling issue is a cycle issue of length one and so these two issues can be detected and solved with the same mechanism.

The MWTR problem is a MBP problem, so we will focus on MBP problems in the remainder of this section.

The Feasibility Pump heuristic is described in Algorithm 1 (in page 107).

The algorithm has as input the parameters TL, T and $maxIT$. TL is the time limit imposed and T is the number of variables to be flipped, that is the number of variables that change their current value, from zero to one or from one to zero. $maxIT$ is the maximum number of iterations allowed.

The first step is to initialize the counter nIT, which indicates the number of the iterations, and the two points, x^*_{LP} and $\widetilde{x}$. The first x^*_{LP} is the solution of the LP relaxation of the MBP problem. Then it is checked if this LP solution x^*_{LP} is binary. If x^*_{LP} is binary the procedure terminates. If it is not the case, then the LP solution is rounded, obtaining the first $\widetilde{x} := [x^*_{LP}]^{\mathcal{I}}$.

After initializing the variables, the so-called *pumping cycles* are performed (the while cycle from line 7 to line 19) until a feasible integer solution is found or the time-limit TL is exceed or the maximum number of iterations is exceed. If the procedure stops because the time-limit is exceed or the maximum number of iteration is exceed, a failure must be reported, once no binary feasible solution was obtained.

In each pumping cycle the counter nIT is incremented and a new point $x^*_{LP} \in P$, with minimum distance from the current binary point $\widetilde{x}$, is obtained. If this new x^*_{LP} is not binary, the procedure verifies if the current binary point $\widetilde{x}$ is equal to $[x^*_{LP}]^{\mathcal{I}}$. If it is not equal, $\widetilde{x}$ is replaced by $[x^*_{LP}]^{\mathcal{I}}$. In case the rounding of the new x^*_{LP} is equal to the current binary point $\widetilde{x}$, a random number $TT \in \left\{\frac{1}{2}T, \ldots, \frac{3}{2}T\right\}$ of binary entries of the current $\widetilde{x}$ are flipped and so stalling problems can be avoided. These entries are chosen to minimize the increase of the total distance $\Delta^{\mathcal{I}}(x^*_{LP}, \widetilde{x})$. The procedures then verifies if a cycle is detected, by checking if the new $\widetilde{x}$ is equal to a $\widetilde{x}$ point found in a previous iterations or if $\Delta^{\mathcal{I}}(x^*_{LP}, \widetilde{x})$ did not decrease by at least 10% in the last three iterations. If a cycle is detected then for all $i \in \mathcal{I}$ an uniformly random value $\rho_i \in [-0.3, 0.7]$ is generated and the variable $\widetilde{x}_i$ is flipped if $|x^*_{LP_i} - \widetilde{x}_i| + \max\{\rho_i, 0\} > 0.5$.

Algorithm 1 The *Feasibility Pump* heuristic ([14, 62])

Input: $TL > 0$, the time limit, $T > 0$, the number of variables to be flipped and $maxIT > 0$, the maximum number of iterations

initialize $nIT := 0$ and $x^*_{LP} := argmin\{c^T x : Ax \leq b\}$
if (x^*_{LP} is binary) **then**
    return x^*_{LP}
else
    $\widetilde{x} := [x^*_{LP}]^{\mathcal{I}}$
end if
while (time $< TL$ and $nIT < maxIT$) **do**
    $nIT := nIT + 1$
    $x^*_{LP} := argmin\{\Delta^{\mathcal{I}}(x, \widetilde{x}) : Ax \leq b\}$
    if (x^*_{LP} is binary) **then**
        return x^*_{LP}
    else if ($\exists i \in \mathcal{I} : [x^*_{LP_i}] \neq \widetilde{x}_i$) **then**
        $\widetilde{x} := [x^*_{LP}]^{\mathcal{I}}$
    else
        flip the $TT = rand\left(\frac{1}{2}T, \frac{3}{2}T\right)$ entries $\widetilde{x}_i$ ($i \in \mathcal{I}$) with highest $|x^*_{LP_i} - \widetilde{x}_i|$
    end if
    if a cycle is detected **then**
        flip $\widetilde{x}_i$ in case $|x^*_{LP_i} - \widetilde{x}_i| + \max\{\rho_i, 0\} > 0.5$,
        where $\rho_i = rand(-0.3, 0.7)$, $\forall i \in \mathcal{I}$
    end if
end while

6.1.1 Application to MWTR problem with use of the Path-edges$^+$ formulation

We applied the FP heuristic to the MWTR problem and used the Path-edges$^+$ formulation presented in Chapter 5 to obtain feasible solutions. In order to obtain a high quality feasible solution in a reasonable small time we tried several combinations, namely by applying the FP heuristic to the Path-edges formulation, that is the Path-edges$^+$ formulation without the valid equalities and inequalites presented in Section 5.3 or by using only some of the equalities and inequalities from Section 5.3. Comparing the results obtained by the several combinations, we concluded that some of these combinations obtained a feasible solution in less time than the one that uses the Path-edges$^+$ formulation, but the quality of the feasible solution obtained was quite poor. Since the performance of the *Local Branching* heuristic depends highly on the first feasible solution used, we chose to use the Path-edges$^+$ formulation. The time gained in some of these combinations did not compensate for the poor quality of the solution obtained, once the time gained in the FP heuristic would be spend when running the *Local Branching* heuristic to improve the feasible solution obtained.

In the Path-edges$^+$ formulation we have three types of binary variables. We have the variables x_{ij}, $i \in V_a$, $j \in V$, $i < j$ that indicate whether edge $\{i, j\}$ belongs to the tree solution, variables p_{ij}^{ℓ} that indicate whether the path P_{ij} connecting terminal node i to terminal node j has exactly ℓ edges and the binary flow variables $f_{ij}^{k\ell}, \forall i, j \in V_a \cup \{k, \ell\}, k, \ell \in V_t$, $i \neq j$ and $k < \ell$, that indicate whether the flow traverses the edge $\{i, j\}$ belonging to the path connecting terminal node k to terminal node ℓ in the direction from node i to node j. Let y be a vector formed of all the binary variables of the Path-edges$^+$ formulation, that is $y = (x, f, p)$. So in this case the FP heuristic will generate two type of points, $y_{LP}^* = (x_{LP}^*, f_{LP}^*, p_{LP}^*)$ (LP solutions) and $\widetilde{y} = (\widetilde{x}, \widetilde{f}, \widetilde{p})$ (roundings of the LP solutions).

In Algorithm 2 we present the algorithm we developed based on the *Feasibility Pump's* ideas to obtain a feasible solution of the MWTR problem. Our algorithm is very similar

to the Algorithm 1 taking the fact that we initialize the first y^*_{LP} differently, detect cycles differently and apply different perturbation mechanism.

The FP heuristic as presented in Algorithm 1 initializes y^*_{LP} with the LP relaxation solution of the MBP problem. To improve the quality of the feasible solution obtained by the FP heuristic we initialize the y^*_{LP} with an improved solution of the LP relaxation solution. First we obtain the LP relaxation solution of the Path-edges$^+$ formulation, $y^{LP} = (x^{LP}, f^{LP}, p^{LP})$ and fix the values of the component $x^*_{LP_{ij}},\ i \in V_a,\ j \in V,\ i < j$ of y^*_{LP} to be zero if $x^{LP}_{ij} < 0.1$. The remaining components of the initial y^*_{LP} are obtained through a new LP relaxation solution of the Path-edges$^+$ formulation.

To detect cycles we use the process used by Santos [117] who applied the FP heuristic to the Weight Constrained Minimum Spanning Tree Problem. Santos [117] detected cycles by comparing the value $\Delta(y^*_{LP}, \widetilde{y})$ of the actual iteration with the values obtained in the previous three iteration. If the value $\Delta(y^*_{LP}, \widetilde{y})$ is equal in three consecutive iterations then a perturbation mechanism is applied.

The particular case of cycles of length one are detected by verifying if the $\widetilde{y}$ of the actual iteration is equal to the solution of the previous iteration.

We applied two different perturbation mechanism according to the type of cycle detected. In case of a cycle of length one, we applied a perturbation only to the binary components $\widetilde{x}$ of $\widetilde{y}$, keeping the value of the remaining components. The perturbation we applied is the following:

$$\widetilde{x}_{ij} = \begin{cases} 0 & \text{if} \quad x^*_{LP_{ij}} < 0.1 \text{ or } (x^*_{LP_{ij}} > 0.5 \text{ and } x^*_{LP_{ij}} < 0.9) \\ 1 & \text{otherwise} \end{cases}$$

In case of the other type of cycles, we flip the value of the components $\widetilde{f}$ and $\widetilde{p}$ of $\widetilde{y}$ keeping the value of the components $\widetilde{x}$ of $\widetilde{y}$ unchanged.

Finally, we set the time limit TL at 600 seconds and the maximum number of iterations at 1000 iterations.

The computational results will be presented in Section 6.3.

Algorithm 2 Algorithm to obtain a feasible solution of the MWTR problem.

Input: The time limit TL and the maximum number of iterations $maxIT$.

initialize $nIT := 0$
Obtain the LP relaxation solution, $y^{LP} = (x^{LP}, f^{LP}, p^{LP})$, using Path-edges$^+$ formulation
if ($x^{LP}_{ij} < 0.1$, $i \in V_a$, $j \in V$, $i < j$) **then**
    $x^*_{LP_{ij}} = 0$
end if
Obtain a new LP relaxation solution, y^*_{LP}, of the Path-edges$^+$ formulation
if (y^*_{LP} is binary) **then**
    return y^*_{LP}
else
    $\widetilde{y} := [y^*_{LP}]$
end if
while (time $< TL$ and $nIT < maxIT$) **do**
    $nIT := nIT + 1$
    $y^*_{LP} := argmin\{\Delta(y, \widetilde{y}) :$ constraints of the Path-edges$^+$formulation$\}$
    if (y^*_{LP} is binary) **then**
        return y^*_{LP}
    else
        if ($[y^*_{LP}] \neq \widetilde{y}$) **then**
            $\widetilde{y} := [y^*_{LP}]$
        else if $x^*_{LP_{ij}} < 0.1$ or ($x^*_{LP_{ij}} > 0.5$ and $x^*_{LP_{ij}} < 0.9$) **then**
            $\widetilde{x}_{ij} = 0$
        else
            $\widetilde{x}_{ij} = 1$
        end if
        if a cycle is detected **then**
            flip the values of the components $\widetilde{f}$ and $\widetilde{p}$ of $\widetilde{y}$
        end if
    end if
end while

6.2 Local Branching

The feasible solution obtained using the FP heuristic can be improved by using a local search heuristic. A local search heuristic iteratively defines a neighborhood of a feasible solution and searches for a better feasible solution in that neighborhood. Then this new feasible solution is used in the next iteration.

Fischetti and Lodi [63] introduced a procedure, called *Local Branching* (LB), which is in the spirit of local search heuristics. The neighborhood of a feasible solution is obtained by adding constraints to the original problem. Fischetti and Lodi [63] designated these constrains as *local branching* cuts.

Consider the following generic MIP with $0-1$ variables:

$$\begin{aligned} & min \quad c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0,1\} \qquad \forall i \in \mathcal{B} \neq \emptyset \\ & x_i \geq 0, \text{integer} \qquad \forall i \in \mathcal{I} \\ & x_i \geq 0 \qquad \forall i \in \mathcal{C} \end{aligned} \tag{6.5}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. $N = \{1, \ldots, n\}$ is the variable index set and is partitioned into $(\mathcal{B}, \mathcal{I}, \mathcal{C})$, where $\mathcal{B}$, $\mathcal{I}$ and $\mathcal{C}$ are the index set of the binary, general integer and continuous variables, respectively. $\mathcal{B}$ is a nonempty set while $\mathcal{I}$ and $\mathcal{C}$ may be empty sets.

Let X be the set of feasible solutions of the generic MIP problem (6.5) and consider a feasible reference solution $\overline{x} \in X$ of that problem.

Definition 6.3. The *binary support*, $\overline{S}$ of $\overline{x}$ is given by $\overline{S} = \{i \in \mathcal{B} : \overline{x}_i = 1\}$.

$\overline{S}$ is the set of indices of the components of $\overline{x}$ with value one. Consequently, $\mathcal{B} \setminus \overline{S}$ is the set of indices of the components with value zero.

Definition 6.4. Let k be a positive integer parameter. The *local branching constraint* is

given by

$$\Delta^{\mathcal{B}}(x, \overline{x}) = \sum_{i \in \overline{S}} (1 - x_i) + \sum_{i \in \mathcal{B} \setminus \overline{S}} x_i \leq k \tag{6.6}$$

The left-hand side of the constraint (6.6) counts the number of binary variables which flipped their value, with respect to $\overline{x}$, from one to zero or from zero to one.

When the cardinality of the binary support $\overline{S}$ of any feasible solution is constant the local branching constraint (6.6) is given by:

$$\Delta^{\mathcal{B}}(x, \overline{x}) = \sum_{i \in \overline{S}} (1 - x_i) \leq \frac{k}{2} \tag{6.7}$$

The left-hand side of the constraint (6.7) counts the number of binary variables which flipped their value, with respect to $\overline{x}$, from one to zero. Since the cardinality of $\overline{S}$ is constant, whenever a variable x_j, $j \in \overline{S}$ flips value from one to zero another variable must flip value from zero to one.

Definition 6.5. Given a positive integer parameter k, the *k-OPT neighborhood* $\mathcal{N}(\overline{x}, k)$ of a feasible reference solution $\overline{x} \in X$ is given by

$$\mathcal{N}(\overline{x}, k) = \{x \in X : \sum_{i \in \overline{S}} (1 - x_i) + \sum_{i \in \mathcal{B} \setminus \overline{S}} x_i \leq k\}$$

So the *k-OPT* neighborhood is the set of feasible solutions of the generic MIP problem (6.5) that satisfy the local branching constraint (6.6).

A local branching constraint divides the solution space X into two disjoint parts, the *k-OPT* neighborhood $\mathcal{N}(\overline{x}, k)$ and the set $X \setminus \mathcal{N}(\overline{x}, k) = \{x \in X : \Delta^{\mathcal{B}}(x, \overline{x}) > k\}$ [15]. Consequently, the local branching constraint divides a MIP problem in two subproblems, $(SP1) : \max\{c^T x : x \in \mathcal{N}(\overline{x}, k)\}$ and $(SP2) : \max\{c^T x : x \in X \setminus \mathcal{N}(\overline{x}, k)\}$.

The LB procedure uses local branching constraints with the intention of rapidly finding good feasible solutions of a MIP problem [94]. This procedure starts with a feasible solution, $\overline{x}^1$, that can be obtained, for example, through the FP heuristic. Using the feasible solution $\overline{x}^1$, the local branching constraint, $\Delta^{\mathcal{B}}(x, \overline{x}^1) \leq k$, is defined. To solve the two subproblems, $(SP1) : \max\{c^T x : x \in \mathcal{N}(\overline{x}^1, k)\}$ and $(SP2) : \max\{c^T x : x \in X \setminus \mathcal{N}(\overline{x}^1, k)\}$, originated

from the local branching constraint, the LB procedure uses a MIP solver like, for example, FICO Xpress [1]. The first subproblem to be solved is the subproblem $(SP1)$ and if the optimal solution, $\overline{x}^2$, of this subproblem is better than the feasible solution $\overline{x}^1$ then this new solution is used to define a new local branching constraint $\Delta^{\mathcal{B}}(x, \overline{x}^2) \leq k$ which is added to the subproblem $(SP2)$. This way two new subproblems are defined, $(SP3) : \max\{c^T x : x \in (X \setminus \mathcal{N}(\overline{x}^1, k)) \cap \mathcal{N}(\overline{x}^2, k)\}$ and $(SP4) : \max\{c^T x : x \in (X \setminus \mathcal{N}(\overline{x}^1, k)) \setminus \mathcal{N}(\overline{x}^2, k)\}$. This procedure continues as long as better feasible solutions are found and is illustrated in Figure 6.2. The nodes of the tree are numbered in the order the subproblems are generated and processed. The subproblem of the right branch of the tree is only solved if no improved solution is found in the subproblem of the left branch. Note that following this scheme the LB procedure is an exact method.

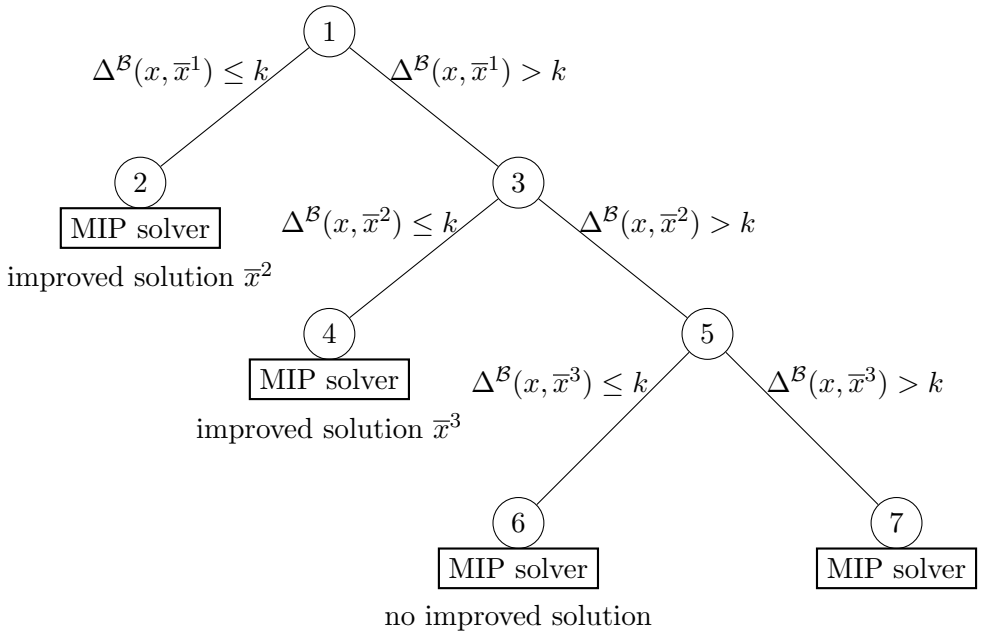


Figure 6.2: The basic Local Branching scheme [63, 94].

To rapidly find better feasible solutions, the parameter k must be carefully chosen. Small values of k imply small neighborhoods $\mathcal{N}(\overline{x}, k)$ and therefore an easy subproblem that can be solve in short time. But a small neighborhood may not contain a better feasible solution than the current one. In the other hand, large k increases the neighborhood size

and therefore the time to solve the subproblem associate to the neighborhood. Fischetti and Lodi [63] concluded through computational experience that values of k in the range $[10, 20]$ are the most effective.

Algorithm 3 presents the basic Local Branching procedure.

Algorithm 3 Basic Local Branching procedure.

Input: The parameter k and a feasible solution $\overline{x}$.

function BASIC LB(k, $\overline{x}$)
  initialize $x^* = \overline{x}$, the reference solution
    $END = false$, boolean variable to indicate when to stop the repeat cycle
    $Y = \{x \in X : \Delta^{\mathcal{B}}(x, \overline{x}) \leq k\}$, the k-OPT neighborhood and
    $Z = \{x \in X : \Delta^{\mathcal{B}}(x, \overline{x}) > k\}$
  repeat
    $\overline{x} := argmin\{c^T x : x \in Y\}$
    if $(c^T \overline{x} < c^T x^*)$ **then**
      $x^* = \overline{x}$
      $Y = \{x \in Z : \Delta^{\mathcal{B}}(x, \overline{x}) \leq k\}$ and $Z = \{x \in Z : \Delta^{\mathcal{B}}(x, \overline{x}) > k\}$
    else
      $END = true$
      $\overline{x} := argmin\{c^T x : x \in Z\}$
      if $(c^T \overline{x} < c^T x^*)$ **then**
        $x^* = \overline{x}$
      end if
    end if
  until END
  return x^*
end function

As referred before, the basic LB procedure is an exact method but it can be used as a heuristic if a criteria is set to stop the method before it has exhaustively search all the

neighborhoods.

Fischetti and Lodi [63] presented two mechanisms to improve the performance of the LB procedure when used as a heuristic: imposing a time limit on the left-branch nodes of the tree presented in Figure 6.2 and diversification mechanisms.

Depending on the parameter k, in some case, finding the exact solution of the left-branch node of the tree presented in Figure 6.2 can be very time consuming. So if using the LB procedure as a heuristic, a time limit can be imposed to solve the subproblems. This time limit imposition also allows to use larger values of k, since, in this case, the neighborhood is not completely explored. If the optimum solution of the subproblem can not be found within the time limit imposed, two situations can occur: an improved feasible solution is found or no improved feasible solution is found.

If no improved feasible solution is found, Fischetti and Lodi [63] suggest to reduce the size of the neighborhood, in order to accelerate its exploration. This reduction is achieved through the introduction of the local branching constraint $\Delta^{\mathcal{B}}(x, \overline{x}) \leq \lfloor \frac{k}{2} \rfloor$. This situation is illustrated in Figure 6.3.

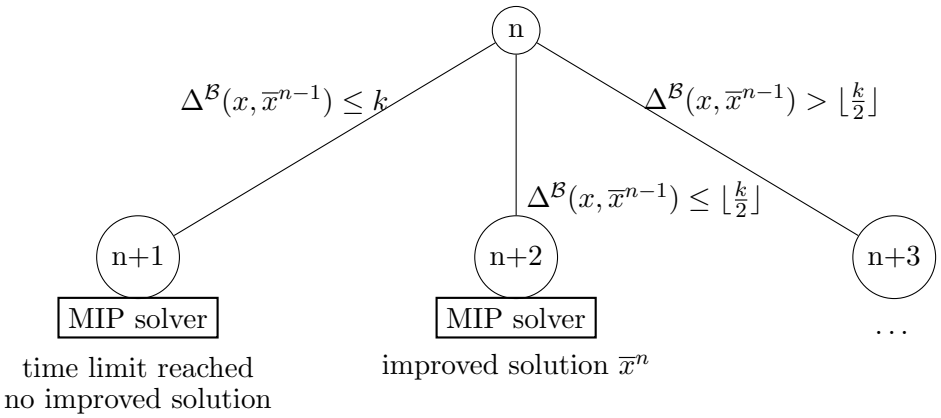


Figure 6.3: Local Branching scheme when the time limit is reached and no improved solution has been found [63].

If an improved feasible solution is found then a new left-branch node associated with the new improved feasible solution is created, as we can see in Figure 6.4.

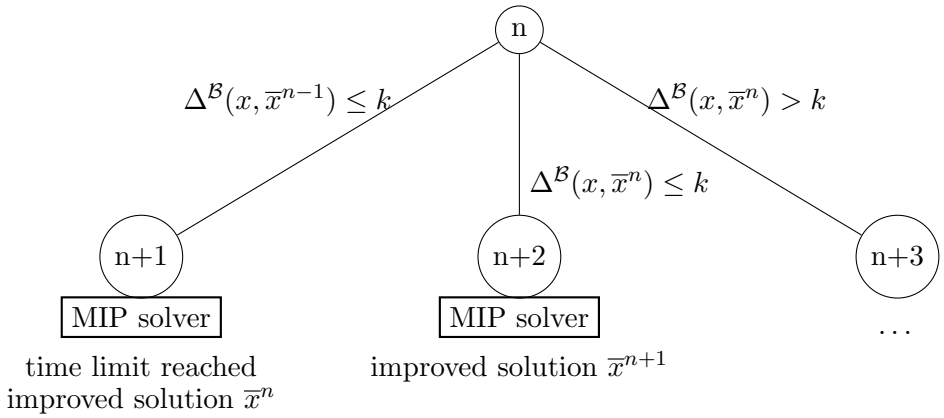


Figure 6.4: Local Branching scheme when the time limit is reached and an improved solution has been found [63].

Notice that in this case the neighborhood associated with node $n+1$ was not completely explored and therefore it can not be excluded, since the optimum solution may still be in it.

The diversification mechanisms are applied when no improved feasible solution can be found by completely exploring the neighborhood defined in a left-branch node of the search tree. In the tree presented in Figure 6.2 this would be the case in node 6.

Fischetti and Lodi [63] proposed two different kinds of diversification mechanisms, a *soft* diversification and a *strong* diversification. The strong diversification should only be applied if even after applying the soft diversification no improved feasible solution could be found.

The soft diversification consists in increasing the size of the neighborhood by, for example an amount of $\lceil \frac{k}{2} \rceil$. This increase is achieved through the introduction of a new left-branch node associated to the local branching constraint $\Delta^{\mathcal{B}}(x, \overline{x}) \leq k + \lceil \frac{k}{2} \rceil$. If still no improved feasible solution can be found, a strong diversification is applied, which consists in taking another solution (typically worse) in the vicinity of the current reference solution and restart the LB procedure with this solution. This other solution is obtained by introducing the constraint $\Delta^{\mathcal{B}}(x, \overline{x}) \leq k + 2\lceil \frac{k}{2} \rceil$ and abort the exploration as soon as

the first solution is found.

6.2.1 Application to MWTR problem with use of the Path-edges$^+$ formulation

To improve the feasible solution obtained by using the FP heuristic, we applied the LB procedure to the MWTR problem and used the Path-edges$^+$ formulation presented in Chapter 5. Since we used the Path-edges$^+$ formulation in the FP heuristic we also used this formulation when applying the LB procedure. Once again, we tried several combinations in order to obtain an improved high quality feasible solution in a reasonable time. By analyzing the results of the several combinations, we noticed that if we would not include, in the LB heuristic, the inequalities (5.45) and (5.46), presented in Section 5.3 we obtained the same feasible solution but in less time. So we used the Path-edges$^+$ formulation without these two inequalities.

Regarding the parameter k, after several experiences, we concluded that the choice of the value of k depends on the value of n. When the value of n is small, the value of k can also be small and even so the LB heuristic finds an improved feasible solution in the neighborhood. But for values of n larger this is not the case and therefore the value of k must be larger to find an improved feasible solution in the neighborhood. Since we used the local branching constraints (6.7) the values we used for k were $k = 3$ when $n < 12$ and $k = 5$ when $n \geq 12$.

We imposed a time limit for solving each subproblem, the node time limit, varying according to the value of n. We first set a time limit of 1000 seconds but noticed that for $n \geq 14$ no feasible solution of the subproblem was found within this time limit, even when reducing the size of the neighborhood. So we used a time limit of 1000 seconds if $n < 14$ and 1400 if $n \geq 14$. We also imposed a total time limit for the heuristic to run. To be able to compare the quality of the feasible solution obtained using the LB heuristic with the solution obtained using the exact formulation, we set a total time limit of 7000 seconds.

When the time limit is reached and a feasible solution is found but not an improved

one, we did not reduce the neighborhood as suggested by Fischetti and Lodi [63] because in the experiences we performed, even the reduction of the neighborhood did not improve the reference solution. So, to avoid wasting time, instead of reducing the neighborhood we used a soft diversification.

We reduced the neighborhood whenever no feasible solutions were found in the time limit imposed unless the neighborhood was already reduced in the previous iteration. When no feasible solution is found after reducing the neighborhood size, the heuristic stops and returns the actual best feasible solution. The reduction of the neighborhood size is done by reducing one unit the parameter of the right hand side of the local branching constraint.

We only used soft diversification, since we realized, in the experiences we performed, that the strong diversification is time consuming and most of the time there is no improvement in the feasible solution returned by the LB heuristic relatively to the one obtained when no strong diversification is used. Soft diversification is used every time no improved feasible solution is found, unless a soft diversification was performed in the previous iteration. When no improved feasible solution is found after applying a soft diversification, the heuristic stops and returns the actual best feasible solution. The soft diversification is performed by increasing the parameter of the right hand side of the local branching constraint of two units.

As for the FP heuristic we consider y to be a vector formed of all the binary variables of the Path-edges$^+$ formulation, that is $y = (x, f, p)$. Let Z be the set of all feasible solutions of the MWTR problem regarding to the Path-edges$^+$ formulation and $DZ(y)$ be the value of the objective function $\sum_{k \in V_t} \sum_{\substack{\ell \in V_t \\ \ell > k}} d_{k\ell} \sum_{i=2}^{n-1} 2^{-i} \cdot p^i_{k\ell}$ at $y = (x, f, p) \in Z$.

In Algorithm 4 we present the heuristic we developed based on the *Local Branching* procedures' ideas to improve the feasible solution of the MWTR problem obtained using the FP heuristic. This heuristic receives as input the parameter k, the node time limit nTL, the total time limit tTL, and the initial feasible solution $\overline{y}$ and returns on output the best feasible solution found (x^*). The heuristic consists of a repeat-until cycle which is performed until the total time limit is exceeded or the boolean variable END has value

true. This variable becomes true, after performing a soft diversification if no improved feasible solution is found or after reducing the neighborhood, if no feasible solution can be found. Since we increase the value of k two units when performing a soft diversification and reduce the value of k one unit when reducing the neighborhood, after performing a soft diversification if no feasible solution is found, the value of k is reduced one unit and after that if no feasible solution is found the variable END becomes true.

At each iteration a new local branching constraint is added and the MWTR problem with the local branching constraints is solved, using the function MIPSOLVE(nTL). This function receives on input the node time limit (nTL) and returns on output the optimum value or the best value found ($\overline{y}$) within the time limit imposed, along with the final optimization status ($state$). To solve the MWTR problem with the local branching constraints, the function MIPSOLVE(nTL) uses the Path-edges$^+$ formulation without the inequalities (5.45) and (5.46), presented in Section 5.3.

When running the MIPSOLVE function, three final optimization status may arise: "optimum found", "feasible solution found" and "no feasible solution found". The status "optimum found" and "feasible solution found" have the same consequences to the following iteration. The only difference is that when the solution is improved and the neighborhood was completely searched (optimum solution is found) the local branching constraint can be reversed. If the MIPSOLVE function, because of the node time limit imposition, only returns an improved feasible solution the local branching constraint can not be reversed. This is because the optimum solution could still be in that neighborhood. In this case the local branching constraint $\Delta^{\mathcal{B}}(y, \overline{y}) \leq s$ is replaced by $\Delta^{\mathcal{B}}(y, \overline{y}) \geq 1$, to cut off the reference solution $\overline{y}$.

Whenever an improved solution is found the value of y^* is updated with the value of the new improved feasible solution and the value of s, the current parameter of the right hand side of the local branching constraint, becomes equal to k if it is not already equal.

In Algorithm 4 the case $state$="no feasible solution found" is developed from line 23 to line 28.

Algorithm 4 Heuristic to improve the feasible solution of the MWTR problem.

function LB HEURISTIC(k, nTL, tTL, $\overline{y}$)
  initialize $y^* = \overline{y}$, the reference solution
    $END = false$, boolean variable to indicate when to stop the repeat cycle
    $s = k$, the current parameter of the right side of the local branching constraint
  repeat
    Add the local branching constraint $\Delta^{\mathcal{B}}(y, \overline{y}) \leq s$
    ($\overline{y}$, *state*)=MIPSOLVE(nTL)
    if (*state*="optimum found" or *state*="feasible solution found") **then**
      if ($DZ(\overline{y}) < DZ(y^*)$) **then**
        $y^* = \overline{y}$
        if (*state*="optimum found") **then**
          reverse the last local branching constraint into $\Delta^{\mathcal{B}}(y, \overline{y}) \geq s + 1$
        else
          replace the last local branching constraint $\Delta^{\mathcal{B}}(y, \overline{y}) \leq s$ by $\Delta^{\mathcal{B}}(y, \overline{y}) \geq 1$
        end if
        if ($s \neq k$) **then**
          $s = k$
        end if
      else if ($s = k$) **then**
        delete the last local branching constraint $\Delta^{\mathcal{B}}(y, \overline{y}) \leq s$
        $s = s + 2$
      else
        $END = true$
      end if
    else if ($s = k + 2$ or $s = k$) **then**
      delete the last local branching constraint $\Delta^{\mathcal{B}}(y, \overline{y}) \leq s$
      $s = s - 1$
    else
      $END = true$
    end if
  until (END or time $> TL$)
  return y^*
end function

6.3 Computational results

In this section, we present the computational results we obtained running the FP heuristic presented in Algorithm 2 and the LB heuristic presented in Algorithm 4. The computational tests were performed on an Intel(R) Core(TM) i7-3770 CPU 3.40 GHz processor and 16Gb of RAM.

The two heuristics were implemented using the Mosel language and solved with FICO Xpress 7.8 [1] (Xpress-IVE 1.24.06 64 bit, Xpress-Optimizer 27.01.02 and Xpress-Mosel 3.8.0). We compare the performance of the two heuristics using the Path-edges$^+$ formulation.

We used the instances presented in Chapter 3 and Chapter 4.

The computational results are summarized in Tables 6.1 - 6.6 in which the first column, labeled **M**, indicates the name of the matrix instance used and the second column, labeled $|V_t|$, indicates the size of the instance. The third, forth and fifth columns concern the results of the FP heuristic, from the sixth to the ninth columns the results of the LB heuristic are presented and in the tenth and eleventh column we present again the results of the Path-edges$^+$ formulation to facilitate the comparison of the values obtained.

The columns labeled **T** show the execution time, in seconds, used to solve the instance and having a maximum runtime of 600 seconds for the FP heuristic and 7000 seconds for the LB heuristic. The columns labeled **W** present the optimum value obtained or the best value obtained within the runtime limit, where W stands for $\sum_{i \in V_a} \sum_{\substack{j \in V \\ j>i}} w_{ij}$. For matrices A15M887 and A20M887 for $n = 12$ the Path-edges$^+$ formulation did not obtain any feasible solution within the runtime of 7200 seconds and therefore for these two instances we used the best value obtained by the Path-edges^{+2} formulation. The column labeled TT show the total execution time, that is, the sum of the execution time of the FP heuristic with the execution time of the LB heuristic. The columns labeled **GAP** present the gap between the value obtained by the heuristic and the best lower bound value: $GAP = \frac{W_H - LB}{W_H} \times 100$, where W_H represents the best value obtained by the heuristic within the runtime imposed

and LB the best lower bound value.

Table 6.1: Computational results of the heuristics for data from the phylogenetics application.

		FP heuristic			LB heuristic				Path-edges$^+$				FP heuristic			LB heuristic				Path-edges$^+$	
M	$\lvert V_t\rvert$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**	**M**	$\lvert V_t\rvert$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
M391	5	0	0.0487	0	0	0	0.0487	0	0.02	0.0487	A10M391	5	0	0.0515	0	0	0	0.0515	0	0	0.0515
	6	0.03	0.0592	0	0.08	0.11	0.0592	0	0.05	0.0592		6	0.03	0.0654	0.5	1.34	1.37	0.0622	0	0.06	0.0622
	7	0.11	0.0657	3.6	2.5	2.61	0.0634	0.1	2	0.0634		7	0.13	0.0672	0.6	3.87	4	0.0668	0	2.42	0.0668
	8	0.41	0.0765	8.8	13.46	13.87	0.0705	1	21.29	0.0698		8	0.61	0.0829	11.4	17.11	17.72	0.0735	0.1	14.87	0.0734
	9	0.86	0.1014	6.1	56.93	57.79	0.0952	0	11.06	0.0952		9	1.15	0.1111	10.2	55.43	56.58	0.0998	0	9.89	0.0998
	10	12.04	0.1233	13.4	262.47	274.51	0.1068	0	557.26	0.1068		10	2.81	0.1265	11.1	288.58	291.39	0.1124	0	197.67	0.1124
	11	10.06	0.1533	12.7	1073.05	1083.11	0.1338	0	2837.07	0.1338		11	6.93	0.1628	13.4	1368.09	1375.02	0.141	0	1197.99	0.141
	12	22.17	0.1619	19.1	6908.38	6930.55	0.1378	0.9	7200	0.1378		12	9.22	0.1579	12.7	5445.05	5454.27	0.1454	1	7200	0.1461
	13	193.66	0.1809	21.5	7000	7193.66	0.1557	2.6	7200	0.151		13	27.05	0.1826	18.2	6414.28	6441.33	0.1586	2	7200	0.1595
	14	77.45	0.1901	23.9	7000	7077.45	0.1684	9.4	7200	0.16		14	72.17	0.2008	24.3	7000	7072.17	0.1808	9.5	7200	0.167
	15	104.04	0.1926	22	3000.06	3104.1	0.1926	15.7	7200	0.1714		15	464.77	0.2023	21.9	6002.45	6467.22	0.1786	7	7200	0.1838
Primate	5	0.02	0.0843	0	0.01	0.03	0.0843	0	0	0.0843	A10Pri	5	0.02	0.0893	0	0	0.02	0.0893	0	0	0.0893
	6	0.03	0.1195	11.4	0.83	0.86	0.1059	0	0.01	0.1059		6	2	0.1501	25.7	0.02	2.02	0.1115	0	0.05	0.1115
	7	0.11	0.1549	17.9	5.3	5.41	0.1272	0	0.25	0.1272		7	0.22	0.169	20.5	6.58	6.8	0.1343	0	0.42	0.1343
	8	0.36	0.1667	23.2	9.44	9.8	0.128	0	1.59	0.128		8	0.25	0.1541	12.3	22.14	22.39	0.1351	0	1.89	0.1351
	9	0.77	0.197	34.2	76.6	77.37	0.1348	3.8	14.18	0.1296		9	0.83	0.1958	30.4	78.36	79.19	0.1397	2.4	12.93	0.1364
	10	2.42	0.2154	39.7	362.14	364.56	0.1326	2.1	91.68	0.1298		10	2.68	0.2483	45.2	280.02	282.7	0.1404	3.1	138.33	0.1361
	11	26.24	0.2903	41.5	713.78	740.02	0.17	0	598.11	0.17		11	3.87	0.2741	34.5	1163.34	1167.21	0.1794	0	574.74	0.1794
	12	12.87	0.3143	38.6	6202.03	6214.9	0.2058	1.5	7200	0.2058		12	7.44	0.2632	23	4751	4758.44	0.2162	1.7	7200	0.2162
M887	5	0	0.1033	0	0.03	0.03	0.1033	0	0.01	0.1033	A10M887	5	0	0.109	0	0.03	0.03	0.109	0	0.01	0.109
	6	0.05	0.119	2.8	1.17	1.22	0.1157	0	0.61	0.1157		6	0.06	0.1254	3.1	1.2	1.26	0.1215	0	0.66	0.1215
	7	0.08	0.1633	16.5	4.55	4.63	0.1364	0	1.34	0.1364		7	0.11	0.1788	19.9	3.96	4.07	0.1432	0	1.51	0.1432
	8	0.31	0.214	17.6	20.06	20.37	0.1763	0	17.43	0.1763		8	0.33	0.2078	10.6	24.12	24.45	0.1858	0	9.45	0.1858
	9	1.12	0.2182	14.8	86.97	88.09	0.186	0	13.56	0.186		9	3.25	0.227	13.1	62.79	66.04	0.1972	0	9.23	0.1972
	10	3.98	0.2327	11.5	266.12	270.1	0.2078	0.9	178.87	0.2059		10	9.02	0.2361	8.7	224.59	233.61	0.2156	0	276.4	0.2156
	11	7.55	0.2567	16.8	1493.73	1501.28	0.2147	0.6	2085.04	0.2135		11	6.55	0.2834	20.9	1352.8	1359.35	0.2243	0	2253.32	0.2243
	12	20.17	0.2754	30.4	6000.17	6020.34	0.218	2.6	7200	0.2146		12	10.72	0.2631	23.4	4893.51	4904.23	0.2243	2.6	7200	0.229
	13	17.21	0.2842	26.9	7000	7017.21	0.2376	3.8	7200	0.238		13	62.03	0.3005	27.4	3999.6	4061.63	0.2537	4.4	7200	0.2454
	14	125.41	0.3151	32.8	7000	7125.41	0.2784	16.7	7200	0.2527		14	318.93	0.3345	33.5	7000	7318.93	0.2755	7.8	7200	0.2668
	15	235.01	0.3247	29.7	4501.68	4736.69	0.305	15.9	7200	0.2844		15	273.53	0.351	31.5	6001.81	6275.34	0.3208	15.6	7200	0.2857

Table 6.2: Computational results of the heuristics for data from the phylogenetics application (continuation).

M	$\|V_t\|$	FP heuristic			LB heuristic				Path-edges$^+$		M	$\|V_t\|$	FP heuristic			LB heuristic				Path-edges$^+$	
		T	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**			**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
A15M391	5	0.01	0.0529	0	0	0.01	0.0529	0	0.01	0.0529	A20M391	5	0.02	0.0543	0	0.02	0.04	0.0543	0	0	0.0543
	6	0.03	0.0664	4	1.25	1.28	0.0637	0	0.08	0.0637		6	0.03	0.0674	3.2	0.9	0.93	0.0652	0	0.06	0.0652
	7	0.11	0.0688	0.5	2.89	30	0.0685	0	1.37	0.0685		7	0.09	0.073	3.8	6.27	6.36	0.0702	0	1.4	0.0702
	8	0.37	0.0851	11.6	15.71	16.08	0.0755	0.4	16.43	0.0752		8	0.55	0.0908	15.2	23.06	23.61	0.0772	0.3	11.59	0.077
	9	1.23	0.1213	15.8	69.97	71.2	0.1021	0	13.37	0.1021		9	1.09	0.1193	12.5	139.81	140.9	0.1048	0.3	10.39	0.1044
	10	2.34	0.1305	11.7	448.89	451.23	0.1153	0	211.55	0.1153		10	5.05	0.1387	14.9	313.92	318.97	0.1177	0	274.01	0.1181
	11	4.79	0.1645	12.9	1134.28	1139.07	0.1447	1	2032.65	0.1433		11	7.8	0.1667	12	1672.18	1679.98	0.14669	0	1793.47	0.1467
	12	34.12	0.1736	18.6	6973.2	7007.32	0.1481	1.1	7200	0.1492		12	13.85	0.1735	16.5	7000	7013.85	0.1514	1.1	7200	0.1515
	13	28.16	0.2013	24	6314.64	6342.8	0.1647	2.1	7200	0.1642		13	63.21	0.2013	22.2	6113.15	6176.36	0.1715	3.2	7200	0.1686
	14	82.48	0.1999	22.1	4200.82	4283.3	0.1999	15.9	7200	0.1713		14	131.56	0.2072	23.1	7000	7131.56	0.1914	12.3	7200	0.1716
	15	112.52	0.2102	23	2999.48	3112	0.2102	18.2	7200	0.1799		15	159.84	0.2233	25.8	3000.88	3160.72	0.2233	20.2	7200	0.1883
A15Pri	5	0	0.0917	0	0	0	0.0917	0	0.02	0.0917	A20Pri	5	0	0.0942	0	0.02	0.02	0.0942	0	0.01	0.0942
	6	0.05	0.1194	4.3	0.5	0.55	0.1143	0	0.02	0.1143		6	0.03	0.1171	0	0.08	0.11	0.1171	0	0.05	0.1171
	7	0.09	0.1675	17.7	6.86	6.95	0.1379	0	0.67	0.1379		7	0.11	0.1646	14.1	7.43	7.54	0.1414	0	0.89	0.1414
	8	0.33	0.1775	21.9	21.67	220	0.1386	0	2.21	0.1386		8	0.33	0.2119	32.9	22.53	22.86	0.1461	2.66	0.92	0.1422
	9	0.91	0.2095	33.3	139.35	140.26	0.1397	0	12.68	0.1397		9	2.79	0.1905	24.9	118.05	120.84	0.1431	0	9.08	0.1431
	10	2.62	0.2302	39.5	288.9	291.52	0.1439	3.2	185.83	0.1393		10	5.63	0.2126	32.9	279.21	284.84	0.1474	3.2	119.22	0.1427
	11	4.77	0.2763	33.4	921.07	925.84	0.1841	0	322.87	0.1841		11	8.08	0.2824	33.1	793.28	801.36	0.1889	0	569.43	0.1889
	12	10.01	0.3348	37.9	6557.32	6567.33	0.2214	1.7	7200	0.2214		12	8.44	0.3469	38.6	6753.36	6761.8	0.2265	1.5	7200	0.2265
A15M887	5	0.02	0.1118	0	0.02	0.04	0.1118	0	0.01	0.1118	A20M887	5	0	0.1147	0	0.01	0.01	0.1147	0	0.02	0.1147
	6	0.03	0.1286	3.2	1.2	1.23	0.1245	0	0.47	0.1245		6	0.05	0.1527	16.6	0.95	10	0.1274	0	0.75	0.1274
	7	0.11	0.1585	7.5	3.48	3.59	0.1466	0	1.86	0.1466		7	0.11	0.1689	11.1	4.37	4.48	0.15	0	1.64	0.15
	8	0.3	0.2081	8.5	15.15	15.45	0.1951	2.3	11.61	0.1905		8	0.53	0.2284	14.5	16.66	17.19	0.1998	2.2	11.89	0.1953
	9	0.95	0.2257	10.5	35.96	36.91	0.2035	0.8	9.7	0.202		9	1.34	0.2493	17.1	34.54	35.88	0.2067	0	9.02	0.2067
	10	4.01	0.2637	16.4	368.27	372.28	0.2237	1.4	128.84	0.2205		10	6.13	0.2705	16.7	176.44	182.57	0.2253	0	120.14	0.2253
	11	5.85	0.284	19.1	796.55	802.4	0.2297	0	1881.22	0.2297		11	14.74	0.2815	16.5	1371.93	1386.67	0.2352	0	2079.15	0.2352
	12	15.69	0.2966	30.4	6332.15	6347.84	0.2345	2.4	7200	0.2345		12	23.9	0.3	29.6	3501.27	3525.17	0.24	2.4	7200	0.24
	13	24.52	0.3199	30.1	7000	7024.52	0.2584	3.9	7200	0.2527		13	35.87	0.3124	26.8	4999.56	5035.43	0.2593	4.2	7200	0.2639
	14	68.05	0.3249	29.8	7000	7068.05	0.3022	13.6	7200	0.2714		14	57.53	0.3363	30.6	5601.28	5658.81	0.2955	11.9	7200	0.2728
	15	68.58	0.3509	29.7	3000.63	3069.21	0.3509	22.3	7200	0.305		15	64.55	0.3628	30.3	4500.37	4564.92	0.3107	9.9	7200	0.311

Table 6.3: Computational results of the heuristics for data from the phylogenetics application (continuation).

		FP heuristic			LB heuristic				Path-edges$^+$	
M	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
A100M391	5	0	0.0748	0	0.02	0.02	0.0748	0	0	0.0748
	6	0.03	0.0963	8.4	0.73	0.76	0.0882	0	0.08	0.0882
	7	0.14	0.1086	10.9	5.43	5.57	0.0968	0	1.65	0.0968
	8	0.41	0.123	15.1	10.22	10.63	0.1044	0	9.64	0.1044
	9	1.09	0.1721	16.9	53.45	54.54	0.143	0	13.6	0.143
	10	5.84	0.185	11.3	432.61	438.45	0.164	0	279.43	0.164
	11	7.07	0.2254	9	891.78	898.85	0.2058	0.3	3594	0.2052
	12	12.29	0.2456	19.7	7000	7012.29	0.2106	1.8	7200	0.2097
	13	42.28	0.2792	23.4	7000	7042.28	0.2304	1.9	7200	0.2358
	14	151.32	0.2743	20.9	7000	7151.32	0.255	9.2	7200	0.2369
	15	526.22	0.3075	26.2	2999.68	3525.9	0.3075	19.6	7200	0.2616
A100Pri	5	0.02	0.1338	0	0.02	0.04	0.1338	0	0	0.1338
	6	0.03	0.1765	8.3	0.87	0.9	0.1618	0	0.05	0.1618
	7	0.2	0.2384	17.6	6.93	7.13	0.1966	0	2.18	0.1966
	8	0.33	0.2907	31.5	11.9	12.23	0.2048	2.8	1.93	0.1991
	9	1.51	0.3454	41.1	94.24	95.75	0.2053	0.9	9.77	0.2035
	10	2.82	0.3642	46	347.08	349.9	0.2034	3.3	94.63	0.1966
	11	4.23	0.3741	29	1435.08	1439.31	0.2655	0	1008.31	0.2655
	12	18.24	0.4084	28.5	4991.84	5010.08	0.3166	2.2	7200	0.3166
A100M887	5	0.02	0.1601	0	0	0.02	0.1601	0	0	0.1601
	6	0.03	0.1932	9.1	0.76	0.79	0.1756	0	0.08	0.1756
	7	0.08	0.2287	9.7	6.4	6.48	0.2065	0	2.45	0.2065
	8	0.59	0.319	14.9	11.56	12.15	0.2714	0	2.75	0.2714
	9	1.15	0.3262	13.1	35.52	36.67	0.2833	0	8.08	0.2833
	10	2.53	0.3611	15.2	138.28	140.81	0.3064	0	144.63	0.3064
	11	5.66	0.407	21.9	760.88	766.54	0.3177	0	3532.92	0.3177
	12	18.02	0.4312	33.3	6893.39	6911.41	0.3265	2.6	7200	0.3265
	13	71.48	0.4541	31.5	7000	7071.48	0.35785	4.2	7200	0.3604
	14	224.58	0.465	31.6	7000	7224.58	0.4028	13.3	7200	0.3759
	15	484.61	0.5172	32.8	4500.5	4985.11	0.4451	14.6	7200	0.446

Table 6.4: Computational results of the heuristics for data from the networking application.

		FP heuristic			LB heuristic				Path-edges$^+$				FP heuristic			LB heuristic				Path-edges$^+$	
M	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**	**M**	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
S7	5	0	0.4043	0	0.08	0.08	0.4043	0	0.02	0.4043	A10S7	5	0	0.4266	0	0.02	0.02	0.4266	0	0.02	0.4266
	6	0.03	0.6264	30.6	0.89	0.92	0.4348	0	0.05	0.4348		6	0.03	0.4579	0	0.05	0.08	0.4579	0	0.03	0.4579
	7	0.08	0.5963	15.2	1.29	1.37	0.5055	0	0.05	0.5055		7	0.09	0.7331	27.3	1.11	1.2	0.5328	0	0.13	0.5328
S15	5	0	0.1893	0	0.06	0.06	0.1893	0	0	0.1893	A10S15	5	0	0.1995	0	0.02	0.02	0.1995	0	0	0.1995
	6	0.02	0.2756	0	0.17	0.19	0.2756	0	0.02	0.2756		6	0.05	0.3424	15.1	0.84	0.89	0.2906	0	0.52	0.2906
	7	0.13	0.3526	14.6	2.09	2.22	0.3013	0	0.11	0.3013		7	0.09	0.3175	0.03	1.14	1.23	0.3175	0.03	0.69	0.3173
	8	0.87	0.5103	26	12.51	13.38	0.3778	0	7.29	0.3778		8	0.52	0.5282	24.5	29.77	30.29	0.4189	4.8	5.74	0.3987
	9	0.94	0.6896	43	45.4	46.34	0.3932	0	8.74	0.3932		9	1.03	0.6521	36.4	56.61	57.64	0.415	0	9.86	0.415
	10	2.21	0.7145	40	333.79	336	0.4286	0	119.56	0.4286		10	3.15	0.6368	29	205.55	208.7	0.4528	0	170.62	0.4528
	11	16.64	0.9018	30.6	700.35	716.99	0.626	0	1317.44	0.626		11	5.96	0.9739	32.1	1238.49	1244.45	0.6612	0	895.27	0.6612
	12	34.89	1.2111	41.4	4287.08	4321.97	0.722	1.7	7200	0.722		12	17.35	1.2981	42.3	5530.32	5547.67	0.7634	1.3	7200	0.7634
	13	33.65	1.4488	49.3	5999.47	6033.12	0.7474	1.7	7200	0.7474		13	21.64	1.3191	41.5	6400.75	6422.39	0.8002	2.4	7200	0.7891
	14	324.75	1.6359	55.5	7000	7324.75	0.8239	8.8	7200	0.7831		14	98.69	1.5418	40.4	5600.16	5698.85	1.2005	28.5	7200	0.8276
	15	122.24	1.6105	37.8	7000	7122.24	1.1941	24.2	7200	0.9813		15	116.7	1.7335	49	2800.59	2917.29	1.5422	36.1	7200	1.1003
S20	5	0	0.299	0	0.03	0.03	0.299	0	0	0.299	A10S20	5	0	0.3152	0	0.02	0.02	0.3152	0	0	0.3152
	6	0.02	0.4003	15.2	0.64	0.66	0.3395	0	0.01	0.3395		6	0.03	0.4194	14.6	1.2	1.23	0.3583	0	0.06	0.3583
	7	0.08	0.3751	0	5.51	5.59	0.3751	0	0.3	0.3751		7	0.11	0.4599	14.1	2.33	2.44	0.3951	0	0.86	0.3951
	8	0.44	0.5275	13.5	16.72	17.16	0.4564	0	7.41	0.4564		8	0.31	0.4934	2.6	22.57	22.88	0.4814	0.2	11.54	0.4807
	9	0.78	0.6188	16.4	78.05	78.83	0.5174	0	15.85	0.5174		9	0.73	0.6208	12.2	49.83	50.56	0.546	0.1	12.9	0.5454
	10	4.98	0.7564	24.2	246.53	251.51	0.5736	0	172.71	0.5736		10	4.12	0.7748	21.9	736.18	740.3	0.6048	0	206.08	0.6048
	11	5.12	0.9646	36.9	1194.12	1199.24	0.6193	1.7	2039.97	0.609		11	6.18	0.9102	29.3	1054.94	1061.12	0.6435	0.1	3055.13	0.6429
	12	15.2	0.9903	38.3	4745.4	4760.6	0.6347	3	7200	0.6245		12	42.26	1.0354	37.9	6938.02	6980.28	0.6603	2	7200	0.6596
	13	15.05	0.8735	18.1	3999.4	4014.45	0.7515	3.5	7200	0.7413		13	30.2	1.1687	35.9	7000	7030.2	0.7811	3	7200	0.7821
	14	531.09	1.2439	38.1	7000	7531.09	0.8781	9.7	7200	0.7967		14	86.55	1.3491	40	7000	7086.55	0.9182	8.1	7200	0.883
	15	199.66	1.3002	33.3	7000	7199.66	1.1071	15.8	7200	0.9746		15	97.83	1.5686	42.1	6000.47	6098.3	1.2649	22.8	7200	0.9413

Table 6.5: Computational results of the heuristics for data from the networking application (continuation).

		FP heuristic			LB heuristic				Path-edges$^+$				FP heuristic			LB heuristic				Path-edges$^+$	
M	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**	**M**	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
A15S7	5	0.02	0.4377	0	0.01	0.03	0.4377	0	0	0.4377	A20S7	5	0	0.4488	0	0.01	0.01	0.4488	0	0.02	0.4488
	6	0.02	0.6525	28.1	0.81	0.83	0.4694	0	0.05	0.4694		6	0.05	0.6951	30.8	0.86	0.91	0.481	0	0.05	0.481
	7	0.09	0.752	27.3	1.48	1.57	0.5465	0	0.09	0.5465		7	0.11	0.584	4.1	2.26	2.37	0.5635	0.6	0.14	0.5602
A15S15	5	0	0.2046	0	0.02	0.02	0.2046	0	0.01	0.2046	A20S15	5	0	0.2098	0	0.02	0.02	0.2098	0	0.01	0.2098
	6	0.05	0.3501	14.9	0.87	0.92	0.2981	0	0.23	0.2981		6	0.03	0.3604	15.2	1.25	1.28	0.3055	0	0.53	0.3055
	7	0.2	0.379	14.2	3.01	3.21	0.3252	0	0.72	0.3252		7	0.09	0.4576	27.1	4.09	4.18	0.3334	0	0.87	0.3334
	8	0.55	0.6086	32.8	29.64	30.19	0.4102	0.2	3.79	0.4092		8	0.53	0.6021	30.3	25.76	26.29	0.4197	0	7.71	0.4197
	9	1.54	0.671	36.5	69.7	71.24	0.426	0	10.37	0.426		9	0.62	0.7094	38.41	52.9	53.52	0.4369	0	8.52	0.4369
	10	2.2	0.6635	30	384.48	386.68	0.4646	0	119.43	0.4646		10	2.82	0.6806	30	305.56	308.38	0.4766	0	158.62	0.4766
	11	5.13	0.9557	29	587.64	592.77	0.6788	0	902.2	0.6788		11	2.61	0.8724	20.2	1594.18	1596.79	0.6964	0	909.73	0.6964
	12	18.38	1.0718	28.6	4821.25	4839.63	0.7841	1.5	7200	0.7841		12	22.85	1.3066	40.2	6245.47	6268.32	0.8048	1.7	7200	0.8048
	13	35.65	1.3749	42.4	7000	7035.65	0.8528	4.1	7200	0.8105		13	12.87	1.116	27.4	7000	7012.87	0.8315	1.7	7200	0.8307
	14	57.74	1.5761	50.3	7000	7057.74	1.0515	20.1	7200	0.9138		14	201.43	1.6262	51	7000	7201.43	0.9661	12.8	7200	1.0304
	15	154.92	1.8676	51.6	6000.55	6155.47	1.5214	34.4	7200	1.0738		15	77.09	2.0582	55	7000	7077.09	1.5668	32.6	7200	1.2272
A15S20	5	0	0.3233	0	0.01	0.01	0.3233	0	0.01	0.3233	A20S20	5	0.02	0.3315	0	0	0.02	0.3315	0	0	0.3315
	6	0.03	0.368	0.1	1.7	1.73	0.3678	0	0.16	0.3678		6	0.03	0.4409	14.5	1.61	1.64	0.3772	0	0.08	0.3772
	7	0.08	0.4688	13.6	2.53	2.61	0.4054	0.1	0.75	0.405		7	0.09	0.4831	14.1	4.03	4.12	0.415	0	0.87	0.415
	8	0.41	0.5697	13.5	9.95	10.36	0.4928	0	7.96	0.4928		8	0.47	0.5883	14.2	23.28	23.75	0.5049	0	13	0.5049
	9	1.39	0.7036	20.5	67.17	68.56	0.5593	0	11.75	0.5593		9	1.44	0.7646	25	66.44	67.88	0.5733	0	11.45	0.5733
	10	6.37	0.808	23.2	217.65	224.02	0.6215	0.2	179.48	0.6204		10	21.89	0.8726	27.1	603.35	625.24	0.6359	0	178.03	0.6359
	11	7.21	1.0445	36.8	738.29	745.5	0.6608	0.2	1804.07	0.6598		11	9.52	0.9222	26.6	968.82	978.34	0.6871	1.5	944.07	0.6768
	12	28.14	1.0764	38.8	6794.65	6822.79	0.6771	2	7200	0.6771		12	7.89	1.0314	34.7	6672.18	6680.07	0.6947	2	7200	0.6947
	13	31.33	1.1518	33.3	5999.35	6030.68	0.8122	3.6	7200	0.7917		13	28.89	1.1722	32.9	4999.51	5028.4	0.835	4	7200	0.812
	14	42.68	1.183	29.9	7000	7042.68	0.9105	6.8	7200	0.8884		14	62.65	1.3819	38.6	7000	7062.65	0.9934	9	7200	0.8989
	15	417.66	1.6028	42	6001.44	6419.1	1.4244	29.1	7200	1.01		15	126.83	1.4517	34.6	6002.27	6129.1	1.2953	20.3	7200	0.9988

Table 6.6: Computational results of the heuristics for data from the networking application (continuation).

		FP heuristic			LB heuristic				Path-edges$^+$				FP heuristic			LB heuristic				Path-edges$^+$	
M	$\lvert V_t\rvert$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**	**M**	$\lvert V_t\rvert$	**T**	**W**	**GAP**	**T**	**TT**	**W**	**GAP**	**T**	**W**
A100S7	5	0.02	0.6269	0	0.02	0.04	0.6269	0	0	0.6269	SS7	5	0	0.3961	0	0.03	0.03	0.3961	0	0	0.3961
	6	0.03	0.7048	5.5	0.62	0.65	0.6662	0	0.03	0.6662		6	0.03	0.5997	29.4	0.8	0.83	0.4237	0	0.05	0.4237
	7	0.11	1.0764	27.6	4.01	4.12	0.7791	0	0.16	0.7791		7	0.09	0.6033	17.4	3.43	3.52	0.5048	1.3	0.09	0.4984
A100S15	5	0.02	0.2916	0	0.02	0.04	0.2916	0	0.01	0.2916	SS15	5	0	0.2081	0	0.02	0.02	0.2081	0	0.02	0.2081
	6	0.03	0.494	13.9	1.61	1.64	0.4255	0	0.42	0.4255		6	0.03	0.2951	0.2	1.28	1.31	0.2945	0	0.14	0.2945
	7	0.14	0.7029	34.4	4.42	4.56	0.4609	0	0.61	0.4609		7	0.14	0.3211	0.2	1.44	1.58	0.3211	0.2	0.37	0.3205
	8	0.37	0.7342	20	19.72	20.09	0.5937	1.1	8.56	0.5873		8	0.47	0.5617	28.8	33.34	33.81	0.3997	0	4.51	0.3997
	9	1	0.9982	38.7	68.25	69.25	0.6116	0	9.63	0.6116		9	1.67	0.6899	39.79	45.3	46.99	0.4154	0	8.5	0.4154
	10	4.13	1.0909	38.7	405.85	409.98	0.6685	0	142.88	0.6685		10	2.08	0.6367	29.1	238.66	240.74	0.4517	0	103.65	0.4517
	11	5.16	1.5028	34.9	1037.53	1042.69	1.0214	4.2	700.94	0.9782		11	14.74	0.9298	29.9	1031.54	1046.28	0.6518	0	762.98	0.6518
	12	81.42	1.8071	40.9	4423.61	4505.03	1.1363	2	7200	1.1363		12	12.62	1.2374	40.7	5610.45	5623.07	0.7478	1.5	7200	0.7478
	13	72.93	2.0986	47.1	7000	7072.93	1.2298	4.4	7200	1.186		–	–	–	–	–	–	–	–	–	–
	14	142.18	2.572	57.8	7000	7142.18	1.4485	18	7200	1.3554		–	–	–	–	–	–	–	–	–	–
	15	89.61	2.557	50.4	6566.83	6656.44	2.1839	33.2	7200	1.6641		–	–	–	–	–	–	–	–	–	–
A100S20	5	0	0.4611	0	0.02	0.02	0.4611	0	0.02	0.4611	SS20	5	0	0.3026	0	0.02	0.02	0.3026	0	0.01	0.3026
	6	0.03	0.5295	0.3	1.08	1.11	0.5281	0	0.28	0.5281		6	0.05	0.4062	15.6	1.11	1.16	0.3434	0.2	0.19	0.3427
	7	0.11	0.6654	13.7	4.37	4.48	0.5742	0	0.94	0.5742		7	0.09	0.5029	24.7	3.93	4.02	0.3789	0	0.55	0.3789
	8	0.47	0.9234	24.3	12.28	12.75	0.7062	1	8.53	0.699		8	0.7	0.5223	11.7	46.74	47.44	0.4614	0.1	7.67	0.461
	9	0.94	1.0606	24.8	51.17	52.11	0.7972	0	12.36	0.7972		9	0.73	0.5625	7.2	41.39	42.12	0.5227	0.1	12.21	0.522
	10	2.92	1.2049	26.5	287.18	290.1	0.8927	0.8	137.65	0.8853		10	4.07	0.7328	21.1	455.65	459.72	0.5784	0	280.8	0.5784
	11	7.25	1.3079	27.5	585.28	592.53	0.9477	0	1042.36	0.9477		11	16.6	0.9079	32.4	563.5	580.1	0.6136	0	1108.13	0.6136
	12	23.56	1.321	30.3	7000	7023.56	0.9756	2	7200	0.9756		12	34.94	0.8805	30.4	6445.89	6480.83	0.6287	2.1	7200	0.6287
	13	25.37	1.6664	35.9	7000	7025.37	1.1514	3.2	7200	1.1311		–	–	–	–	–	–	–	–	–	–
	14	46.4	1.9935	41.7	7000	7046.4	1.4268	11.6	7200	1.2801		–	–	–	–	–	–	–	–	–	–
	15	146.13	2.2779	44	6002.14	6148.27	1.4971	9.3	7200	1.4284		–	–	–	–	–	–	–	–	–	–

The FP heuristic obtains the same solution as the one obtained by the Path-edges$^+$ for all instances with $n = 5$ terminal nodes and for instances obtained from matrices M391, A20Pri, S15 and A10S7 with $n = 6$.

The LB heuristic obtains the same solution as the one obtained by the Path-edges$^+$ in 183 of the 294 instances (approximately, 62%) and obtains a better solution in 14 of the 294 instances (approximately, 5%).

For instances with the same number of terminal nodes, Tables 6.7 and 6.8 display the average time and the average GAP, respectively, and their corresponding standard deviation values for the FP heuristic and the LB heuristic. Tables 6.7 also displays the average times and corresponding standard deviation values of the Path-edges$^+$ formulation to facilitate the comparison.

The average time of the FP heuristic varies from 0 to 202.12 seconds and the average GAP varies from 0 to 36.29%. The average time of the LB heuristic varies from 0.02 to 6720.11 seconds and the average GAP varies from 0 to 20.84%. It is worth nothing that for $n < 12$ the average GAP of the LB heuristic is less than 1%. The FP heuristic finds a feasible solution very quickly and the LB heuristic finds good solutions.

Figure 6.5 displays the average GAP of the FP and of the LB heuristic. As we can see the LB heuristic significantly improves the feasible solution obtained by the FP heuristic.

Table 6.7: Average and standard deviation (SD) values for the computational time of the FP heuristic, LB heuristic and Path-edges$^+$ formulations.

n		FP heuristic	LB heuristic	Path-edges$^+$
5	average	0.01	0.02	0.01
	SD	0.01	0.02	0.01
6	average	0.1	0.91	0.18
	SD	0.34	0.45	0.22
7	average	0.11	3.92	0.92
	SD	0.04	1.82	0.72
8	average	0.45	19.89	8.49
	SD	0.14	8.43	5.28
9	average	1.22	67.79	11.06
	SD	0.59	28.09	2.07
10	average	4.85	329.55	184
	SD	4.1	128.06	93.44
11	average	8.38	1045.8	1549.73
	SD	5.2	320.01	920.16
12	average	21.36	5915.44	–
	SD	15.15	1032.85	–
13	average	43.65	6261.95	–
	SD	39.6	1002.03	–
14	average	145.18	6720.11	–
	SD	123.38	732.14	–
15	average	202.12	5044.09	–
	SD	149.83	1573.13	–

Table 6.8: Average and standard deviation (SD) values for the GAP of the FP heuristic and the LB heuristic.

n		FP heuristic	LB heuristic
5	average	0	0
	SD	0	0
6	average	10.57	0.01
	SD	9.74	0.03
7	average	13.99	0.07
	SD	9.21	0.24
8	average	18.23	0.71
	SD	8.32	1.21
9	average	23.44	0.31
	SD	11.87	0.86
10	average	25	0.68
	SD	11.35	1.19
11	average	25.67	0.35
	SD	9.13	0.9
12	average	31.29	1.8
	SD	10.08	0.03
13	average	30.79	3.2
	SD	11.09	1.9
14	average	36.29	12.65
	SD	12.55	6.98
15	average	36.13	20.84
	SD	12.27	10

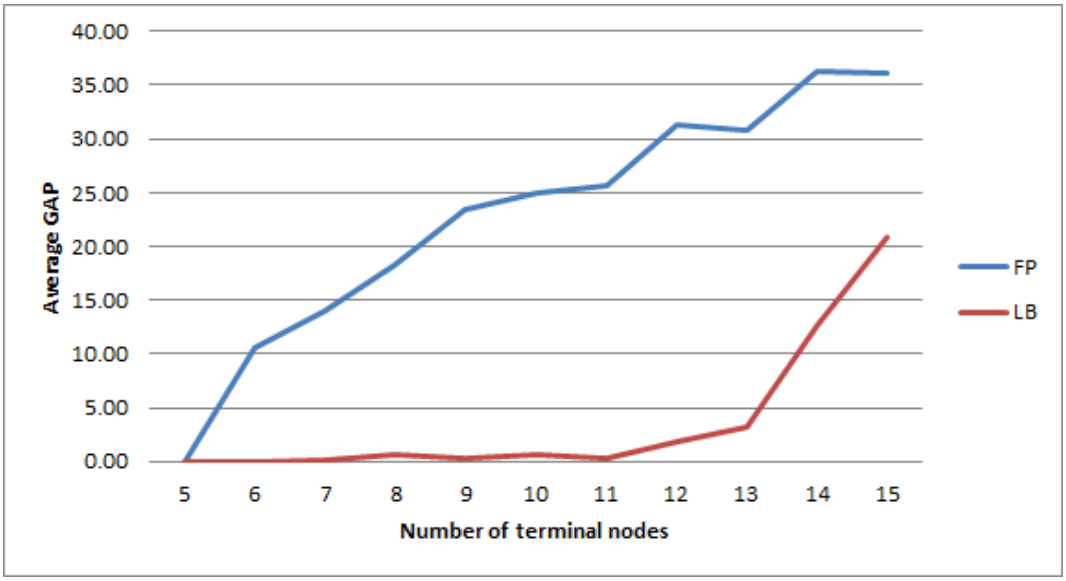


Figure 6.5: Average Gap of the FP and of the LB heuristic

Chapter 7

Robust Approach

As we saw in Chapter 3 and Chapter 4 the input data (the distance matrix), from the two application areas studied, used in the MWTR problem, are obtained through methods which involve imprecisions and therefore these distances are uncertain. To handle this uncertainty we used two robust approaches, one to control the maximum number of deviations and the other to reduce the risk of high cost.

In this chapter, we present the two robust approaches and the three formulations [1], Robust-Deviation-Dual formulation, Robust-Deviation formulation and Robust-CVaR formulation, that we derived from these two approaches for the MWTR problem using the Path-edges$^+$ formulation. We also present the computational results obtained by running the three robust formulations using the data instances presented in Section 3.2 and Section 4.2.

A classical approach in mathematical programming is to develop a model that assumes that the input data is precisely known and equal to some nominal values. However the data used in real-world optimization problems are mostly uncertain, due, for example, to measurement or estimation errors. Under data uncertainty as the data may take different values than the nominal values, several constraints may be violated and the optimal solution found using nominal data may no longer be optimal or even feasible. When assuming data

[1]The three formulations were developed in collaboration with CNRS researcher Michael Poss.

uncertainty, the usual deterministic solution may become completely meaningless from a practical viewpoint. Therefore, it is important to derive solution approaches that are immune to data uncertainty, that is, that are *robust* [10, 17].

Two main approaches used in optimization to deal with data uncertainty are stochastic and robust optimization. In stochastic optimization the uncertain data are random variables whose probability distributions is known or can be estimated. On the other hand, in robust optimization the uncertain data lay in a predefined uncertain set and their probability distributions is unknown [72].

Recently efforts have been done to connect robust optimization and stochastic optimization. In distributionally robust optimization, it is assumed that the distribution of the uncertainty data is partially-characterized in the sense that certain distributional properties, such as some data moments (mean, variance, ...), are known [71].

Soyster [130] was the first to propose a linear optimization model to construct a solution that is feasible for all data belonging to a convex set. Other significant works to the development of robust optimization and which use Soyster's general approach are Ben-Tal and Nemirovski [11, 12, 13], Bertsimas and Sim [17], El-Ghaoui and Lebret [55] and El-Ghaoui et al. [56].

Consider a deterministic nominal linear optimization problem of the form

$$\min_x \{c^\intercal x : Ax \leq b\} \tag{7.1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.

In many applications some or all of the entries in (c, A, b) are uncertain. In robust optimization the data (c, A, b) varies in a given uncertainty set, $\mathcal{U}$, that is, $(c, A, b) \in \mathcal{U}$ [10].

The collection of all linear optimization problems (7.1) such that $(c, A, b) \in \mathcal{U}$,

$$\left\{ \min_x \{c^\intercal x : Ax \leq b\} \right\}_{(c,A,b)\in\mathcal{U}} \tag{7.2}$$

is called an *uncertain linear optimization problem.*

A solution to an uncertain linear optimization problem is said to be robust if it has both feasibility robustness and optimality robustness. This two concepts are define in the following.

Definition 7.1. ([10]) A vector $x \in \mathbb{R}^n$ is a *robust feasible solution* of an uncertain linear optimization problem (7.2), if it satisfies all the realizations of the constraints from the uncertainty set, that is, $Ax \leq b,\ \forall (c, A, b) \in \mathcal{U}$.

Since an uncertain linear optimization problem is a collection of instances of linear optimization problems, we must define how to determine the value of the objective function at a robust feasible solution. The techniques of robust optimization are based on the worst case analysis and therefore the objective function value at a robust feasible solution is determined as follows.

Definition 7.2. ([10]) Let x be a robust feasible solution of an uncertain linear optimization problem (7.2). The *robust value* $\widehat{z}(x)$ of the objective function of the uncertain linear optimization problem at x is the largest value of the objective function $z(x) = c^\intercal x$ over all realizations of the data from the uncertainty set, that is

$$\widehat{z}(x) = \sup_{(c,A,b)\in\mathcal{U}} c^\intercal x.$$

Definition 7.3. ([10]) The *robust counterpart* of an uncertain linear optimization problem is the optimization problem:

$$\min_x \left\{ \sup_{(c,A,b)\in\mathcal{U}} c^\intercal x : Ax \leq b,\ \forall (c, A, b) \in \mathcal{U} \right\} \tag{7.3}$$

A *robust optimal solution* of an uncertain linear optimization problem is an optimal solution of the robust counterpart, that is, the robust feasible solution that minimizes the robust value $\widehat{z}(x)$. In other words, the robust optimal solution is the "best uncertainty-immunized" solution of the uncertain linear optimization problem [10].

We can distinguish two types of uncertainty: uncertainty on the feasibility of the solutions, that is, uncertainty on the matrix A and the vector b, and uncertainty on the objective coefficients, that is, uncertainty on the vector c [66].

Since we are interested in developing a robust formulation of the Path-edges$^+$ formulation and in this formulation the uncertain data (the distances) only appear in the objective coefficients, in what follows, we focus only on the uncertainty on the objective value, that is, on the cost vector c.

In order to account for the uncertainty of the cost coefficients we consider two alternative approaches: (i) control the maximum number of deviations from the nominal cost values [17] and (ii) control the conditional expectation of the total cost [108] in which the objective is to reduce the risk of high cost. In the first case, we will use the approach of Bertsimas and Sim [16, 17]. In the second case, we will use the approach of Rochafellar and Uryasev [114].

The first approach assumes that cost coefficients belong to a predefined uncertainty set and that only a fixed number of coefficients can change value. While the second approach uses the several values of the cost coefficients obtained in practice.

In both cases, the problem turns to be searching for a feasible solution that minimizes the worst solution cost.

7.1 Control the maximum number of deviations

Bertsimas and Sim [16] present a technique for polyhedral uncertainty that leads to linear robust counterparts while controlling the level of conservatism of the solution.

Consider the following deterministic nominal linear optimization problem:

$$\min_{x \in \mathcal{X}} \; z := c^\intercal x = \sum_{i=1}^{n} c_i \; x_i \tag{7.4}$$

with $\mathcal{X} \subseteq \mathbb{R}^n_+$ the set of feasible solutions and $c = (c_i) \in \mathbb{R}^n$ a cost vector.

The set of feasible solutions is not subject to uncertainty and the cost coefficients are uncertain, with unknown probabilistic distribution. However, it is assumed that the uncertain cost coefficients $\tilde{c}_i$, $i \in N = \{1, \ldots, n\}$ take values in $[c_i, c_i + \delta_i]$, where $\delta_i \geq 0$, $i \in N$ represents the deviation from the nominal cost value c_i.

The aim is to find a solution $x \in \mathcal{X}$ that minimizes the maximum cost $z = \sum_{i=1}^{n} \tilde{c}_i \; x_i$ such that at most Γ of the coefficients $\tilde{c}_i$ are allowed to change [16].

Taking the set $S = \{i \in N : \delta_i > 0\}$ of the indices of the cost coefficients that may change from their nominal value and the uncertainty set

$$\mathcal{U} := \{\tilde{c} \in \mathbb{R}^n : \tilde{c}_i \in [c_i, c_i + \delta_i],\ \delta_i \geq 0\}$$

the problem is to

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & \max \tilde{c}^{\intercal} \; x \\ \text{subject to} \quad & \tilde{c} \in \mathcal{A} = \{\tilde{c} \in \mathcal{U} : \text{ at most } \Gamma \text{ of the } \tilde{c}_i \ (i \in S) \text{ can change}\} \end{aligned}$$

More precisely

$$\min_{x \in \mathcal{X}} \left(\sum_{i=1}^{n} \tilde{c}_i \; x_i + \max_{\{J : J \subseteq S, |J| \leq \Gamma\}} \sum_{j \in J} \delta_j \; x_j \right). \tag{7.5}$$

which is the robust counterpart of problem (7.4).

If $\Gamma = 0$ then the cost deviations are ignored and if $\Gamma = |S|$ then all possible cost deviations are considered.

Without loss of generality assume that the indices are ordered such that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$. For notational convenience define $\delta_{n+1} = 0$.

The robust counterpart (7.5) can be rewritten as:

$$\begin{gathered} \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{n} \tilde{c}_i \; x_i + \max \sum_{j \in N} \delta_j \; x_j \; u_j \right) \\ s.t. \quad 0 \leq u_j \leq 1, \quad \forall j \in N \\ \sum_{j \in N} u_j \leq \Gamma \end{gathered}$$

By dualizing the inner maximization problem and joining the two minimization problems,

we obtain [16]:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \tilde{c}_i \; x_i + \Gamma\theta + \sum_{j\in N} \nu_i \\ s.t. \quad & \nu_j + \theta \geq \delta_j \; x_j, \quad j \in N \\ & \nu_j, \; \theta \geq 0, \\ & x \in \mathcal{X} \end{aligned} \tag{7.6}$$

For the case when the decisions variables $x_i, \; i \in N$ are binary ($\mathcal{X} \subseteq \{0,1\}^n$), Bertsimas and Sim [16] proved that the robust counterpart (7.5) can be solved by solving at most $n+1$ nominal problems:

Theorem 7.1. *Problem* (7.5) *can be solved by solving the* $n+1$ *nominal problems:*

$$\min_{\ell=1,\ldots,n+1} G^{\ell}$$

where for $\ell = 1, \ldots, n+1$*:*

$$G^{\ell} = \Gamma \; \delta_{\ell} + \min_{x\in\mathcal{X}} \left(\sum_{i=1}^{n} \tilde{c}_i \; x_i + \sum_{j=1}^{\ell} (\delta_j - \delta_{\ell}) \; x_j \right) \tag{7.7}$$

The proof of this theorem can be found in [16].

In what follows we extend Bertsimas and Sim's [16] Theorem 7.1 to the problems with the special structure given by:

$$\begin{aligned} z = \quad \min \quad & \sum_{i=1}^{I} c_i \; \sum_{j=1}^{J} a_j \; x_{ij} \\ s.t. \quad & x \in \mathcal{X}. \end{aligned} \tag{7.8}$$

where $c \in \mathbb{R}^I, a \in \mathbb{R}^J$ are cost vectors and $\mathcal{X} \subset \{0,1\}^{I\times J}$ is a set of binary vectors such that any $x \in \mathcal{X}$ satisfies

$$\sum_{j=1}^{J} x_{ij} = 1, \quad i = 1, \ldots, I, \tag{7.9}$$

We consider that, $\forall i \in I, j \in J$, the cost of variable x_{ij} is equal to $\widetilde{c}_i \; a_j$ where a_j is fixed, $\widetilde{c}_i$ is an uncertain parameter that takes value in $[c_i, c_i + \delta_i]$ and $\delta_i \geq 0$.

This structure of cost function combined with constraint (7.9) arises in many applications where one must choose one among a set of different options (facility location, network design, ...).

We are interested in the class of problems that minimizes the maximum cost z such that at most Γ of the coefficients of the objective function are allowed to change. The following optimization problem is obtained:

$$\begin{aligned} z = \quad & \min \sum_{i=1}^{I} c_i \sum_{j=1}^{J} a_j x_{ij} + \max_{\{S|S\subset\{1,\ldots,I\},|S|\leq\ \Gamma\}} \sum_{i=1}^{I} \delta_i \sum_{j\in J} a_j x_{ij} \\ & s.t. \qquad x \in \mathcal{X}. \end{aligned} \tag{7.10}$$

We show next that Problem (7.10) can be solved by solving at most $I \times J + 1$ nominal problems. Let $k = (i, j)$, for each $k = 1, \ldots, K = I \times J$, be a new indice and $\Delta_k = \delta_i\ a_j$ such that $\Delta_1 \geq \Delta_2 \geq \ldots \geq \Delta_K \geq \Delta_{K+1} = 0$.

Theorem 7.2. *Problem* (7.10) *can be solved by solving at most* $K + 1$ *nominal problems:*

$$z^* = \min_{\ell=1,\ldots,K+1} z^\ell,$$

where for $\ell = 1, \ldots, K + 1$*:*

$$\begin{aligned} z^\ell = \Gamma\ \Delta_\ell + \quad & \min \sum_{i=1}^{I} c_i \sum_{j=1}^{J} a_j\ x_{ij} + \sum_{k=1}^{\ell} (\Delta_k - \Delta_\ell)\ x_k \\ & s.t. \quad x \in \mathcal{X}. \end{aligned} \tag{7.11}$$

Proof. This proof is based on the proof of Theorem 7.1 presented in [16].

Expressing the inner maximization problem of (7.10) as a linear program, we obtain:

$$z = \min_{x\in\mathcal{X}} \left(\sum_{i=1}^{I} c_i\ \sum_{j=1}^{J} a_j\ x_{ij} \quad + \quad \max \sum_{i=1}^{I} \delta_i\ u_i \sum_{j=1}^{J} a_j\ x_{ij} \right)$$

$$s.t. \quad 0 \leq u_i \leq 1, \quad \forall i \in I \tag{7.12}$$

$$\sum_{i=1}^{I} u_i \leq \Gamma \tag{7.13}$$

where integrality requirements on variables u are omitted because the constraints of the inner maximization problem are defined by totally unimodular matrix.

By dualizing the inner maximization problem and joining the two minimization problems, we obtain

$$\begin{aligned} z = \min & \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \quad + \quad \Gamma \ \theta \ + \ \sum_{i=1}^{I} v_i \\ s.t. \quad & x \in \mathcal{X} \\ & v_i + \theta \ \geq \ \delta_i \ \left(\sum_{j=1}^{J} a_j \ x_{ij} \right), \quad \forall i \in I \\ & \theta, \ v_i \ \geq \ 0, \quad \forall i \in I \end{aligned}$$

For all $i \in I$, we have,

$$v_i \geq \delta_i \ \left(\sum_{j=1}^{J} a_j \ x_{ij} \right) - \theta$$

and since $v_i \geq 0, \ \forall i \in I$, we have

$$v_i = \max \left\{ 0, \ \delta_i \ \left(\sum_{j=1}^{J} a_j \ x_{ij} \right) - \theta \right\}$$

as $x_{ij} \in \{0, 1\}$ and $\sum_{j=1}^{J} x_{ij} = 1$ it holds

$$v_i = \sum_{j=1}^{J} \max \left\{ 0, \ \delta_i \ a_j - \theta \right\} \ x_{ij}$$

the problem becomes

$$\begin{aligned} z = \min & \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \quad + \quad \Gamma \ \theta \ + \ \sum_{i=1}^{I} \sum_{j=1}^{J} \max \left\{ 0, \ \delta_i \ a_j - \theta \right\} \ x_{ij} \\ s.t. \quad & x \in \mathcal{X} \\ & \theta \geq 0 \end{aligned}$$

using $\Delta_k = \delta_i \ a_j$ the problem becomes

$$
\begin{aligned}
z = \min & \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \quad + \quad \Gamma \ \theta \ + \ \sum_{k=1}^{K} \max\{0, \ \Delta_k - \theta\} \ x_k \\
s.t. \quad & x \in \mathcal{X} \\
& \theta \geq 0
\end{aligned}
$$

By decomposing $\mathbb{R}^+$ into the intervals $[0, \Delta_K], [\Delta_K, \Delta_{K-1}], \ldots, [\Delta_2, \Delta_1]$ and $[\Delta_1, +\infty[$, we obtain

$$
\sum_{k=1}^{K} \max\{0, \ \Delta_k - \theta\} \ x_k = \begin{cases} \displaystyle\sum_{k=1}^{\ell-1} (\Delta_k - \theta) \ x_k & \text{if } \theta \in [\Delta_\ell, \ \Delta_{\ell-1}], \ell = K+1, \ldots, 2 \\ 0 & \text{if } \theta \in [\Delta_1, +\infty[\end{cases}
$$

Therefore,

$$
z = \min_{\ell=1,\ldots,K+1} z^\ell
$$

where for $\ell = 2, \ldots, K+1$:

$$
\begin{aligned}
z^\ell = \min & \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \quad + \quad \Gamma \ \theta \ + \ \sum_{k=1}^{\ell-1} (\Delta_k - \theta) \ x_k \\
s.t. \quad & x \in \mathcal{X} \\
& \theta \in [\Delta_\ell, \ \Delta_{\ell-1}]
\end{aligned}
$$

and

$$
\begin{aligned}
z^1 = \min & \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \quad + \quad \Gamma \ \theta \\
s.t. \quad & x \in \mathcal{X} \\
& \theta \in [\Delta_1, +\infty[
\end{aligned}
$$

Since we are optimizing a linear function of θ over the interval $[\Delta_\ell, \ \Delta_{\ell-1}]$, the optimal is obtained for $\theta = \Delta_\ell$ or $\theta = \Delta_{\ell-1}$, and thus for $\ell = 2, \ldots, K+1$:

$$
\begin{aligned}
z^{\ell} = \min \Bigg\{ & \Gamma\ \Delta_{\ell} \ + \ \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{\ell-1} (\Delta_k - \Delta_{\ell})\ x_k \right), \\
& \Gamma\ \Delta_{\ell-1} \ + \ \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{\ell-1} (\Delta_k - \Delta_{\ell-1})\ x_k \right) \Bigg\} \\
= \min \Bigg\{ & \Gamma\ \Delta_{\ell} \ + \ \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{\ell} (\Delta_k - \Delta_{\ell})\ x_k \right), \\
& \Gamma\ \Delta_{\ell-1} \ + \ \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{\ell-1} (\Delta_k - \Delta_{\ell-1})\ x_k \right) \Bigg\}
\end{aligned}
$$

Thus,

$$
\begin{aligned}
z = \min \Bigg\{ & \Gamma\ \Delta_1 \ + \ \min_{x \in \mathcal{X}} \sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij}, \ldots, \\
& \Gamma\ \Delta_{\ell} \ + \ \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{\ell} (\Delta_k - \Delta_{\ell})\ x_k \right), \ldots, \\
& \min_{x \in \mathcal{X}} \left(\sum_{i=1}^{I} c_i \ \sum_{j=1}^{J} a_j \ x_{ij} \ + \ \sum_{k=1}^{K} \Delta_k \ x_k \right) \Bigg\}
\end{aligned}
$$

□

7.1.1 Application to MWTR problem with use of the Path-edges$^+$ formulation

We used Theorem 7.2 presented above to obtain a robust formulation for the MWTR problem using the Path-edges$^+$ formulation, presented in Chapter 5.

In order to simplify the reading we will refer to the Path-edges$^+$ formulation, presented in Chapter 5 as follow:

$$
\begin{aligned}
min \quad & z = \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} p_{ij}^{\ell} \\
s.t. \quad & y \in Y
\end{aligned}
$$

where Y represents the constrains (5.26) – (5.47) and y represents the set of variables $(x_{ij}, p_{ij}^{\ell}, f_{ij}^{k\ell})$. The binary variables x_{ij}, $i \in V_a$, $j \in V$, $i < j$ indicate whether edge $\{i, j\}$ belongs to the tree solution, binary variables p_{ij}^{ℓ} indicate whether the path P_{ij} connecting terminal node i to terminal node j has exactly ℓ edges and the binary flow variables $f_{ij}^{k\ell}, \forall i, j \in V_a \cup \{k, \ell\}, k, \ell \in V_t$, $i \neq j$ and $k < \ell$ indicate whether the flow traverses the edge $\{i, j\}$ belonging to the path connecting terminal node k to terminal node ℓ in the direction from node i to node j.

Assuming that each $\widetilde{d}_{ij}$, $i, j \in V_t, i < j$ takes values in the interval $[d_{ij}, d_{ij} + b_{ij}]$ with $b_{ij} \geq 0$, we are interested in minimizing the maximum cost z such that at most Γ of the coefficients of the objective function are allowed to change. Defining $M = \{i, j \in V_t : i < j\}$ and $S = \{(i, j) \in M : b_{ij} > 0\}$, the problem reads:

$$\widetilde{z} = \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \cdot p_{ij}^{\ell} \quad + \quad \max_{\{S : S \subset M, |S| \leq \Gamma\}} \sum_{\substack{(i,j) \in S \\ i<j}} b_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \, p_{ij}^{\ell}$$

$$s.t. \quad y \in Y$$

Using the approach we used before, we can express the inner maximization problem as a linear program:

$$\widetilde{z} = \min_{y \in Y} \left(\sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \cdot p_{ij}^{\ell} \quad + \quad \max \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} b_{ij} \, u_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \, p_{ij}^{\ell} \right)$$

$$s.t. \quad 0 \leq u_{ij} \leq 1, \; i, j \in V_t, \; i < j \tag{7.14}$$

$$\sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} u_{ij} \leq \Gamma \tag{7.15}$$

Dualizing the inner maximization problem and joining the two minimizations problems, we obtain the formulation that we designated as Robust-Dual formulation.

Robust-Deviation-Dual formulation

$$\widetilde{z} = \min \left(\sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell}\, p_{ij}^{\ell} \quad + \quad \Gamma\ \theta\ + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} v_{ij} \right)$$

$$\begin{aligned} s.t. \quad & y \in Y \\ & v_{ij} + \theta \ \geq \ b_{ij} \left(\sum_{\ell=2}^{n-1} 2^{-\ell}\, p_{ij}^{\ell} \right), \quad \forall i,j \in V_t, i<j \\ & \theta \geq 0 \\ & v_{ij} \geq 0, \quad \forall i,j \in V_t, i<j \end{aligned}$$

which is the classical dualization for robust linear programs and can be applied whenever the uncertainty is described by a polytope (here (7.14) and (7.15)).

We also applied Theorem 7.2. To simplify the implementation we considered that all the deviations $b_{ij}, i,j \in V_t, i<j$ are fixed and equal to $b \in \mathbb{R}^+$. Doing so we obtained the formulation that we named Robust-Theorem formulation.

Robust-Deviation formulation

$$\widetilde{z} = \min_{s=2\ldots n} z^s$$

and where for $s = 2 \ldots n$:

$$z^s = \min \left(\Gamma\ b\ 2^{-s} + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} \cdot p_{ij}^{\ell} \quad + \quad \sum_{k=1}^{s} b\left(2^{-k} - 2^{-s}\right) \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} p_{ij}^{k} \right)$$

$$s.t. \quad y \in Y$$

The computational results will be presented in Section 7.3.

7.2 Reducing the risk of high cost

Rockafellar and Uryasev [114, 115] introduced a distributionally robust problem which uses the risk measure, *Conditional Value-at-Risk* (CVaR).

The Conditional Value-at-Risk, also known as Mean Expected Loss, Mean Shortfall, and Tail VaR, is a modification of the Value-at-Risk (VaR), another risk measure commonly used [38].

The Value-at-Risk is the maximum loss which can occur with $\alpha \times 100\%$ confidence over a certain period of time [31]. The Conditional Value-at-Risk is the expected loss given that the loss exceeds VaR at a given confidence level [38].

In optimization modeling CVaR has superior properties than VaR, namely the fact that CVaR can be expressed as a minimization formula, that can be incorporated into optimization problems, preserving problem features like convexity [115].

Definition 7.4. [92, 111] Let X be a random loss variable, F_X be the cumulative distribution function of X, that is, $F_X(\theta) = \mathbb{P}(X \leq \theta)$, $\theta \in \mathbb{R}$ and $F_X^{-1}(y) = \min_{\theta \in \mathbb{R}} \{F_X(\theta) \geq y\}$. Given a probability level $\alpha,\ 0 < \alpha < 1$, the α-**VaR** of X is given by:

$$\mathrm{VaR}_\alpha(X) = \min_{\theta \in \mathbb{R}} \{\mathbb{P}\left(X \leq \theta\right) \geq \alpha\} = F_X^{-1}(\alpha).$$

Definition 7.5. [92] Let X be a continuous random loss variable. Given a probability level $\alpha,\ 0 < \alpha < 1$, the α-**CVaR** of X is given by:

$$\mathrm{CVaR}_\alpha(X) = \mathbb{E}\left[X : X \geq \mathrm{VaR}_\alpha(X)\right].$$

Pflug [111] defined the CVaR of a random variable X with confidence level $\alpha \in]0, 1[$ via an optimization problem as follows:

$$\mathrm{CVaR}_\alpha(X) := \inf_{\theta \in \mathbb{R}} \left\{ \theta + \frac{1}{1-\alpha} \mathbb{E}[X - \theta]^+ \right\} \tag{7.16}$$

where $\mathbb{E}(\cdot)$ denotes the expectation of a random variable and $[a]^+ = \max\{a, 0\}$.

Consider the optimization problem (7.4) presented above:

$$\min_{x \in \mathcal{X}} \ z := c^\intercal x = \sum_{i=1}^{n} c_i \ x_i$$

with $\mathcal{X} \subseteq \mathbb{R}^n_+$, and $c = (c_i) \in \mathbb{R}^n$ a cost vector.

As before the set of feasible solutions is not subject to uncertainty and the cost coefficients are uncertain and such that $\tilde{c} \in \mathcal{U}$, where $\mathcal{U} \subseteq \mathbb{R}^{n+1}$ is the uncertainty set.

The distributionally robust problem introduced by Rockafellar and Uryasev [114, 115] is given by:

$$\min_{x \in \mathcal{X}} \sup_{\mathbb{P} \in \mathcal{P}} \mathrm{CVaR}_{\alpha} \left[\sum_{i=1}^{n} \tilde{c}_i x_i \right] \tag{7.17}$$

where

$$\mathcal{P} = \{\mathbb{P} \in \mathcal{M}_{+}(\mathbb{R}^{n+1}) : \mathbb{P}[c \in \mathcal{U}] = 1, \mathbb{E}_{\mathbb{P}}[c^0] \leq \mu, \mathbb{E}[\max\{\mu - c^0, c^0 - \mu\}] \leq \sigma\},$$

the vector $c \in \mathcal{U}$ is a random vector that represents the uncertain problem parameters, such that $c^{\intercal} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n & 1 \end{bmatrix}$, $(c^0)^{\intercal} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}$, $\mathcal{M}_{+}(\mathbb{R}^{n+1})$ denotes the cone of nonnegative Borel measures supported on $\mathbb{R}^n$, $\mathbb{E}_{\mathbb{P}}[c^0]$ denotes the expectation of the random vector c^0, $(\mu_i) \in \mathbb{R}^n$ is the mean of the values obtained experimentally, $\mathbb{E}[\max\{\mu - c^0, c^0 - \mu\}]$ denotes the first center moment of the random vector c^0, and $\mu, \sigma \in \mathbb{R}$ are fixed parameters.

The uncertainty set is defined as

$$\mathcal{U} = \{c \in \mathbb{R}^{n+1} : Ac \leq b\} = [\underline{c}, \overline{c}]^n \times \{1\}$$

where $\underline{c}, \overline{c} \in \mathbb{R}^n$ are fixed parameters, $A \in \mathbb{R}^{(2n+2)\times(n+1)}$, $b \in \mathbb{R}^{n+1}$ are such that

$$A = \begin{bmatrix} -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & -1 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & -1 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -\underline{c}_1 \\ -\underline{c}_2 \\ \vdots \\ -\underline{c}_n \\ \overline{c}_1 \\ \overline{c}_2 \\ \vdots \\ \overline{c}_n \\ -1 \\ 1 \end{bmatrix}$$

The first n rows represent the inequalities $c_i \geq \underline{c}_i \Leftrightarrow -c_i \leq -\underline{c}_i,\ i \in \{1, \cdots, n\}$. The following n rows represent the inequalities $c_i \leq \overline{c}_i,\ i \in \{1, \cdots, n\}$. The last two rows represent the equality $c_{n+1} = 1 \Leftrightarrow -c_{n+1} \leq -1$ and $c_{n+1} \leq 1$. Thus

$$\mathcal{U} = \{c \in \mathbb{R}^{n+1} : \underline{c}_i \leq c_i \leq \overline{c}_i,\ i \in \{1, \cdots, n\} \text{ and } c_{n+1} = 1\}$$

Using Pflug's [111] definition of CVaR, we obtain:

$$\text{CVaR}_\alpha \left[\sum_{i=1}^{n} c_i x_i\right] = \inf_{\theta \in \mathbb{R}} \left\{\theta + \frac{1}{1-\alpha}\ \mathbb{E}_{\mathbb{P}}\left[\left(\sum_{i=1}^{n} c_i x_i - \theta\right)^+\right]\right\} \tag{7.18}$$

where $\left(\sum_{i=1}^{n} c_i x_i - \theta\right)^+ = \max\left\{\sum_{i=1}^{n} c_i x_i - \theta,\ 0\right\}$, $\theta \in \mathbb{R}$ and $\alpha \in [0, 1]$ is the confidence level.

The objective criterion in problem (7.17) is set to the worst-case conditional value-at-risk at level α.

Adapting Theorem 3 from [79] to problem (7.17), we obtain

Theorem 7.3. *Problem* (7.17) *is equivalent to the following MILP.*

$$
\begin{aligned}
min \quad & \theta + \epsilon + \mu \sum_{i=1}^{n} \beta_i + \sigma \sum_{i=1}^{n} B_i \\
s.t. \quad & -\sum_{i=1}^{n} \underline{c}_i \gamma_i^1 + \sum_{i=1}^{n} \overline{c}_i \gamma_{n+i}^1 - \gamma_{2n+1}^1 + \gamma_{2n+2}^1 \leq \epsilon + \frac{1}{1-\alpha}\theta && (7.19)\\
& -\sum_{i=1}^{n} \underline{c}_i \gamma_i^2 + \sum_{i=1}^{n} \overline{c}_i \gamma_{n+i}^2 - \gamma_{2n+1}^2 + \gamma_{2n+2}^2 \leq \epsilon && (7.20)\\
& -\gamma_i^1 + \gamma_{n+i}^1 + \Lambda_{i0}^1 - \Lambda_{i1}^1 + \Lambda_{i2}^1 = \frac{1}{1-\alpha} x_i \qquad \forall i \in \{1, \cdots n\} && (7.21)\\
& -\gamma_i^2 + \gamma_{n+i}^2 + \Lambda_{i0}^2 - \Lambda_{i1}^2 + \Lambda_{i2}^2 = 0 \qquad \forall i \in \{1, \cdots n\} && (7.22)\\
& -\gamma_{2n+1}^1 + \gamma_{2n+2}^1 + \sum_{i=1}^{n} \mu_i \Lambda_{i1}^1 - \sum_{i=1}^{n} \mu_i \Lambda_{i2}^1 = 0 && (7.23)\\
& -\gamma_{2n+1}^2 + \gamma_{2n+2}^2 + \sum_{i=1}^{n} \mu_i \lambda_{i2}^2 - \sum_{i=1}^{n} \mu_i \Lambda_{i1}^2 = 0 && (7.24)\\
& \Lambda_{i0}^1 = \beta_i \qquad \forall i \in \{1, \cdots n\} && (7.25)\\
& \Lambda_{i0}^2 = \beta_i \qquad \forall i \in \{1, \cdots n\} && (7.26)\\
& \Lambda_{i1}^1 + \Lambda_{i2}^1 = B_i \qquad \forall i \in \{1, \cdots n\} && (7.27)\\
& \Lambda_{i1}^2 + \Lambda_{i2}^2 = B_i \qquad \forall i \in \{1, \cdots n\} && (7.28)\\
& x \in \mathcal{X} && (7.29)\\
& \epsilon \in \mathbb{R} && (7.30)\\
& \theta \in \mathbb{R} && (7.31)\\
& \beta_i, B_i \in \mathbb{R}_+ \qquad \forall i \in \{1, \cdots, n\} && (7.32)\\
& \gamma_i^1, \gamma_i^2 \in \mathbb{R}_+ \qquad \forall i \in \{1, \cdots, 2n+2\} && (7.33)\\
& \Lambda_{i0}^1, \Lambda_{i0}^2, \Lambda_{i1}^1, \Lambda_{i2}^1, \Lambda_{i1}^2, \Lambda_{i2}^2 \in \mathbb{R}_+ \qquad \forall i \in \{1, \cdots, n\} && (7.34)
\end{aligned}
$$

Proof. We will follow the proof presented in [79].

Replacing, in problem (7.17), $\text{CVaR}_\alpha \left[\sum_{i=1}^{n} c_i x_i \right]$ by its definition we obtain

$$\min_{x \in \mathcal{X}} \sup_{\mathbb{P} \in \mathcal{P}} \inf_{\theta \in \mathbb{R}} \left\{ \theta + \frac{1}{1-\alpha} \, \mathbb{E}_{\mathbb{P}} \left[\left(\sum_{i=1}^{n} c_i x_i - \theta \right)^+ \right] \right\}.$$

Since the expectation is linear in $\mathbb{P}$ and convex in θ, we can use Sion's min-max theorem [131][2] to reformulate problem (7.17) as

$$\begin{aligned} \min \quad & \sup_{\mathbb{P}\in\mathcal{P}} \ \theta + \frac{1}{1-\alpha}\ \mathbb{E}_{\mathbb{P}}\left[\left(\sum_{i=1}^{n} c_i x_i - \theta\right)^+\right] \\ s.t. \quad & x \in \mathcal{X}, \\ & \theta \in \mathbb{R} \end{aligned}$$

We can express the objective function of this problem as the optimal value of the moment problem

$$\begin{aligned} \max \quad & \theta + \int_{\mathcal{U}} \frac{1}{1-\alpha}\left(\sum_{i=1}^{n} c_i x_i - \theta\right)^+ \mathrm{d}\mathbb{P}(c) \\ s.t. \quad & \mathbb{P} \in \mathcal{M}_+(\mathbb{R}^{n+1}) \\ & \int_{\mathcal{U}} \mathrm{d}\mathbb{P}(c) = 1 \\ & \int_{\mathcal{U}} c_i \,\mathrm{d}\mathbb{P}(c) \le \mu, \quad \forall i \in \{1,\cdots,n\} \\ & \int_{\mathcal{U}} (\max\{\mu_i - c_i, c_i - \mu_i\})\,\mathrm{d}\mathbb{P}(c) \le \sigma, \quad \forall i \in \{1,\cdots,n\} \end{aligned}$$

Strong duality is guaranteed by Proposition 3.4 in [119], which is applicable since the ambiguity set $\mathcal{P}$ contains a Slater point. By associating dual variables ϵ, β_i and B_i,

[2]Let M be a compact convex subset of a linear topological space and Y a convex subset of a linear topological space. Let f be a real-valued function on $M \times N$ such that (i) $f(x,\cdot)$ is upper semicontinuous and quasi-concave on N for each $x \in M$; (ii) $f(\cdot, y)$ is a lower semicontinuous and quasi-convex on M for each $y \in N$. Then

$$\inf_{x\in M}\sup_{y\in N} f(x,y) = \sup_{y\in N}\inf_{x\in M} f(x,y)$$

$i = 1, \ldots, n$, to the constraints, the dual problem is given by:

$$\begin{aligned}
\min \quad & \theta + \epsilon + \mu \sum_{i=1}^{n} \beta_i + \sigma \sum_{i=1}^{n} B_i \\
s.t. \quad & \epsilon + \sum_{i=1}^{n} c_i \beta_i + \sum_{i=1}^{n} (\max\{\mu_i - c_i, c_i - \mu_i\}) B_i \geq \frac{1}{1-\alpha} \left(\sum_{i=1}^{n} c_i x_i - \theta \right)^+, \forall c \in \mathcal{U} \qquad (7.35) \\
& \epsilon \in \mathbb{R} \qquad (7.36) \\
& \beta_i, B_i \in \mathbb{R}_+, \quad \forall i \in \{1, \cdots, n\} \qquad (7.37)
\end{aligned}$$

Replacing $\left(\sum_{i=1}^{n} c_i x_i - \theta\right)^+$ with its definition, constraint (7.35) is equivalent to

$$\max_{c \in \mathcal{U}} \left\{ \max \left\{ \frac{1}{1-\alpha} \sum_{i=1}^{n} c_i x_i - \frac{1}{1-\alpha} \theta - \sum_{i=1}^{n} c_i \beta_i - \sum_{i=1}^{n} (\max\{\mu_i - c_i, c_i - \mu_i\}) B_i; \right. \right.$$
$$\left. \left. 0 - \sum_{i=1}^{n} c_i \beta_i - \sum_{i=1}^{n} (\max\{\mu_i - c_i, c_i - \mu_i\}) B_i \right\} \right\} \leq \epsilon$$

and equivalent to

$$\max \left\{ \max_{c \in \mathcal{U}} \left\{ \frac{1}{1-\alpha} \sum_{i=1}^{n} c_i x_i - \frac{1}{1-\alpha} \theta - \sum_{i=1}^{n} c_i \beta_i - \sum_{i=1}^{n} (\max\{\mu_i - c_i, c_i - \mu_i\}) B_i \right\} \leq \epsilon \right.$$
$$\left. \max_{c \in \mathcal{U}} \left\{ 0 - \sum_{i=1}^{n} c_i \beta_i - \sum_{i=1}^{n} (\max\{\mu_i - c_i, c_i - \mu_i\}) B_i \right\} \leq \epsilon \right\}$$

Expressing the maximization embedded in the first constraint as the optimal value of an LP we obtain

$$\begin{aligned}
\max \quad & \frac{1}{1-\alpha} \sum_{i=1}^{n} c_i x_i - \frac{1}{1-\alpha} \theta + \sum_{i=1}^{n} \eta_i^1 \beta_i + \sum_{i=1}^{n} \eta_i^2 B_i \\
s.t. \quad & Ac \leq b \\
& \eta_i^1 \leq -c_i && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(\mu_i - c_i) && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(c_i - \mu_i) && \forall i \in \{1, \cdots, n\} \\
& c_i \in \mathbb{R} && \forall i \in \{1, \cdots, n\} \\
& \eta_i^1, \eta_i^2 \in \mathbb{R} && \forall i \in \{1, \cdots, n\}
\end{aligned}$$

Expressing the maximization embedded in the second constraint as the optimal value of an LP we obtain

$$\begin{aligned}
\max \quad & 0 + \sum_{i=1}^{n} \eta_i^1 \beta_i + \sum_{i=1}^{n} \eta_i^2 B_i \\
s.t. \quad & Ac \leq b \\
& \eta_i^1 \leq -c_i && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(\mu_i - c_i) && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(c_i - \mu_i) && \forall i \in \{1, \cdots, n\} \\
& c_i \in \mathbb{R} && \forall i \in \{1, \cdots, n\} \\
& \eta_i^1, \eta_i^2 \in \mathbb{R} && \forall i \in \{1, \cdots, n\}
\end{aligned}$$

Replacing A and b with the respective definition we obtain the following two problems.

First problem:

$$\begin{aligned}
\max \quad & \frac{1}{1-\alpha} \sum_{i=1}^{n} c_i x_i - \frac{1}{1-\alpha} \theta + \sum_{i=1}^{n} \eta_i^1 \beta_i + \sum_{i=1}^{n} \eta_i^2 B_i \\
s.t. \quad & -c_i \leq -\underline{c}_i && \forall i \in \{1, \cdots, n\} \\
& c_i \leq \overline{c}_i && \forall i \in \{1, \cdots, n\} \\
& -c_{n+1} \leq -1 \\
& c_{n+1} \leq 1 \\
& \eta_i^1 \leq -c_i && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(\mu_i - c_i) && \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(c_i - \mu_i) && \forall i \in \{1, \cdots, n\} \\
& c_i \in \mathbb{R} && \forall i \in \{1, \cdots, n\} \\
& \eta_i^1, \eta_i^2 \in \mathbb{R} && \forall i \in \{1, \cdots, n\}
\end{aligned}$$

Second problem:

$$
\begin{aligned}
\max \quad & 0 + \sum_{i=1}^{n} \eta_i^1 \beta_i + \sum_{i=1}^{n} \eta_i^2 B_i & \\
s.t. \quad & -c_i \leq -\underline{c}_i & \forall i \in \{1, \cdots, n\} \\
& c_i \leq \overline{c}_i & \forall i \in \{1, \cdots, n\} \\
& -c_{n+1} \leq -1 & \\
& c_{n+1} \leq 1 & \\
& \eta_i^1 \leq -c_i & \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(\mu_i - c_i) & \forall i \in \{1, \cdots, n\} \\
& \eta_i^2 \leq -(c_i - \mu_i) & \forall i \in \{1, \cdots, n\} \\
& c_i \in \mathbb{R} & \forall i \in \{1, \cdots, n\} \\
& \eta_i^1, \eta_i^2 \in \mathbb{R} & \forall i \in \{1, \cdots, n\}
\end{aligned}
$$

The dual of the first problem is given by

$$
\begin{aligned}
\min \quad & -\sum_{i=1}^{n} \underline{c}_i \gamma_i^1 + \sum_{i=1}^{n} \overline{c}_i \gamma_{n+i}^1 - \gamma_{2n+1}^1 + \gamma_{2n+2}^1 - \frac{1}{1-\alpha}\theta & \\
s.t. \quad & -\gamma_i^1 + \gamma_{n+i}^1 + \Lambda_{i0}^1 - \Lambda_{i1}^1 + \Lambda_{i2}^1 = \frac{1}{1-\alpha} x_i & \forall i \in \{1, \cdots n\} \\
& -\gamma_{2n+1}^1 + \gamma_{2n+2}^1 + \sum_{i=1}^{n} \mu_i \Lambda_{i1}^1 - \sum_{i=1}^{n} \mu_i \Lambda_{i2}^1 = 0 & \\
& \Lambda_{i0}^1 = \beta_i & \forall i \in \{1, \cdots n\} \\
& \Lambda_{i1}^1 + \Lambda_{i2}^1 = B_i & \forall i \in \{1, \cdots n\} \\
& \theta \in \mathbb{R} & \\
& \beta_i, B_i \in \mathbb{R}_+ & \forall i \in \{1, \cdots, n\} \\
& \gamma_i^1, \in \mathbb{R}_+ & \forall i \in \{1, \cdots, 2n+2\} \\
& \Lambda_{i0}^1, \Lambda_{i1}^1, \Lambda_{i2}^1, \in \mathbb{R}_+ & \forall i \in \{1, \cdots, n\}
\end{aligned}
$$

The dual of the second problem is given by

$$
\begin{aligned}
\min \quad & -\sum_{i=1}^{n} \underline{c}_i \gamma_i^2 + \sum_{i=1}^{n} \overline{c}_i \gamma_{n+i}^2 - \gamma_{2n+1}^2 + \gamma_{2n+2}^2 & \\
s.t. \quad & -\gamma_i^2 + \gamma_{n+i}^2 + \Lambda_{i0}^2 - \Lambda_{i1}^2 + \Lambda_{i2}^2 = 0 & \forall i \in \{1, \cdots n\} \\
& -\gamma_{2n+1}^2 + \gamma_{2n+2}^2 + \sum_{i=1}^{n} \mu_i \Lambda_{i1}^2 - \sum_{i=1}^{n} \mu_i \Lambda_{i2}^2 = 0 & \\
& \Lambda_{i0}^2 = \beta_i & \forall i \in \{1, \cdots n\} \\
& \Lambda_{i1}^2 + \Lambda_{i2}^2 = B_i & \forall i \in \{1, \cdots n\} \\
& \theta \in \mathbb{R} & \\
& \beta_i, B_i \in \mathbb{R}_+ & \forall i \in \{1, \cdots, n\} \\
& \gamma_i^2, \in \mathbb{R}_+ & \forall i \in \{1, \cdots, 2n+2\} \\
& \Lambda_{i0}^2, \Lambda_{i1}^2, \Lambda_{i2}^2, \in \mathbb{R}_+ & \forall i \in \{1, \cdots, n\}
\end{aligned}
$$

□

7.2.1 Application to MWTR problem with use of the Path-edges$^+$ formulation

We used Theorem 7.3 presented above to obtain another robust formulation for the MWTR problem using the Path-edges$^+$ formulation, presented in Chapter 5.

In order to simplify the reading, as before, we will refer to the Path-edges$^+$ formulation, presented in Chapter 5 as follow:

$$
\begin{aligned}
min \quad & z = \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j > i}} d_{ij} \sum_{\ell=2}^{n-1} 2^{-\ell} p_{ij}^{\ell} \\
s.t. \quad & y \in Y
\end{aligned}
$$

where Y is the set of solutions satisfying (5.26) – (5.47) and y represents the set of variables $(x_{ij}, p_{ij}^{\ell}, f_{ij}^{k\ell})$. The binary variables x_{ij}, $i \in V_a$, $j \in V$, $i < j$ indicate whether edge $\{i, j\}$ belongs to the tree solution, binary variables p_{ij}^{ℓ} indicate whether the path P_{ij} connecting

terminal node i to terminal node j has exactly ℓ edges and the binary flow variables $f_{ij}^{k\ell}, \forall i,j \in V_a \cup \{k,\ell\}, k,\ell \in V_t,\ i \neq j$ and $k < \ell$ indicate whether the flow traverses the edge $\{i,j\}$ belonging to the path connecting terminal node k to terminal node ℓ in the direction from node i to node j.

In this case we assume that each $\widetilde{d}_{ij},\ i,j \in V_t, i < j$ takes values in the interval $[\underline{d}_{ij}, \overline{d}_{ij}]$. Consider $\mu, \sigma, \alpha \in \mathbb{R}$ fixed parameters and for all $i,j \in V_t, i < j$, $\mu_{ij} \in \mathbb{R}$ are the mean of the values of $\widetilde{d}_{ij}$ obtained experimentally. Applying Theorem 7.3, we obtain the formulation that we designated as Robust-CVaR formulation.

Robust-CVaR formulation

$$\widetilde{z} = \min\ \theta + \epsilon + \mu \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \beta_{ij} + \sigma \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} B_{ij}$$

$$s.t.\ -\sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \underline{d}_{ij}\ \gamma^1_{ij} + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \overline{d}_{ij}\ \lambda^1_{ij} - \rho_1 + \rho_2 \leq \epsilon + \frac{1}{1-\alpha}\theta$$

$$-\sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \underline{d}_{ij}\ \gamma^2_{ij} + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \overline{d}_{ij}\ \lambda^2_{ij} - \rho_3 + \rho_4 \leq \epsilon$$

$$-\gamma^1_{ij} + \lambda^1_{ij} + \Lambda^1_{ij} - \Theta^1_{ij} + \Upsilon^1_{ij} = \frac{1}{1-\alpha} \sum_{\ell=2}^{n-1} 2^\ell\ p^\ell_{ij} \qquad \forall i,j \in V_t,\ i<j$$

$$-\gamma^2_{ij} + \lambda^2_{ij} + \Lambda^2_{ij} - \Theta^2_{ij} + \Upsilon^2_{ij} = 0 \qquad \forall i,j \in V_t,\ i<j$$

$$-\rho_1 + \rho_2 + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \mu_{ij}\ \Theta^1_{ij} - \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \mu_{ij}\ \Upsilon^1_{ij} = 0$$

$$-\rho_3 + \rho_4 + \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \mu_{ij}\ \Theta^2_{ij} - \sum_{i \in V_t} \sum_{\substack{j \in V_t \\ j>i}} \mu_{ij}\ \Upsilon^2_{ij} = 0$$

$$\Lambda^1_{ij} = \beta_{ij} \qquad \forall i,j \in V_t,\ i<j$$

$$\Lambda^2_{ij} = \beta_{ij} \qquad \forall i,j \in V_t,\ i<j$$

$$\Theta^1_{ij} + \Upsilon^1_{ij} = B_{ij} \qquad \forall i,j \in V_t,\ i<j$$

$$\Theta^2_{ij} + \Upsilon^2_{ij} = B_{ij} \qquad \forall i,j \in V_t,\ i<j$$

$$y \in Y$$

$$\epsilon,\ \theta \in \mathbb{R}$$

$$\beta_{ij},\ B_{i,j}\ \in \mathbb{R}_+ \qquad \forall i,j \in V_t,\ i<j$$

$$\gamma^1_{ij},\ \gamma^2_{ij},\ \lambda^1_{ij},\ \lambda^2_{ij},\ \Lambda^1_{ij},\ \Lambda^2_{ij},\ \Theta^1_{ij}, \Theta^2_{ij}, \Upsilon^1_{ij},\ \Upsilon^2_{ij}\ \in \mathbb{R}_+ \qquad \forall i,j \in V_t,\ i<j$$

$$\rho_i \in \mathbb{R}_+ \qquad \forall i \in \{1, \cdots, 4\}$$

7.3 Computational results

In this section we present the computational results we obtained by implementing the three robust formulations, the Robust-Deviation-Dual formulation, the Robust-Deviation formulation and the Robust-CVaR formulation. The computational tests were performed on an Intel(R) Core(TM) i7-3770 CPU 3.40 GHz processor and 16Gb of RAM.

The three formulations were implemented using the Mosel language and solved with FICO Xpress 7.8 [1] (Xpress-IVE 1.24.06 64 bit, Xpress-Optimizer 27.01.02 and Xpress-Mosel 3.8.0).

We used the instances presented in Chapter 3 and Chapter 4. As we described, for each matrix, $D = (d_{ij})$ we generated ten random values belonging to $[d_{ij}, d_{ij} + a \times d_{ij}]$, where $a \in \{0,1; 0,15; 0,2; 1\}$. We also run ten simulations using the network-level simulator NS-3 [3], for three routing trees, where we defined the delays on the intermediate routers to be random. So for each matrix we have ten different values, $d_{ij}^k,\ k \in \{1, \ldots, 10\}$, for each distance d_{ij}.

In the implementation of the Robust-CVaR formulation for each matrix, we used for $\underline{d}_{ij}, i, j \in V_t, i < j$, the minimum of the ten values of $d_{ij}^k,\ k \in \{1, \ldots, 10\}$, for $\overline{d}_{ij}$ the maximum of the ten values and $\mu_{ij} = \dfrac{\sum_{k=1}^{10} d_{ij}^k}{10}$. Regarding the parameter α, after several experiences, we concluded that the choice of the value of α has little influence on the optimal solution obtained. We choose to use $\alpha = 0.02$. We defined parameters

$$\mu = \frac{\sum_{\substack{i,j \in V_t \\ j>i}} d_{ij}}{total} \text{ and } \sigma = \frac{\sum_{\substack{i,j \in V_t \\ j>i}} \max\{\mu_{ij} - d_{ij}, d_{ij} - \mu_{ij}\}}{total}$$

where $total = \frac{n(n-1)}{2}$.

In the implementation of Robust-Deviation-Dual formulation, for each matrix, we used $b_{ij} = \overline{d}_{ij} - \mu_{ij}$, for all $i, j \in V_t, i < j$. Regarding the parameter Γ we run the implementation for all $\Gamma \in \left\{1, \ldots, \dfrac{n(n-1)}{2}\right\}$. For all $\Gamma \in \left\{1, \ldots, \dfrac{n(n-1)}{2}\right\}$ the optimum solution obtained was the same, the only change was the runtime, that increased as Γ increased.

In the implementation of Robust-Deviation formulation, for each matrix, we used $b = \max_{i,j \in V_t, i<j} \overline{d}_{ij} - \min_{i,j \in V_t, i<j} \underline{d}_{ij}$. As for the implementation of Robust-Deviation-Dual

formulation, we run the implementation for all $\Gamma \in \left\{1, \ldots, \frac{n(n-1)}{2}\right\}$ and the conclusions were the same.

Since the chose of Γ does not change the optimum solution obtained, we chose only to present the computation results for $\Gamma = 10$.

The computational results are summarized in Tables 7.1 - 7.5 in which the first column, labeled **Matrix**, indicates the name of the matrix instance used and the second column, labeled $|V_t|$, indicates the size of the instance. The third, forth and fifth columns concern the results of the Robust-Deviation-Dual formulation, from the sixth to the eighth columns the results of the Robust-Deviation formulation are presented, from the ninth to the eleventh columns are the results of the Robust-CVaR formulation and the twelfth and thirteenth columns concern Path-edges$^+$ formulation. We present again the results of the Path-edges$^+$ formulation to facilitate the comparison of the values obtained. The columns labeled **T** show the execution time, in seconds, used to solve the instance and having a maximum runtime of 7200 seconds. The columns labeled **W** present the optimum value obtained or the best value obtained having a runtime limit of 7200 seconds, where W stands for $\sum_{i \in V_a} \sum_{\substack{j \in V \\ j > i}} w_{ij}$. The columns labeled **GAP** present the gap between the value obtained by the robust formulation and the best lower bound value: $GAP = \frac{W_R - LB}{W_R} \times 100$, where W_R represents the best value obtained by the robust formulation within the runtime imposed and LB the best lower bound value.

Table 7.1: Computational results for data generated with $a = 0.1$.

		Robust-Deviation-Dual			Robust-Deviation			Robust-CVaR			Path-edges$^+$	
Matrix	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**
	5	0.01	0.0515	0	0.02	0.0515	0	0.01	0.0515	0	0	0.0515
	6	0.05	0.0622	0	0.16	0.0622	0	0.08	0.0622	0	0.06	0.0622
	7	2.95	0.0668	0	12.79	0.0668	0	0.53	0.0668	0	2.42	0.0668
A10M391	8	7.97	0.0734	0	234.63	0.0734	0	6.13	0.0734	0	14.87	0.0734
	9	18.34	0.0998	0	308.25	0.0998	0	28.92	0.0998	0	9.89	0.0998
	10	2578.25	0.1124	0	> 7200	0.1138	1.2	705.67	0.1137	1.1	197.67	0.1124
	11	> 7200	0.1412	0.1	> 7200	0.141	0	2496.82	0.1412	0.1	1197.99	0.141
	5	0	0.0893	0	0.02	0.0893	0	0.02	0.0893	0	0	0.0893
	6	0.05	0.1115	0	0.08	0.1115	0	0.11	0.1129	1.3	0.05	0.1115
	7	0.78	0.1343	0	10.27	0.1343	0	0.3	0.1343	0	0.42	0.1343
A10Pri	8	3.26	0.1351	0	66.77	0.1351	0	0.48	0.1351	0	1.89	0.1351
	9	27.68	0.1364	0	162.15	0.1364	0	14.1	0.1364	0	12.93	0.1364
	10	192.18	0.1361	0	> 7200	0.1449	6.1	234.95	0.1361	0	138.33	0.1361
	11	3044.03	0.1794	0	> 7200	0.1794	0	2661.78	0.1794	0	574.74	0.1794
	5	0.01	0.109	0	0.02	0.109	0	0.02	0.109	0	0.01	0.109
	6	0.42	0.1215	0	1.62	0.1215	0	0.08	0.126	3.5	0.66	0.1215
	7	1.05	0.1432	0	10.75	0.1432	0	1.95	0.1432	0	1.51	0.1432
A10M887	8	7	0.1858	0	45.93	0.1858	0	9.14	0.1858	0	9.45	0.1858
	9	29.95	0.1972	0	221.5	0.1972	0	25.93	0.1972	0	9.23	0.1972
	10	202.77	0.2156	0	> 7200	0.2156	0	172.69	0.2156	0	276.4	0.2156
	11	> 7200	0.2263	0.9	> 7200	0.2243	0	3110.58	0.2243	0	2253.32	0.2243
	5	0	0.4266	0	0.02	0.4266	0	0	0.4266	0	0.02	0.4266
A10S7	6	0.05	0.4579	0	0.11	0.4579	0	0.08	0.4579	0	0.03	0.4579
	7	0.42	0.5345	0.3	2.93	0.5328	0	0.28	0.5345	0.3	0.13	0.5328
	5	0.01	0.1997	0.1	0.02	0.1995	0	0.02	0.1995	0	0	0.1995
	6	0.78	0.2906	0	1.25	0.2906	0	0.06	0.2906	0	0.52	0.2906
	7	1.9	0.318	0.2	3.1	0.3173	0	0.28	0.3177	0.1	0.69	0.3173
A10S15	8	8.49	0.4	0.3	53.85	0.3987	0	3.9	0.3997	0.3	5.74	0.3987
	9	45.02	0.415	0	110.21	0.415	0	50.54	0.415	0	9.86	0.415
	10	336.18	0.4529	0	> 7200	0.4962	8.7	358.16	0.4535	0.2	170.62	0.4528
	11	> 7200	0.6612	0	> 7200	0.6612	0	6595.52	0.6612	0	895.27	0.6612
	5	0.01	0.3152	0	0.02	0.3152	0	0.01	0.3166	0.4	0	0.3152
	6	0.75	0.3585	0	0.72	0.3583	0	0.05	0.3585	0	0.06	0.3583
	7	3.18	0.397	0.5	6.88	0.3951	0	0.45	0.3957	0.2	0.86	0.3951
A10S20	8	19.42	0.4827	0.4	42.68	0.4807	0	14.25	0.4819	0.3	11.54	0.4807
	9	64.97	0.5464	0.2	272.72	0.5454	0	151.16	0.5464	0.2	12.9	0.5454
	10	229.02	0.6058	0.2	> 7200	0.6366	5	807.58	0.6057	0.2	206.08	0.6048
	11	1553.93	0.6439	0.2	> 7200	0.643	0	> 7200	0.6822	5.8	3055.13	0.6429

Table 7.2: Computational results for data generated with $a = 0.15$.

		Robust-Deviation-Dual			Robust-Deviation			Robust-CVaR			Path-edges$^+$	
Matrix	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**
	5	0.02	0.0529	0	0.02	0.0529	0	0.01	0.0529	0	0.01	0.0529
	6	0.22	0.0637	0	0.13	0.0637	0	0.08	0.0638	0.1	0.08	0.0637
	7	3.28	0.0685	0	10.58	0.0685	0	0.75	0.0685	0	1.37	0.0685
A15M391	8	6.15	0.0752	0	373.25	0.0752	0	9.55	0.0752	0	16.43	0.0752
	9	23.26	0.1021	0	362.14	0.1021	0	30.92	0.1021	0	13.37	0.1021
	10	4347.56	0.1153	0	> 7200	0.1166	1.2	1106.57	0.1166	1.2	211.55	0.1153
	11	> 7200	0.1544	7.2	> 7200	0.1447	1	3209.11	0.1449	1.1	2032.65	0.1433
	5	0.01	0.0917	0	0.02	0.0917	0	0	0.0917	0	0.02	0.0917
	6	0.05	0.1143	0	0.11	0.1143	0	0.09	0.1158	1.3	0.02	0.1143
	7	0.89	0.1379	0	6.6	0.1379	0	0.3	0.1379	0	0.67	0.1379
A15Pri	8	3	0.1386	0	41.65	0.1386	0	0.52	0.1386	0	2.21	0.1386
	9	65.96	0.1397	0	166.45	0.1397	0	18.19	0.1397	0	12.68	0.1397
	10	219.62	0.1393	0	> 7200	0.1393	0	280.16	0.1393	0	185.83	0.1393
	11	6431.95	0.1841	0	> 7200	0.1841	0	2607.31	0.1841	0	322.87	0.1841
	5	0.01	0.1118	0	0.02	0.1118	0	0	0.1118	0	0.01	0.1118
	6	0.58	0.1245	0	1.84	0.1245	0	0.33	0.129	3.5	0.47	0.1245
	7	1.34	0.1517	3.3	9.97	0.1466	0	1.76	0.1466	0	1.86	0.1466
A15M887	8	8.64	0.1905	0	233.13	0.1905	0	8.3	0.1905	0	11.61	0.1905
	9	18.47	0.202	0	304.79	0.202	0	11.4	0.202	0	9.7	0.202
	10	270.11	0.2205	0	> 7200	0.2205	0	71.79	0.2205	0	128.84	0.2205
	11	> 7200	0.2468	6.9	> 7200	0.2297	0	5629.05	0.2297	0	1881.22	0.2297
	5	0	0.4377	0	0.02	0.4377	0	0	0.4377	0	0	0.4377
A15S7	6	0.05	0.4694	0	0.08	0.4694	0	0.06	0.4694	0	0.05	0.4694
	7	0.53	0.549	0.5	2.78	0.5465	0	0.28	0.549	0.5	0.09	0.5465
	5	0.02	0.2046	0	0.01	0.2046	0	0	0.2046	0	0.01	0.2046
	6	0.84	0.2981	0	0.12	0.2981	0	0.06	0.2981	0	0.23	0.2981
	7	1.72	0.3263	0.3	4.07	0.3254	0.1	0.22	0.3259	0.2	0.72	0.3252
A15S15	8	14.91	0.4112	0.5	88.22	0.4092	0	9.49	0.4107	0.4	3.79	0.4092
	9	65.58	0.4262	0.1	102.29	0.426	0	23.07	0.4262	0.1	10.37	0.426
	10	209.32	0.4651	0.1	> 7200	0.4868	4.6	220.79	0.4651	0.1	119.43	0.4646
	11	> 7200	0.6788	0	> 7200	0.6788	0	1405.31	0.6932	2.1	902.2	0.6788
	5	0	0.3274	1.2	0.02	0.3233	0	0	0.3253	0.6	0.01	0.3233
	6	0.94	0.3678	0.1	0.11	0.3678	0	0.11	0.3678	0.1	0.16	0.3678
	7	3.09	0.4079	0.7	5.26	0.405	0	0.36	0.4059	0.2	0.75	0.405
A15S20	8	23.48	0.4958	0.6	448.03	0.4928	0	6.21	0.4947	0.4	7.96	0.4928
	9	96.97	0.5609	0.3	448.33	0.5593	0	33.31	0.5609	0.3	11.75	0.5593
	10	722.05	0.622	0.3	> 7200	0.6529	5	610.84	0.6218	0.2	179.48	0.6204
	11	1944.82	0.6613	0.2	> 7200	0.66	0	2432.34	0.661	0.2	1804.07	0.6598

Table 7.3: Computational results for data generated with $a = 0.2$.

		Robust-Deviation-Dual			Robust-Deviation			Robust-CVaR			Path-edges$^+$	
Matrix	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**
	5	0.01	0.0543	0	0.02	0.0543	0	0.02	0.0543	0	0	0.0543
	6	0.16	0.0652	0	0.33	0.0652	0	0.08	0.0653	0.1	0.06	0.0652
	7	2.51	0.0702	0	13.01	0.0702	0	1.51	0.0702	0	1.4	0.0702
A20M391	8	4.45	0.077	0	250.16	0.077	0	4.13	0.0775	0.7	11.59	0.077
	9	76.85	0.1044	0	359.28	0.1044	0	59.65	0.1044	0	10.39	0.1044
	10	4008.07	0.1181	0	> 7200	0.1195	1.2	1432.04	0.1196	1.2	274.01	0.1181
	11	> 7200	0.1483	1.1	> 7200	0.1467	0	> 7200	0.1573	6.8	1793.47	0.1467
	5	0	0.0942	0	0.01	0.0942	0	0.02	0.0942	0	0.01	0.0942
	6	0.05	0.1171	0	0.17	0.1171	0	0.05	0.1186	1.3	0.05	0.1171
	7	0.89	0.1414	0	8.03	0.1414	0	0.25	0.1414	0	0.89	0.1414
A20Pri	8	4.52	0.1422	0	72.88	0.1422	0	0.47	0.1422	0	0.92	0.1422
	9	27.77	0.1431	0	180.16	0.1431	0	14.76	0.1431	0	9.08	0.1431
	10	295.14	0.1427	0	> 7200	0.1521	6.2	187.78	0.1427	0	119.22	0.1427
	11	> 7200	0.1889	0	> 7200	0.1889	0	2315.31	0.1889	0	569.43	0.1889
	5	0	0.1147	0	0.01	0.1147	0	0.01	0.1147	0	0.02	0.1147
	6	0.44	0.1274	0	1.31	0.1274	0	0.17	0.1319	3.5	0.75	0.1274
	7	1.15	0.1553	3.4	10.64	0.1536	2.3	1.4	0.15	0	1.64	0.15
A20M887	8	8.97	0.1953	0	198.87	0.1953	0	4.15	0.1953	0	11.89	0.1953
	9	20.39	0.2067	0	234.22	0.2067	0	13.07	0.2067	0	9.02	0.2067
	10	1190.77	0.2253	0	> 7200	0.2253	0	589.53	0.2253	0	120.14	0.2253
	11	> 7200	0.2412	2.5	> 7200	0.2352	0	3201.56	0.2352	0	2079.15	0.2352
	5	0	0.4488	0	0.01	0.4488	0	0.02	0.4488	0	0.02	0.4488
A20S7	6	0.03	0.481	0	0.08	0.481	0	0.06	0.481	0	0.05	0.481
	7	0.30	0.5635	0.6	2.45	0.5602	0	0.16	0.5635	0.6	0.14	0.5602
	5	0.02	0.2098	0	0.02	0.2098	0	0.01	0.2098	0	0.01	0.2098
	6	0.78	0.3055	0	1.28	0.3055	0	0.05	0.3055	0	0.53	0.3055
	7	2.43	0.3347	0.4	3.15	0.3334	0	0.25	0.3341	0.2	0.87	0.3334
A20S15	8	10.15	0.4223	0.6	33.06	0.4197	0	3.93	0.4217	0.5	7.71	0.4197
	9	31.95	0.4372	0.1	109.20	0.4372	0.1	33.34	0.4372	0.1	8.52	0.4369
	10	370.69	0.4773	0.1	> 7200	0.5229	8.9	86.53	0.4772	0.1	158.62	0.4766
	11	> 7200	0.6965	0	> 7200	0.6964	0	2343.12	0.7121	2.2	909.73	0.6964
	5	0.02	0.3341	0.8	0.02	0.3315	0	0.02	0.3341	0.8	0	0.3315
	6	0.61	0.3775	0.1	0.61	0.3772	0	0.08	0.3775	0.1	0.08	0.3772
	7	3.5	0.4188	0.9	4.8	0.415	0	0.41	0.4162	0.3	0.87	0.415
A20S20	8	9.27	0.5089	0.8	148.4	0.5049	0	9.3	0.5207	3	13	0.5049
	9	79.67	0.5754	0.4	414.09	0.5733	0	21.64	0.5857	2.1	11.45	0.5733
	10	330.1	0.6381	0.3	> 7200	0.6693	5	418.5	0.6494	2.1	178.03	0.6359
	11	2112.57	0.6787	0.3	> 7200	0.6768	0	2029.97	0.6784	0.2	944.07	0.6768

Table 7.4: Computational results for data generated with $a = 1$.

		Robust-Deviation-Dual			Robust-Deviation			Robust-CVaR			Path-edges$^+$	
Matrix	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**
	5	0.01	0.0769	2.8	0.01	0.0748	0	0	0.0769	2.8	0	0.0748
	6	0.76	0.0882	0	0.14	0.0882	0	0.06	0.0923	4.4	0.08	0.0882
	7	4.43	0.0978	1	13.31	0.0968	0	0.27	0.0974	0.7	1.65	0.0968
A100M391	8	10.89	0.1095	4.6	315	0.1063	1.8	0.56	0.1100	5	9.64	0.1044
	9	229.37	0.143	0	465.71	0.143	0	26.74	0.1436	0.4	13.6	0.143
	10	> 7200	0.1671	1.8	> 7200	0.1647	0.4	1129.24	0.1671	1.8	279.43	0.164
	11	> 7200	0.2131	3.7	> 7200	0.2052	0	5949.4	0.2107	2.6	3594	0.2052
	5	0	0.1338	0	0.01	0.1338	0	0.02	0.1338	0	0	0.1338
	6	0.03	0.1618	0	0.13	0.1618	0	0.08	0.1646	1.7	0.05	0.1618
	7	1.98	0.1986	1	13.91	0.1966	0	0.22	0.1986	1	2.18	0.1966
A100Pri	8	8.21	0.1991	0	172.19	0.1991	0	0.48	0.1991	0	1.93	0.1991
	9	41.68	0.2035	0	314.45	0.2035	0	18.81	0.2035	0	9.77	0.2035
	10	279.61	0.1966	0	> 7200	0.1966	0	612.55	0.1966	0	94.63	0.1966
	11	> 7200	0.3033	12.5	> 7200	0.2694	1.5	2251.51	0.2655	0	1008.31	0.2655
	5	0	0.1601	0	0.02	0.1601	0	0.02	0.1688	5.2	0	0.1601
	6	0.98	0.1756	0	0.44	0.1756	0	0.05	0.1843	4.8	0.08	0.1756
	7	1.78	0.2136	3.3	9.36	0.209	1.2	0.28	0.2225	7.2	2.45	0.2065
A100M887	8	11.37	0.2751	1.3	762.9	0.2786	2.6	2.45	0.2753	1.4	2.75	0.2714
	9	63.04	0.2833	0	344.26	0.2869	1.2	8.3	0.3014	6	8.08	0.2833
	10	6899.47	0.3064	0	> 7200	0.3259	6	116.81	0.3196	4.1	144.63	0.3064
	11	> 7200	0.3731	14.8	> 7200	0.3177	0	5503.58	0.3288	3.4	3532.92	0.3177
	5	0	0.6269	0	0.01	0.6269	0	0.01	0.6479	3.2	0	0.6269
A100S7	6	0.05	0.6662	0	0.09	0.6662	0	0.05	0.7048	5.5	0.03	0.6662
	7	2.12	0.7961	2.1	5.02	0.7791	0	0.23	0.8088	3.7	0.16	0.7791
	5	0.01	0.3009	3.1	0.01	0.2916	0	0	0.2916	0	0.01	0.2916
	6	0.81	0.4255	0	1.12	0.4255	0	0.06	0.4255	0	0.42	0.4255
	7	2.25	0.4683	1.6	4.71	0.4609	0	0.23	0.4654	1	0.61	0.4609
A100S15	8	8.92	0.6004	2.2	25.9	0.5873	0	3.7	0.5974	1.7	8.56	0.5873
	9	71.38	0.6133	0.3	120.73	0.6656	8.1	18	0.6133	0.3	9.63	0.6116
	10	866.4	0.6751	1	> 7200	0.7292	8.3	60.65	0.7032	4.9	142.88	0.6685
	11	> 7200	1.1498	14.9	> 7200	0.9782	0	> 7200	1.0302	5	700.94	0.9782
	5	0.02	0.4647	0.8	0.02	0.4611	0	0.01	0.4744	2.8	0.02	0.4611
	6	0.83	0.5295	0.3	0.83	0.5281	0	0.05	0.5295	0.3	0.28	0.5281
	7	3.56	0.5934	3.2	7.35	0.5742	0	0.23	0.5804	1.1	0.94	0.5742
A100S20	8	4.57	0.719	2.8	71.53	0.699	0	3.45	0.7119	1.8	8.53	0.699
	9	149.81	0.8075	1.3	373.35	0.7972	0	23.59	0.8632	7.6	12.36	0.7972
	10	3626.4	0.8958	1.2	> 7200	0.9307	4.9	351.91	0.9557	7.4	137.65	0.8853
	11	> 7200	0.9574	1	> 7200	0.9644	1.7	1757.95	0.9477	0	1042.36	0.9477

Table 7.5: Computational results for data from the networking application generated with random delays

		Robust-Deviation-Dual			Robust-Deviation			Robust-CVaR			Path-edges$^+$	
Matrix	$\|V_t\|$	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**	**GAP**	**T**	**W**
	5	0	0.3961	0	0.02	0.3961	0	0	0.3961	0	0	0.3961
SS7	6	0.05	0.4237	0	0.06	0.4237	0	0.08	0.4237	0	0.05	0.4237
	7	1.45	0.5048	1.3	3.2	0.4984	0	0.22	0.4984	0	0.09	0.4984
	5	0.02	0.2086	0.3	0.01	0.2081	0	0.01	0.2086	0.3	0.02	0.2081
	6	0.48	0.2955	0.4	0.62	0.2945	0	0.06	0.2951	0.2	0.14	0.2945
	7	1.08	0.3212	0.2	2.36	0.3205	0	0.34	0.3211	0.2	0.37	0.3205
SS15	8	9.33	0.4004	0.2	37.57	0.3997	0	5.8	0.4004	0.2	4.51	0.3997
	9	23.45	0.4164	0.2	102.93	0.4164	0.2	42.46	0.4161	0.2	8.5	0.4154
	10	353.31	0.4524	0.2	> 7200	0.4929	8.4	151.84	0.4524	0.2	103.65	0.4517
	11	4506.08	0.6518	0	> 7200	0.6518	0	2523.85	0.6528	0.2	762.98	0.6518
	5	0.01	0.3026	0	0.01	0.3026	0	0	0.3026	0	0.01	0.3026
	6	0.55	0.3432	0.1	0.5	0.3434	0.2	0.23	0.3434	0.2	0.19	0.3427
	7	1.95	0.3796	0.2	4.87	0.3789	0	0.36	0.3796	0.2	0.55	0.3789
SS20	8	12.56	0.4614	0.1	109.78	0.461	0	8.1	0.4614	0.1	7.67	0.461
	9	41.68	0.5224	0.1	155.95	0.5222	0	52.9	0.5224	0.1	12.21	0.522
	10	775.43	0.5788	0.1	> 7200	0.6518	11.3	636.45	0.5788	0.1	280.8	0.5784
	11	1933.76	0.614	0.1	> 7200	0.6136	0	2648.88	0.6136	0	1108.13	0.6136

The optimum solution within the time limit imposed is obtained in the following cases:

- by the Robust-Deviation-Dual formulation for all instances with $n < 11$ terminal nodes and for instances with $n = 11$ for matrices A10Pri, A10S20, A15Pri, A15S20, A20S20, SS15 and SS20;

- by the Robust-Deviation formulation for all instances with $n < 10$ terminal nodes;

- by the Robust-CVaR formulation for all instances except for instances with $n = 11$ for matrices A10S20, A20M391 and A100S15.

By analyzing the results, we can see that for $a = 0.1$ and $a = 0.15$ the robust optimum solution is mostly the same as the optimum solution obtained using the Path-edges$^+$ formulation. But as a increases the number of instances with the same robust optimum solution as the optimum solution obtained using the Path-edges$^+$ decreases. This is due to the fact that the uncertainty set increases with the increase of a.

Figure 7.1 displays, for instances with the same number of terminal nodes, the average time of the Robust-Deviation-Dual, Robust-Deviation, Robust-CVaR and Path-edges$^+$ formulation. As we can see Robust-CVaR formulation is the fastest of the three robust formulation used.

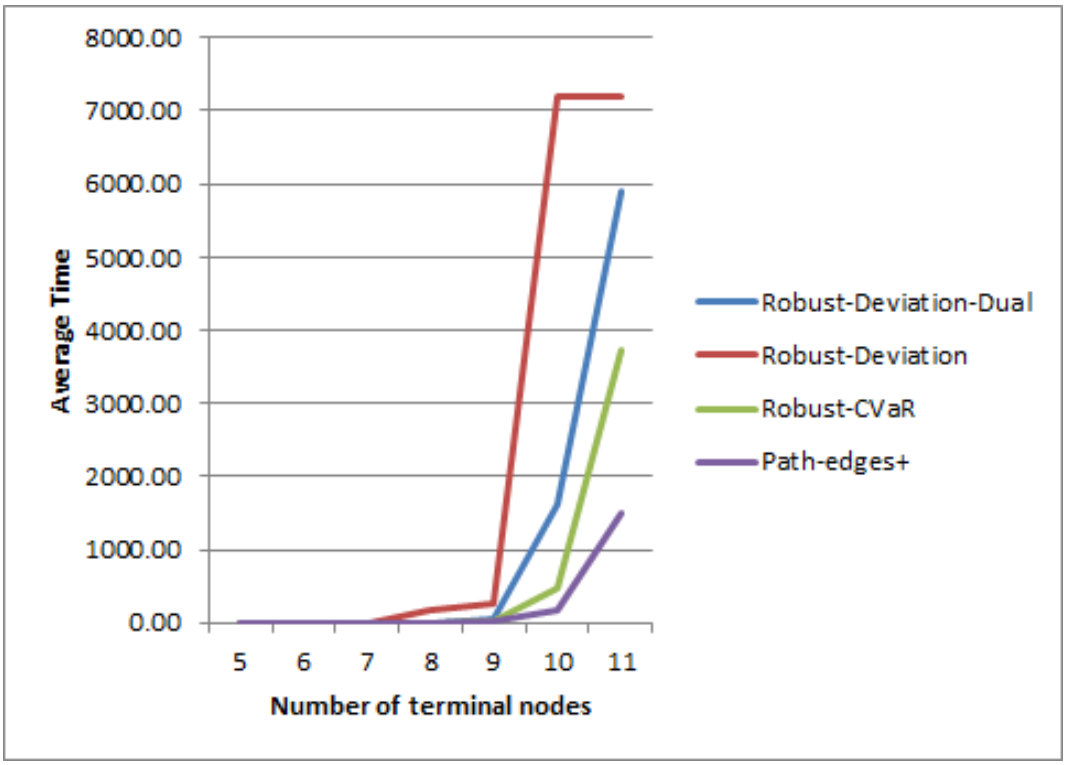


Figure 7.1: Average time of the robust formulations and the Path-edges$^+$ formulation.

Chapter 8

Software

In this chapter, we present a system developed, with the help of João Almeida, within the ambit of the degree in computer engineering [9]. This system displays the topology of an unknown network, using only packet delay measurements between the end-devices of the network that are obtained without the cooperation of the internal nodes. We start by presenting some of the software available for discovering a network topology. Then we present the requirements of the system we developed. Afterward we analyze, in more detail, the Precision Time Protocol, needed to synchronize the devices and present the implementation of our system. Finally, we present the results of the tests we made to evaluate our system.

8.1 Existing software

Nowadays there are several applications that discover and display the topology of a network. In this section, we present some of these applications, as well as some tools used to discover the Internet topology, since these tools could be adapted to determine the topology of any restricted network. All these tools use ICMP commands, such as ping and traceroute, and SNMP querys. As previously mentioned, these techniques require the cooperation of all internal network devices.

To the best of our knowledge, there are no tools available which discover the router-level topology without cooperation of the internal network devices.

In the first subsection we describe some commercial software and in the second subsection some tools used to discover the Internet topology.

8.1.1 Commercial Software

Here we present and describe some commercial software which discovers and displays the map of any network.

SolarWinds Network Topology Mapper

SolarWinds Network Topology Mapper [6] is a network mapping software that discovers and presents a map of the network.

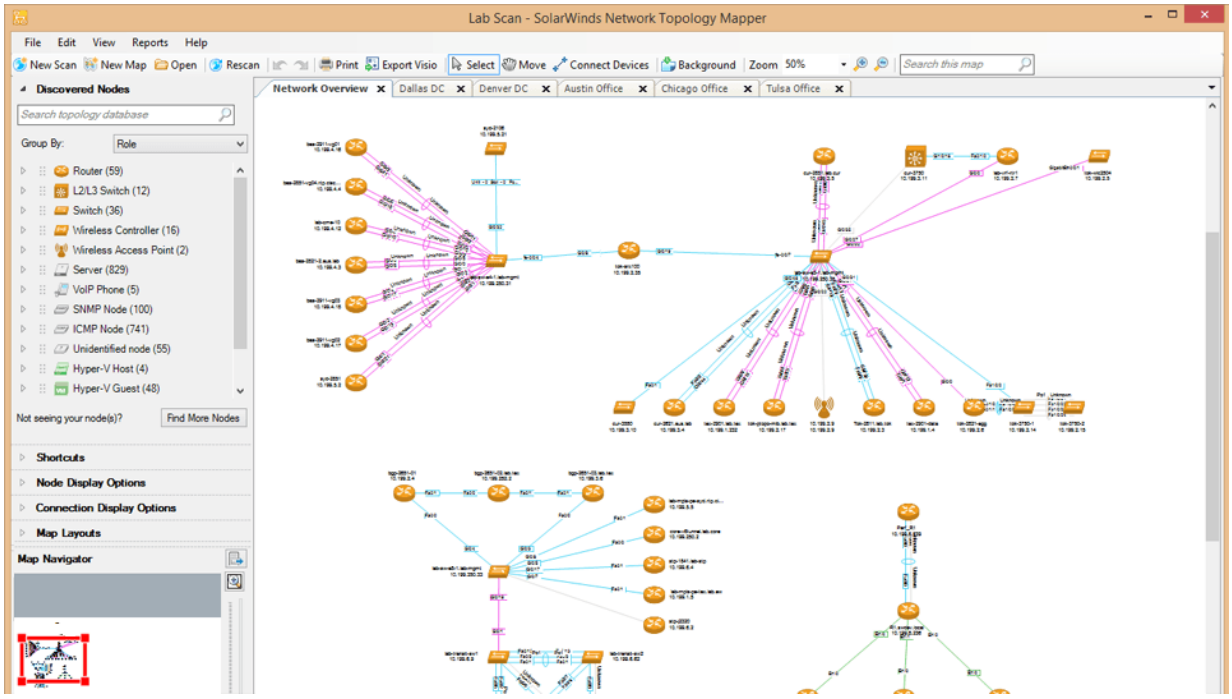


Figure 8.1: Interface of SolarWinds Network Topology Mapper.

This software uses ICMP, SNMP, CDP (Cisco Discovery Protocol), WIM (Windows Management Instrumentation), among other techniques to scan IP address ranges and find

nodes. It discovers and scans all the network segments connected to the users routers. The user can also enter the IP address range(s) and SNMP community strings of the network he wants to discover. The devices of the network that can be discovered are: routers, switches, servers, wireless access points (APs), VoIP phones, desktops, and printers.

After SolarWinds Network Topology Mapper discovers the network, it automatically displays a Layer 2 network map with node icons, coloured lines representing network connectivity speed and labels indicating the switch port information. The user can also select to view the Layer 3 network in order to display IP topology information. In the presented map the user can modify node details (node name, node role, and management IP address) and manually connect nodes.

Beyond discovering and displaying the network, this software allows the user to export network diagrams to Microsoft Office® Visio®, Orion Network Atlas, PDF, and PNG formats.

8.1.2 NetworkView

NetworkView [4] is a compact network discovery tool for Windows which can be run from a USB key.

This tool discovers TCP/IP nodes and routes using SNMP, Ports, NetBIOS and WMI. The search can be made using a single address, range of addresses or a full subnet.

The user can add nodes and routes manually and edit them. He can, also, save a complete map as a EMF (Enhanced MetaFile), which is a vectorial file type that can be modified with an external graphic application.

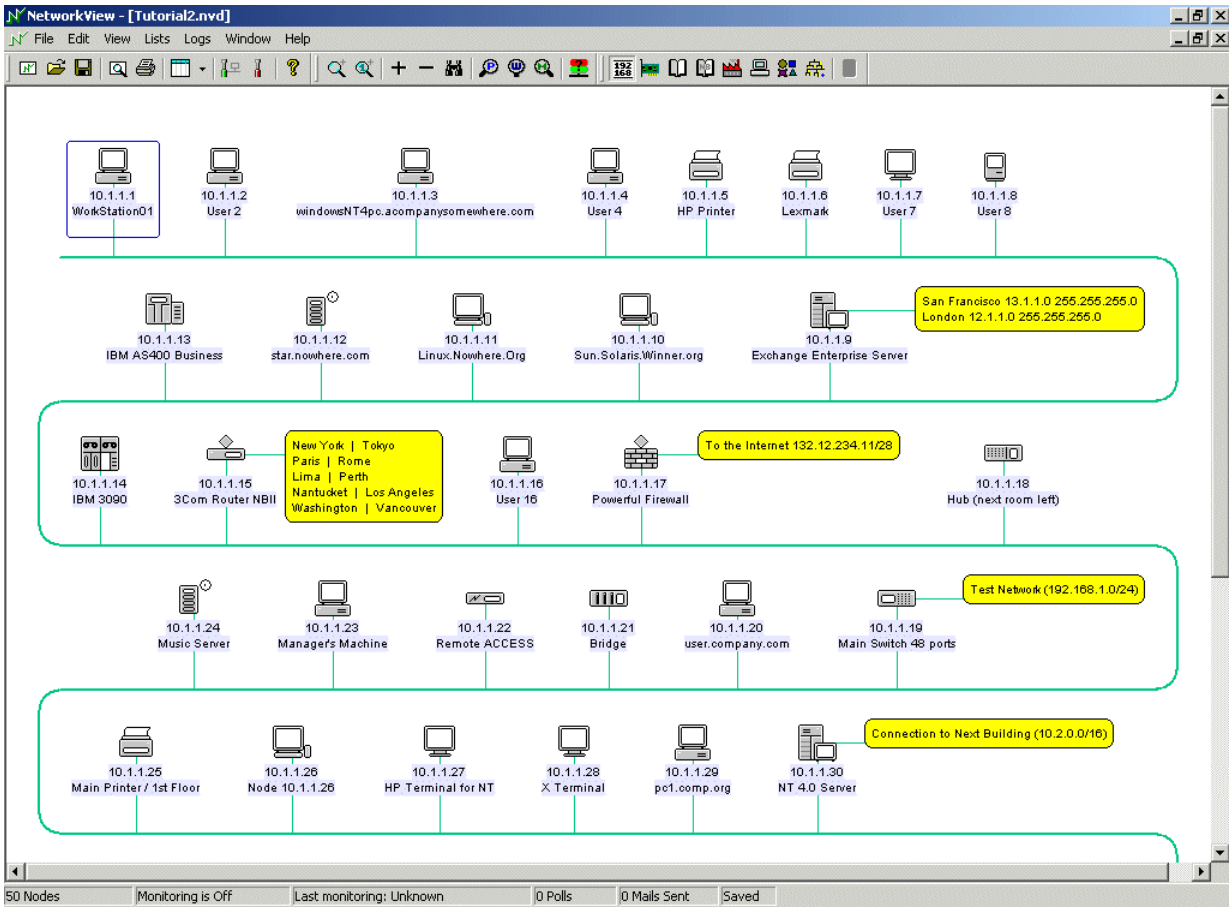


Figure 8.2: Interface of NetworkView.

8.1.3 Ipswitch WhatsConnected

Ipswitch WhatsConnected [2] is a software that automates network discovery and dependency mapping.

This software uses ICMP, SNMP, WIM, among other techniques to build a picture of the Layer 2 and the Layer 3 network. The maps generated show the physical, logical and virtual connectivity, including IP and VLAN-specific information. The devices detected are: routers, switches, servers, wireless access points, printers, Hubs, IP Phones and VMware virtual machines.

The user can customize and manipulate the maps and can export them to Microsoft® Visio™ or PDF format.

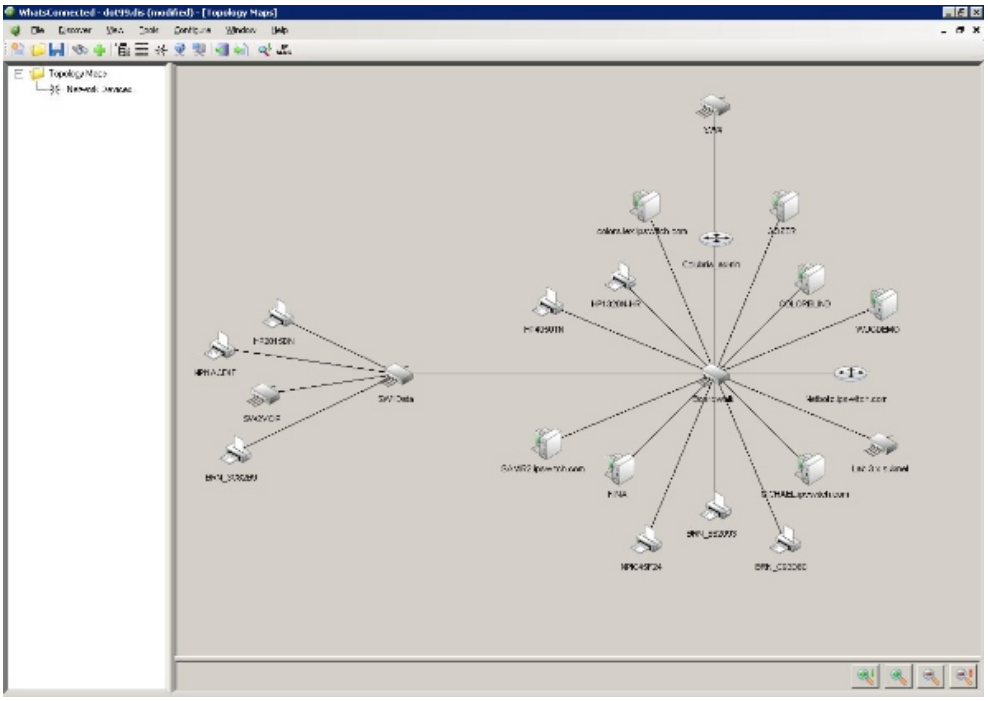


Figure 8.3: Interface of Ipswitch WhatsConnected.

8.2 Internet topology discovery

In the following, we present three projects that intend to discover the Internet topology.

8.2.1 Archipelago

For nearly a decade the project CAIDA (Center for Applied Internet Data Analysis) deploys and maintains a globally distributed measurement platform. They project started with a topology probing tool, the Skitter [84], which was replaced, in 2007, by a new active measurement infrastructure named Archipelago [33].

Physically, Archipelago is composed of measurement nodes (machines) located in various networks worldwide that are connected to a central server (at CAIDA) over Internet, forming a logical star topology. Using multiple teams of geographically distributed Archipelago monitors, the probing work among teams is dynamically and strategically divided to conduct a coordinated, large-scale traceroute-based topology measurements. The

monitors share a common set of destination addresses.

Archipelago uses parallel ICMP traceroutes from one source to many destinations, saving the replies from each router on the path and the round-trip times (RTTs).

CAIDA also has visualizing tools that presents a geographical map of traceroute paths.

8.2.2 DIMES

The project DIMES (Distributed Internet Measurements and Simulations) [120] is a scientific research project, which aims to discover the router-level topology of the Internet, with the help of a volunteer community. The volunteer community is formed by anyone who downloads and installs the so called DIMES agents on a computer connected to the Internet. This DIMES agents perform Internet measurements, such as traceroute and ping, at a low rate, consuming about 1KB/s, sending the results of its own measurements to a central collection station at regular intervals. The user who runs the DIMES agent has access to maps presenting the look of the Internet as it can be seen from his home.

8.2.3 Rocketfuel

Rocketfuel [132] is an ISP (Internet Service Provider) topology mapping engine developed in the University of Washington. The topology is discovered by using routing information and select the measurements that are most valuable. Rocketfuel uses traceroute, BGP (Border Gateway Protocol) routing tables to focus the measurements and DNS (Domain Name System).

8.3 The system's requirements

The system we developed measures the delays of packets sent between various devices, and determine the topology of the network that connects these devices. It consists of two independent applications that work cooperatively. The first application, which we call Synchronisation Network Application (SNA), synchronizes the devices using the Precision

Time Protocol (PTP), determines the delays and compiles these delays in a distance matrix. The second application, called Network Topology Application (NTA), determines the topology of the network using the distance matrix compiled by SNA and displays the topology in a graphical way. The two applications can be used independently. The SNA can determine the packets delays, saving the distances matrix in a file. The NTA can determine and display the topology of the network using the distance matrix previously saved in a file.

To measure the delays we must synchronize the clocks of the devices or use other techniques that do not need clock synchronization, like packet sandwich probe [23]. We decided to synchronize the clocks of the devices, since we consider this a more practical approach. As stated in Chapter 3 the synchronization can be done using several protocols, such as Network Time Protocol (NTP) [99] or Precision Time Protocol (PTP) [83]. Since PTP is considered to be more accurate [97], we choose to implement it and not the NTP. The PTP defines synchronization messages used between a Master and a Slave, where the Master is the provider of the time and the Slave synchronizes to the Master. We describe the Precision Time Protocol in more detail, as well as our implementation of it, in the next sections.

To determine the topology of the network knowing only the packets delays, we implemented the exact formulation Path-edges$^+$ presented in Chapter 5. Since the exact formulation take time to present a solution, we implemented the Feasibility Pump heuristic presented in Chapter 6, which determines an approximation of the topology very quickly. All the delays have an associated error, therefore we also implemented the Robust-Deviation-Dual formulation presented in Section 7.1. We did not implement the Robust-CVaR formulation presented in Section 7.2 once it was developed more recently after we create the system and we did not have the time nor the access to the university's network lab to run the tests.

The SNA has two types of users, the one we call the master and the one we call the slave. The master has access to all the features of the application and this device must have a graphical desktop environment. The slave does not have direct access to the application

since it runs in background. Concerning the NTA, all users have access to the application, but in order to access the SNA through NTA, they must be a master. Unlike the devices running as master, the devices running as slaves do not need to have a graphical desktop environment, but in this case they cannot use the NTA.

All devices must have an internet access via wired Ethernet and have the wireless connection turned off, so that the right IP address is sent in the packets. Beside this, all devices must have *Linux* and *Java 8* installed and the devices running NTA must also have *Gurobi* (a solver for mathematical programming) installed. Initially, we wanted to develop a system that should run on both Windows and Linux, but due to the Windows security restrictions we were unable to run the feature to synchronize the devices in Windows.

8.3.1 Requirements list

In the following we present the list of requirements of both applications, the SNA and the NTA.

Synchronisation Network Application

REQ. 1: Users shall have the ability to choose between running the application as master or slave.

REQ. 2: The application in master mode shall have a security system that is based on a password stored in an encrypted file.

REQ. 3: The application shall synchronize all the devices in the network.

REQ. 4: The application shall measure the packets delays and summarize these delays in a distance matrix.

REQ. 5: The application shall save the distance matrix in a file.

REQ. 6: The application shall read a distance matrix from a previously saved file.

REQ. 7: Users shall have the ability to identify the IP multicast address, which will be used to synchronize and measure the packets delays.

REQ. 8: Users shall have the ability to edit a distance matrix and change it.

REQ. 9: Users shall have the ability to manually create a distance matrix, introducing the devices names and the packets delays.

REQ. 10: The application shall have direct access to the Network Topology Application.

Network Topology Application

REQ. 11: The application shall determine and display the topology of a network using a distance matrix.

REQ. 12: Users shall have the ability to choose between the available formulations, which can be used to determine the topology.

REQ. 13: The application shall save the topology in a file.

REQ. 14: The application shall read the topology from a file and display it.

REQ. 10: The application shall have direct access to the Synchronisation Network Application.

8.3.2 Use-case diagrams

In this subsection, we present the use-case diagrams of the two applications and briefly describe each use-case.

Synchronisation Network Application

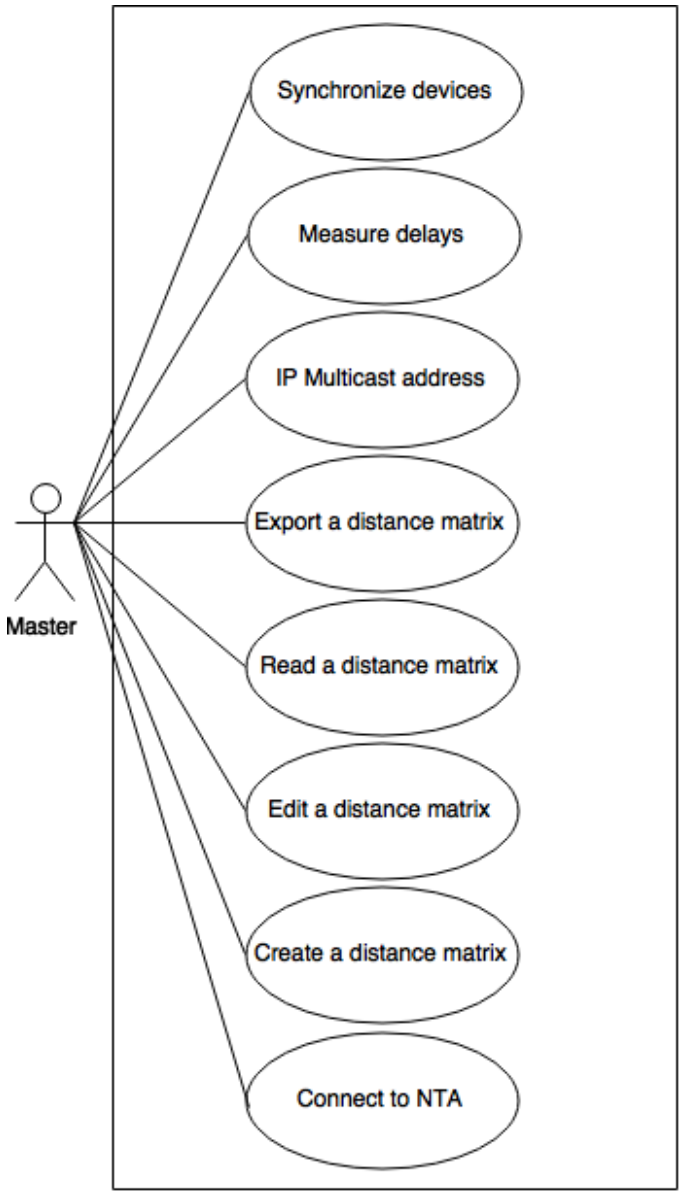


Figure 8.4: Use-case diagram of the Synchronisation Network Application.

- **Synchronize devices:** Allows the master device to start the Network Time Protocol so that the slave devices can synchronize their clock time to the master's clock time.

  Precondition: At least one slave device must respond to the master's request to synchronize.

  Postcondition: All the slave devices have their clock time synchronized with the master clock time.

- **Measure delays:** Allows all devices in the multicast session to send packets and to measure the packets delays. Before measuring the delays, all devices are synchronized. When all the slave devices have their delay, they send them to the master device, which saves them in a distance matrix and displays it on the screen.

  Precondition: At least three slave devices must participate in the multicast session.

  Postcondition: The master device displays the distance matrix.

- **IP Multicast address:** Allows the user to introduce the IP multicast address that is used to send the packets to synchronize the devices and measure the delays.

  Precondition: The IP address introduced must be a valid multicast address.

- **Export a distance matrix:** Allows to save the distance matrix, displayed on the screen, in a file. The location where the file is saved is indicated by the user, as well as the name of the file.

  Precondition: A distance matrix is displayed on the screen; the name of the file is valid; the location to save the file is valid.

  Postcondition: The file is saved in the location indicated by the user.

- **Read a distance matrix:** Allows the user to indicate the location of a file, which contains a distance matrix that was previously saved. The distance matrix is then displayed on the screen.

Precondition: The location of the file is valid; the file is a file containing a distance matrix.

Postcondition: The distance matrix is displayed on the screen.

- **Edit a distance matrix:** Allows the user to change the distance matrix displayed on the screen. The user can change the displayed distances, rename devices and delete all entries corresponding to a device.

  Precondition: A distance matrix is displayed on the screen; the changes introduced by the user are valid.

  Postcondition: The distance matrix is displayed on the screen.

- **Create a distance matrix:** Allows the user to create a distance matrix by introducing the devices names and the corresponding distances.

  Precondition: The data introduced by the user is valid.

  Postcondition: The data introduced by the user is displayed on the screen.

- **Connect to NTA:** Allows the user to open the Network Topology Application.

  Postcondition: Opens the Network Topology Application.

Network Topology Application

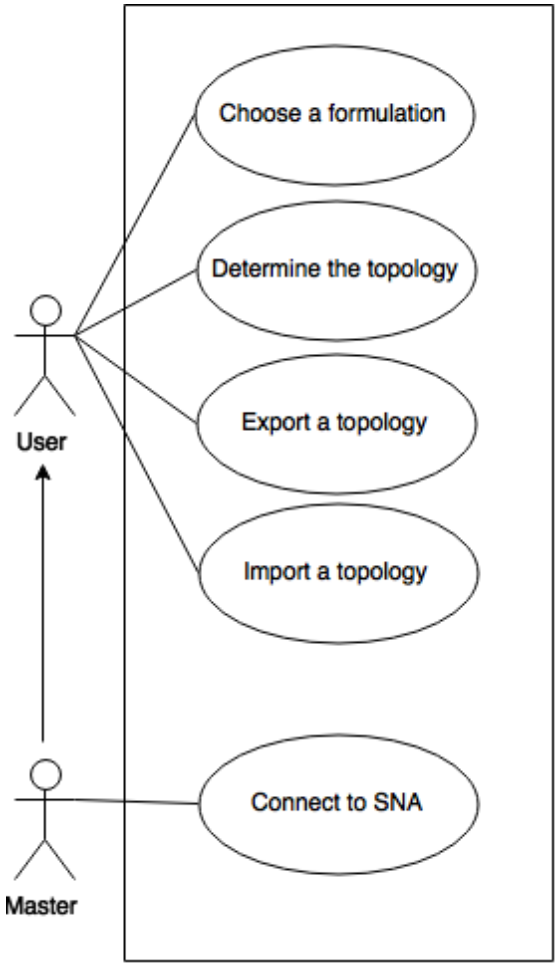


Figure 8.5: Use-case diagram of the Network Topology Application.

- **Choose a formulation:** Allows the user to select the formulation the system uses to determine the topology.

  Postcondition: The system registers the formulation that was selected.

- **Determine the topology:** Allows to determine the topology of a network using a file that contains the distance matrix. The location of the file is given by the user. If the user does not previously select a formulation, the system uses the exact formulation to determine the topology of the network. The network topology obtained is displayed on the screen.

Precondition: The location of the file is valid; the file is a file with a distance matrix.

Postcondition: A network topology is displayed on the screen.

- **Export a topology:** Allows to save the network topology, displayed on the screen, in a file. The location where the file is saved is indicated by the user, as well as the name of the file.

  Precondition: A network topology is displayed on the screen; the name of the file is valid; the location to save the file is valid.

  Postcondition: The file is saved in the location indicated by the user.

- **Import a topology:** Allows the user to indicate the location of a file that contains a previously saved topology, previously saved. The topology is then displayed on the screen.

  Precondition: The location of the file is valid; the file is a file containing a topology.

  Postcondition: The network topology is displayed on the screen.

- **Connect to SNA:** Allows the user to open the Synchronisation Network Application.

  Precondition: The user must be a master.

  Postcondition: Opens the Synchronisation Network Application.

8.4 The Precision Time Protocol (PTP)

In this section we present the Precision Time Protocol [83, 123, 137] in more detail to better understand the implementation of our system.

The Precision Time Protocol is one of the common protocols used to synchronize clocks in packet networks. This protocol defines messages between a master, which provides the time, and a slave, which uses the messages to correct its clock and synchronize to the master's time.

The PTP operates in two phases. The first phase is used to correct the time difference between the master and the slave. In this phase, the master sends a synchronization message (SYNC message), with its time information to the slave. Then the master registers the time at which the SYNC message was sent and sends this time in a Follow-up message. The slave registers the reception time of the SYNC messages, calculates the difference between the reception time and the time sent by the master and updates its clock time.

In the second phase, the slave determines the network delay. To do that, the slave sends a Delay Request message and registers the time instant when the message leaves. The master registers the time instant when the Delay Request message arrives and sends this time in a Delay Response message to the slave. The slave, knowing the difference between these time values (designated Slave to Master difference), calculates the one way delay and updates its clock time. It is assumed that the delay between master and slave is symmetric and so the one way delay is equal to $\frac{\text{Master to Slave difference} + \text{Slave to Master difference}}{2}$, where the Master to Slave difference is considered to be zero, since the slave has updated his time in the first phase in order to be equal to the master's time.

In Figure 8.6 we display an example of the messages that are exchanged by the master and the slave. As we can see, after the slave receives the SYNC message and the Follow-up message, it updates his time. After receiving the Delay Response message, the slave calculates the one way delay and updates its time again.

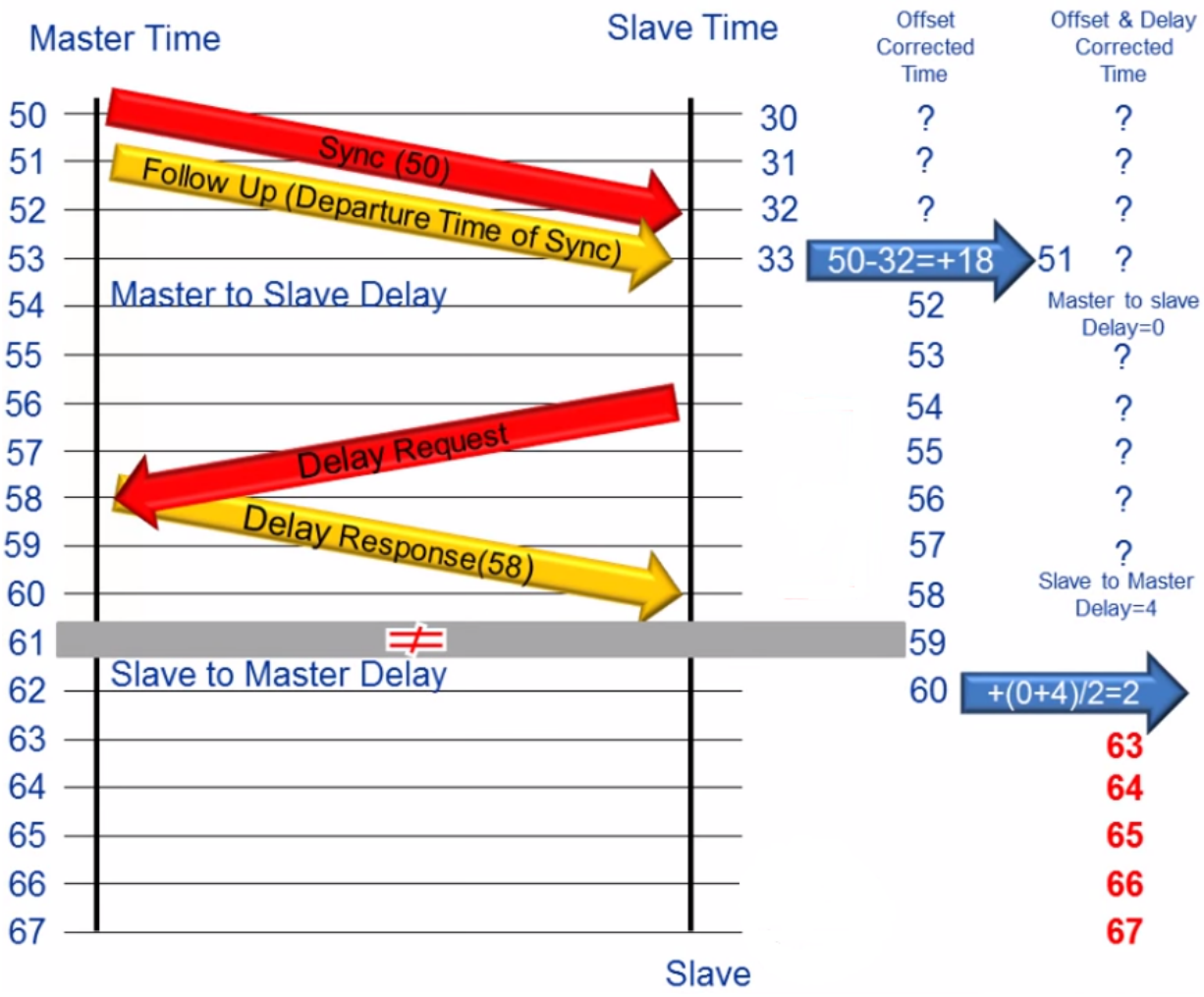


Figure 8.6: Example of PTP messages exchange [7].

8.5 Implementation

In this section we describe our implementation. We start by describing our implementation of the Precision Time Protocol (PTP), then we describe how we measure the delay. We present the way we present the topology on the screen and, finally, we describe the interface of both applications.

8.5.1 Implementation of the Precision Time Protocol

As we referred in the previous section, the PTP relays on messages containing sending times between the master and the slave. So, the more precise the sending and receiving times are, the more precise is the synchronization. Therefore, most of the PTP implementations use specialized hardware, but there are also implementations that do not use specialized hardware and are referred to as software-only implementations [39]. Our implementation of the PTP is a software-only implementation. Throughout the implementation, to be as much precise as possible, all times are read and saved in nanoseconds.

We use multicast packets and unicast packets in our implementation of the PTP. In the first phase of PTP we use multicast packets and in the second phase we use unicast packets. In the first phase the master sends a synchronization message, designated *Sync*, and a follow-up message, designated *SendTimeSync*. The master sends this pair of messages four times more, in order to ensure that all slaves receive this pair of messages. After sending a pair of messages, the master waits 0,5 seconds before sending the next one.

The *Sync* packet has the following composition:

- a flag indicating the type of the packet, in this case indicating that it is a synchronization message;
- the session id, a integer number between zero (inclusive) and 9999 (exclusive);
- the packet id ;
- the IP address size;

- the IP address.

Figure 8.7 displays the composition of a *Sync* packet.

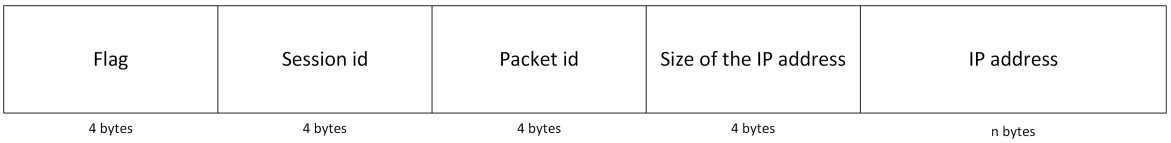


Figure 8.7: Composition of the *Sync* packet.

The *SendTimeSync* packet has the following composition:

- a flag, indicating that it is a follow-up message;
- the session id, which is the same as the session id of the Sync packet;
- the packet id, which is the same as the packet id of the Sync packet;
- the time, at which the Sync packet was sent.

Figure 8.8 displays the composition of a *SendTimeSync* packet.

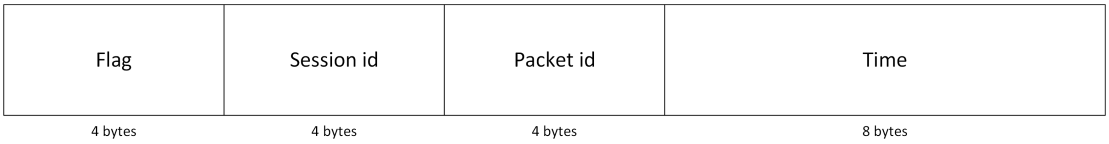


Figure 8.8: Composition of the *SendTimeSync* packet.

When the slave receives the *Sync* packet, it registers the current time and the IP address of the master. After receiving a *SendTimeSync* packet, the slave verifies if the session id and the packet id of this *SendTimeSync* packet correspond to the session id and packed id of the Sync packet received before. If the identification numbers do not correspond, the slave discards the packet and waits to receive another *Sync* packet. If the identification numbers correspond, the slave discards all the *Sync* packets and *SendTimeSync* packets it receives next. Furthermore, the slave saves the current time in variable *machineTime* and the time in the *SendTimeSync* packet is saved in variable *serverTime*. Since we cannot

change the System timer, variable *serverTime* will be used as the new current time and variable *machineTime* is used to update the *serverTime*. We use the following expression to update the *serverTime*: *serverTime* = *serverTime* + (current time – *machineTime*), where the current time is the real current time of the system.

To start the second phase of our implementation of the PTP, the slave waits for the master to send a *StartReadSlaves* message, which it does using a multicast packet, after sending the five pairs of packets (*Sync*, *Send Time Sync*). The *StartReadSlaves* packet is used to advise the slaves that they can start sending packets to register in a session, and has the following composition:

- a flag, indicating the type of message;
- the session id, which is the same as defined before in the Sync packet.

Figure 8.9 displays the composition of a *StartReadSlaves* packet.

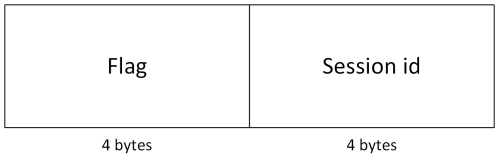


Figure 8.9: Composition of the *StartReadSlaves* packet.

Figure 8.10 illustrates the message exchange that takes place in the second phase of our implementation of the PTP. In this phase all the communication between the master and the slaves is done in unicast.

The *AdvertiseMaster* packet contains the IP address of the slave. The master creates a thread for each *AdvertiseMaster* packet received and saves all the IP addresses in a list.

After receiving the acknowledgement packet from the master, the slave sends a *DelayMeasures* packet and registers the time the packet leaves. The master answers with an acknowledgement packet that contains the time instance when the *DelayMeasures* packet arrived and his current System time. The slave registers the exact time the acknowledgement packet arrives. Then using the time the *DelayMeasures* packet arrives and the time

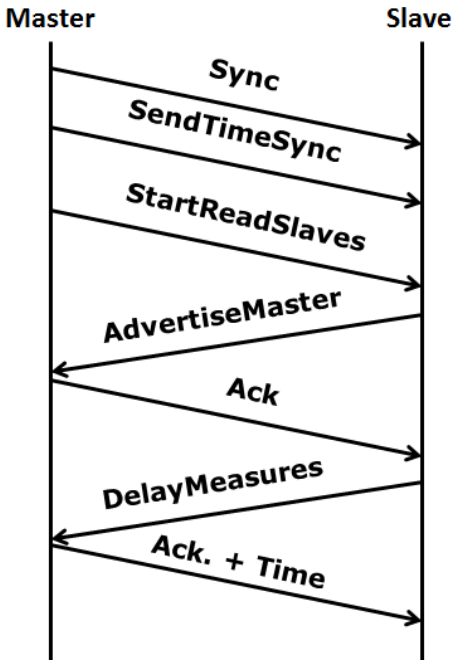


Figure 8.10: Message exchange that takes place in the second phase of our implementation of the PTP.

registered when the *DelayMeasures* packet leaves, *serverTime* is updated. It first calculates the delay, as follows:

$$Delay = \frac{\text{Time } DelayMeasures \text{ arrives} - \text{Time } DelayMeasures \text{ leaves}}{2}$$

and then updates the *serverTime*:

$$serverTime = serverTime + (\text{Slave's System current time - } machineTime) + \text{Delay.}$$

The server, also, calculates de following difference:

$$\text{Difference} = \text{The master's System current time sent in the acknowledgement packet} - (\text{Time the acknowledgement packet arrives} - \text{Delay}).$$

The slave continues to send *DelayMeasures* packets and updates its time as long as the difference is greater then 1000 nanoseconds.

After the synchronization process is complete, the process of measuring the packet delays and construction of the distance matrix starts. We describe it in the next section.

8.5.2 Construction of the distance matrix

The process of determining the packet delays is done by sending unicast packets and is managed by the master. All slaves wait for the authorization message (*SendPermission* packet), sent by the master, that authorizes them to send multicast packets to the other devices, in order to determine the distance between them and identify the device that is sending the packet. The first entity to send multicast packets is the master; when it finishes, it sends a *SendPermission* packet to all slaves one by one.

The device sending the multicast packets starts by sending ten *NewMulticastTree* packets, with a 3-seconds interval between two consecutive packets, which only contains a flag indicating the type of the packet and are discarded by the other devices. This allows to establish the new multicast distribution tree. Subsequently, the device sends 100 *SendDelays* packets, with a 0,05-seconds interval between consecutive packets.

The *SendDelays* packet has the following composition:

- a flag indicating the type of the packet;
- the IP address size;
- the IP address;
- the current time of the device.

Figure 8.11 displays the composition of a *SendDelays* packet.

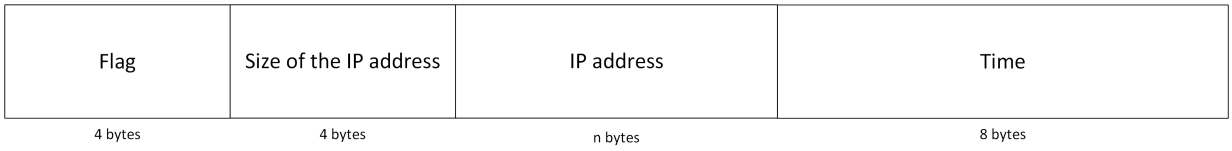


Figure 8.11: Composition of the *SendDelays* packet.

After a device receives a *SendDelays* packet, it registers the IP address of the sender and the time the packet arrives and then calculates the delay:
Delay = Time the *SendDelays* arrives – Time present in the*SendDelays* packet.

The device has then a list of delays and determines the distance to the sender by calculating the average of these delays.

After this process, the master sends a multicast packet (*AdvertiseSlavesToSendDevices*) to allow each slave to send the collected distances. The master compiles all the distances in a matrix and displays this matrix on the screen.

8.5.3 Determination and representation of the topology

With the distance matrix, the system determines the topology using one of the formulation we implemented. The system may use the exact formulation Path-edges$^+$ formulation presented in Chapter 5, the Feasibility Pump heuristic presented in Chapter 6 and the robust approach presented in Section 7.1 of Chapter 7.

To represent the topology on the screen, some calculations are made, in order to uniformly distribute the devices on the screen. To do this, devices will be positioned in a ellipse.

We start by determining the coordinates of the center of the panel and the position of each device relatively to this center. To determine the position of each device relatively to the center, we must determine the angle between two devices and the distance of all devices to the center, which is different for each device since devices are positioned in an ellipse. The angle between two devices is $\frac{2\pi}{n}$, where n is the number of devices.

Let (a, b) be the coordinates of the center. The first device will be positioned in the horizontal line passing through the center, that is, the ordinate of the position of the device is also b. To position the other devices we will use the angle, θ between the device to be positioned and the first device. This angle is calculated using the following expression:

$$\theta = \frac{2\pi(k-1)}{n}, \quad \text{where } n \text{ is the number of devices and } k \text{ is the order of the device.}$$

So, for the first device we have: $\theta = 0$.

Using the angle θ, we can determine the distance, d, between the center and the position

of a device using the following expression:

$$\mathrm{d} = \frac{a^2 b^2}{b^2 + (a^2 - b^2) sin^2(\theta)}$$

The coordinates of the device in the panel are determined using the following expression:

$$(a + d \cdot cos(\theta), b + d \cdot sin(\theta)) .$$

The position of the routers is determined in a similar way, but we reduce the distance to the center by subtracting a constant.

8.6 The Interface

In this section we briefly describe the graphical interface of both applications.

8.6.1 Synchronisation Network Application (SNA)

Login/Registration

When the user enters the application, the login/registration windows displayed in Figure 8.12 is presented.

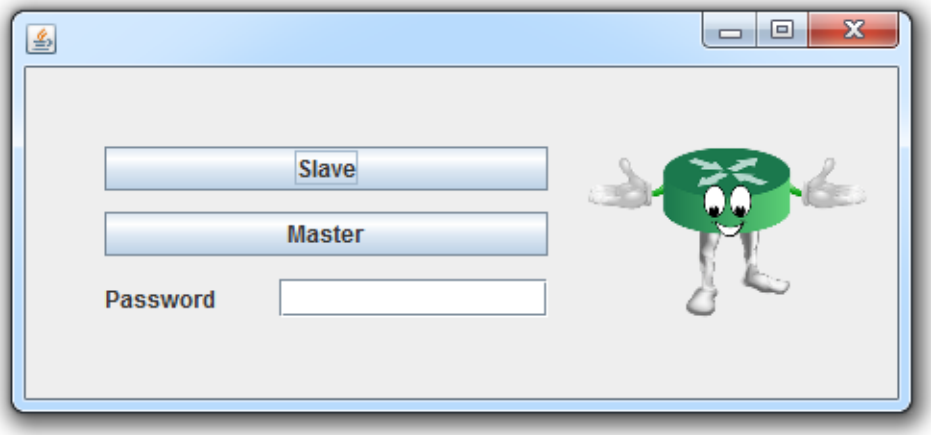


Figure 8.12: Login/registration window of the SNA.

The button *Slave* allows the user to enter in slave mode. After entering in slave mode, the application shows a window, displayed in Figure 8.13, where the user is asked to enter

the IP multicast address used to send multicast packets. The system will verify if the introduced IP multicast address is valid. The multicast address is saved in a file. If the user clicks the *OK* button without indicating the IP multicast address, the system will read the address from the file, if this file exists. If the file does not exist and the user clicks the *OK* button without indicating the IP multicast address, the system shows an error message.

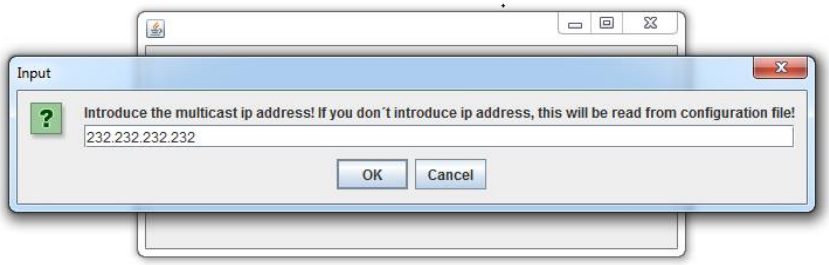


Figure 8.13: The window for entering the IP multicast address, in slave mode.

Button *Master* allows the user to enter in master mode. To enter as master, the user must insert the password. The password is saved in an encrypted file. When no file exists the user is asked to confirm the password.

After entering the application as master, the user is presented with the main window displayed in Figure 8.14.

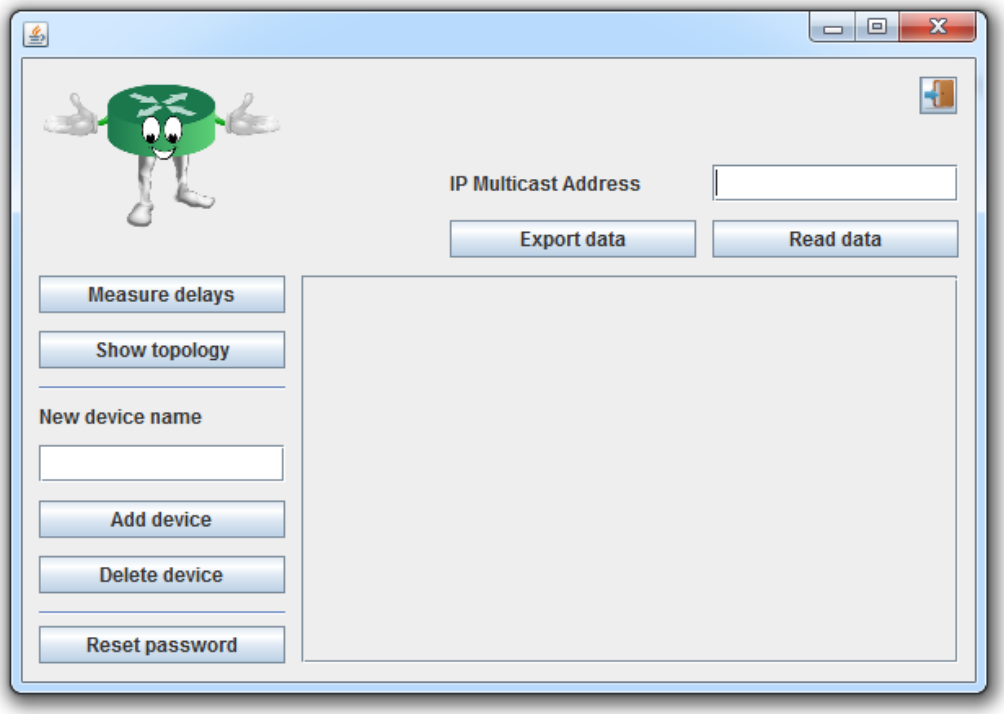


Figure 8.14: Main window of the SNA.

To determine the distance matrix of a network, the user must enter the IP multicast address that is used to send the packets. The system will verify if the introduced IP multicast address is valid.

Button *Export data* allows the user to save the distance matrix presented on the screen in a file.

Button *Read data* allows the user to read a file that contains a distance matrix and displays it on screen.

Button *Measure delays* allows the user to determine the distance matrix of a network, by synchronizing the devices and obtaining the delays. After the distance matrix is determined, it is displayed on the screen as shown in Figure 8.15.

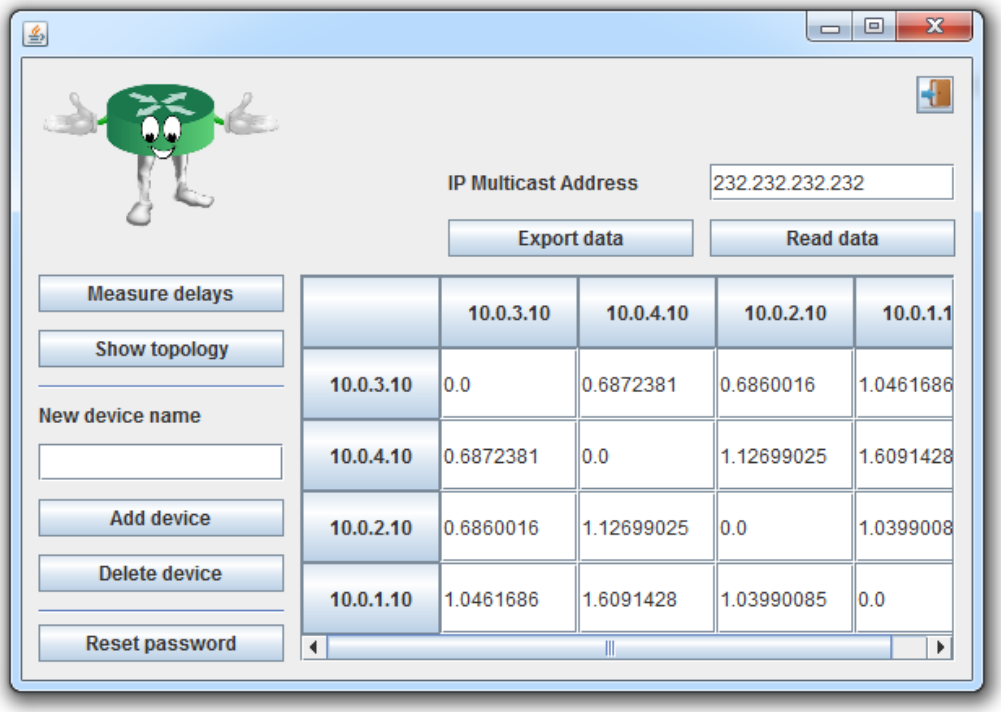


Figure 8.15: Main window of the SNA with a distance matrix displayed on screen.

Button *Show topology* gives the user direct access to the Network Topology Application (NTA). If a matrix is displayed in SNA, when the NTA opens, the topology corresponding to the matrix, is automatically displayed on screen. If no matrix is displayed in SNA, when clicking in the *Show topology* button, the NTA simply opens.

Button *Add device* allows the user to add a device to the matrix; to do so, the user must indicate the device name.

Button *Delete device* allows the user to delete a device from the matrix, to do so, the user must indicate the device name.

The button *Reset password* deletes the content of the encrypted file containing the password. If the user wants to change his password, he deletes the content of the encrypted file and the next time he enters the application, he chooses the new password.

To change the value of the distances shown in the matrix, the user simply clicks on the value and changes it.

8.6.2 Network Topology Application (NTA)

The NTA can be opened by any user, therefore the application only has a main window as displayed in Figure 8.16.

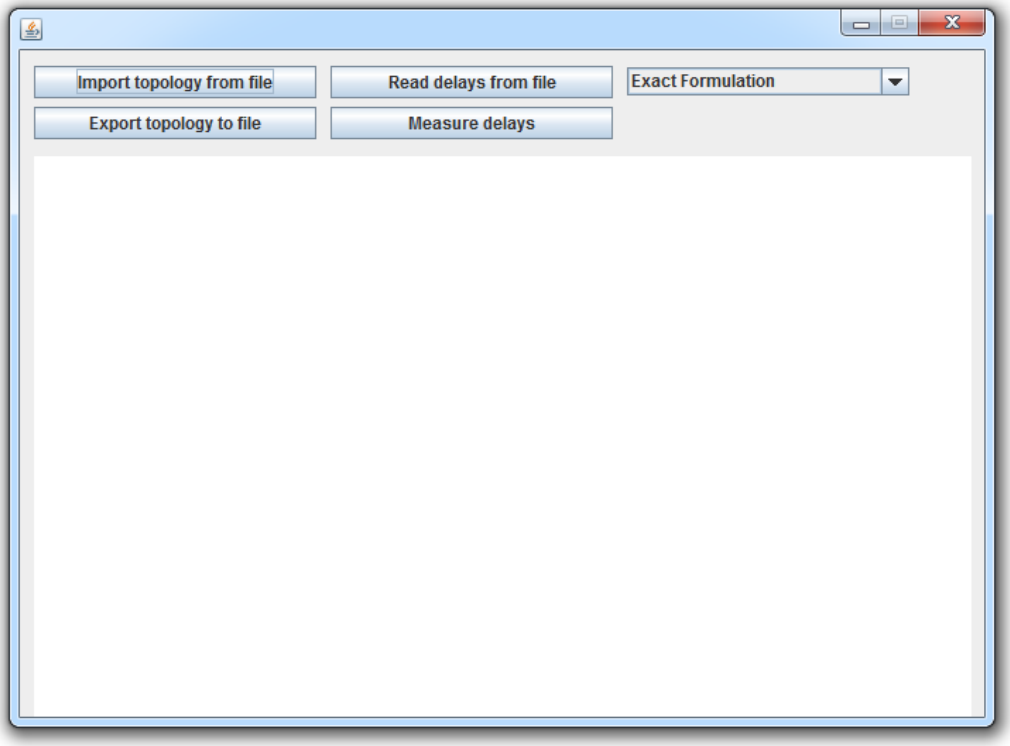


Figure 8.16: Main window of the NTA.

Button *Import topology from file* displays on the screen a topology saved in a file.

Button *Export topology to file* allows to save the topology, displayed on the screen, in a file.

Button *Read delays from file* allows to read the distance matrix from a file and displays the corresponding topology on the screen.

The dropdown button allows the user to choose the formulation that the system will use to determine the topology. The user can choose between three strategies: Exact formulation, Robust formulation and Feasibility Pump. By default, the topology is determined using the Exact Formulation. If a topology is displayed on the screen, and the user chooses another formulation, the system determines the new topology using the chosen formulation

and displays it on the screen.

Button *Measure delays* gives the user direct access to the login window of the SNA.

In Figure 8.17, we can see the main window of the NTA with a topology displayed on the screen.

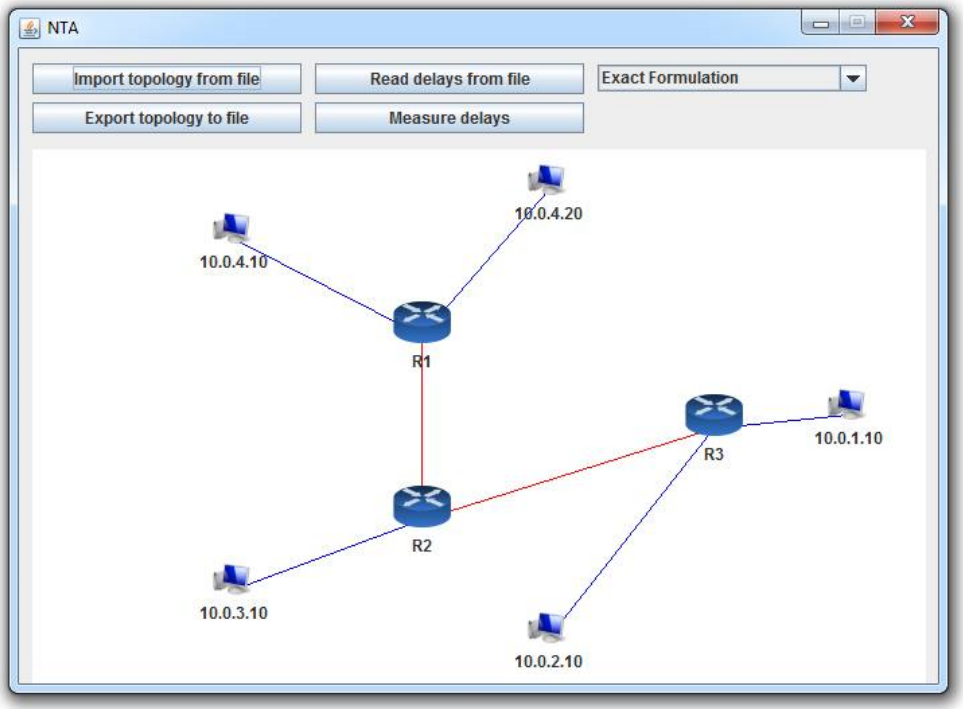


Figure 8.17: Main window of the NTA with a topology displayed on screen.

8.7 Tests and Results

Initially, we wanted to test our system using the GNS3 simulator. So, we created a network using GNS3 and tried to use virtual machines to simulate the hosts, but due to the memory and processing limitations of our computers, we could not connect more than three virtual machines simultaneously. Therefore, we tested our system using the university network lab. We set up a network with four, five and six computers. The university network lab only has six functional computers, so we were limited on the size of the network. We tested our system several times using networks with four, five and six

computers.

In the following, we present the physical topology of the networks we created and the logical topology the system displayed on the screen, after determining the distance matrix through the delay measurements and running each of the three formulations.

8.7.1 Network with four computers

The network we created with four computers is illustrated in Figure 8.18.

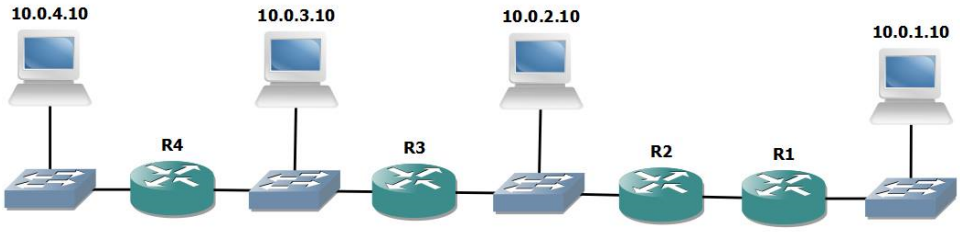


Figure 8.18: Physical network with four computers.

After we run the application, using the three formulations, we obtained the logical topologies that are shown in Figure 8.19, Figure 8.20 and Figure 8.21.

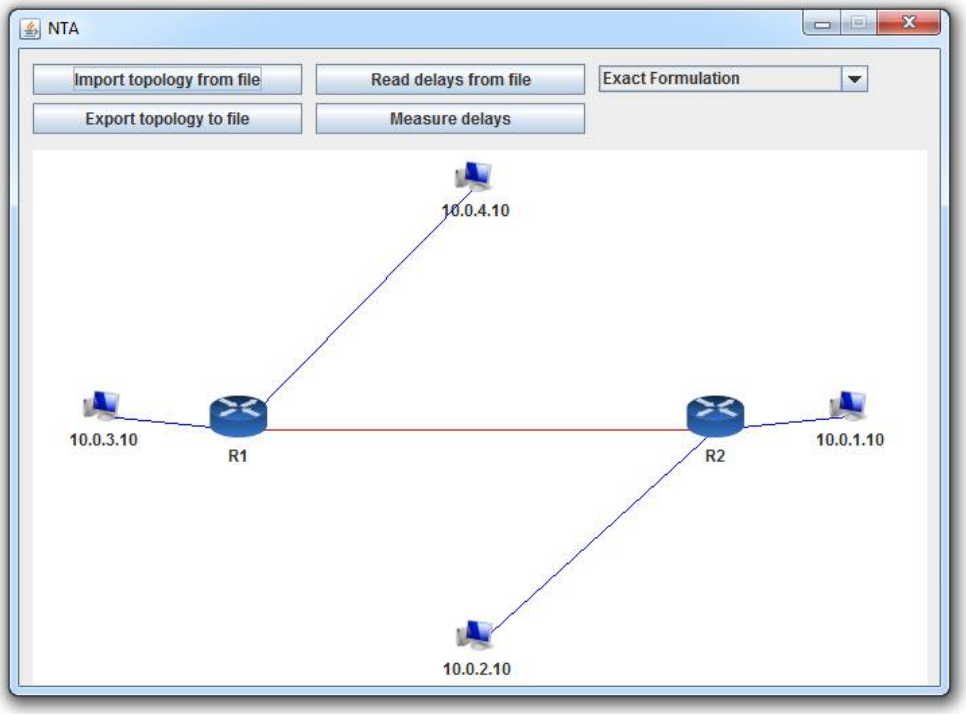


Figure 8.19: Logical network with four computers obtained using the Exact Formulation.

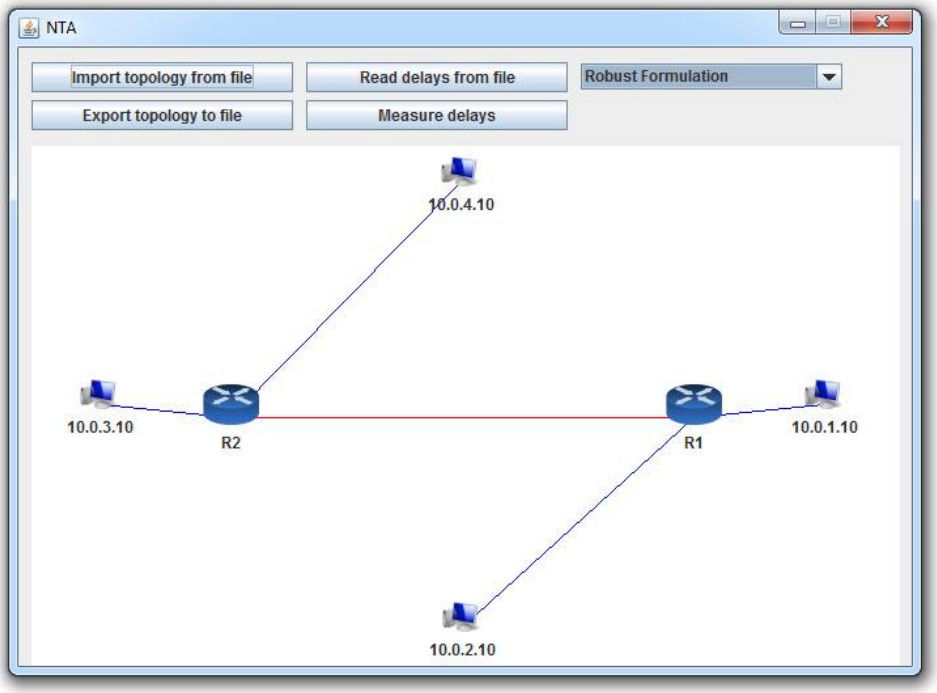


Figure 8.20: Logical network with four computers obtained using the Robust approach.

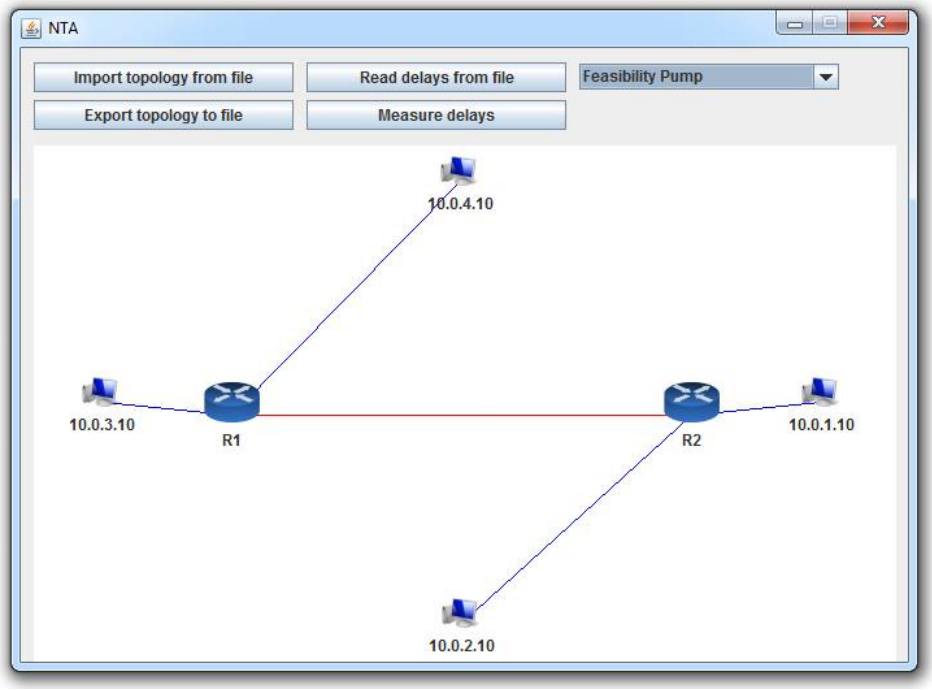


Figure 8.21: Logical network with four computers obtained using the Feasibility Pump Heuristic.

As we can see, the three formulations obtained the same logical topology, despite the fact in the logical topology obtained using the Feasibility Pump Heuristic both routers are swapped. However, as we referred in the previous chapter, the internal nodes, which correspond to the router, can be arbitrarily interchanged without modifying the tree topology.

We can also notice that the logical topology is in accordance with the distribution of the computers in the physical network, since the computer 10.0.4.10 is closer to the computer 10.0.3.10 than to computer 10.0.1.10.

8.7.2 Network with five computers

The network we created with five computers is illustrated in Figure 8.22.

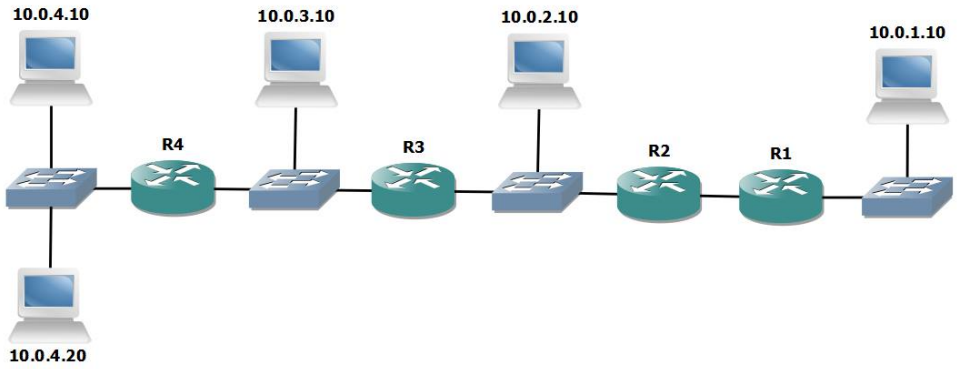


Figure 8.22: Physical network with five computers.

After we run the application, using the three formulations, we obtained the logical topologies that are shown in Figure 8.23, Figure 8.24 and Figure 8.25.

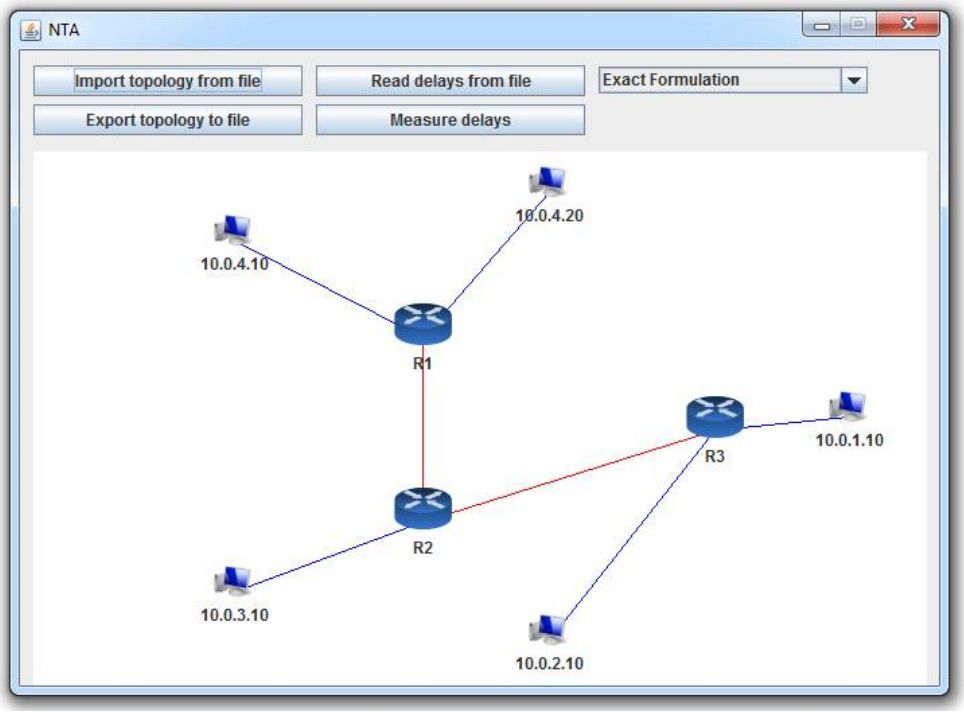


Figure 8.23: Logical network with five computers obtained using the Exact Formulation.

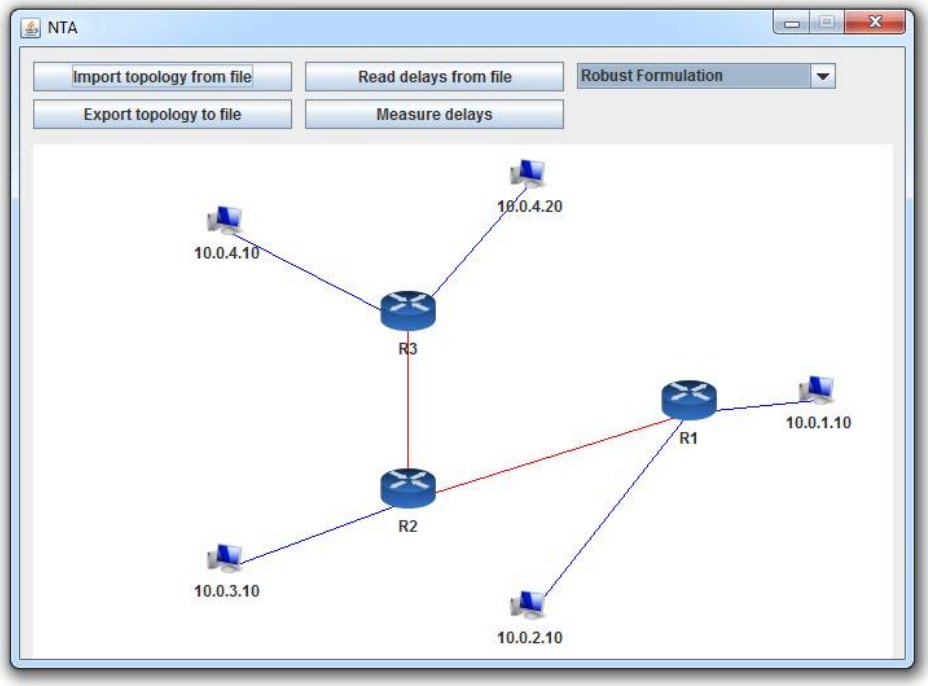


Figure 8.24: Logical network with five computers obtained using the Robust approach.

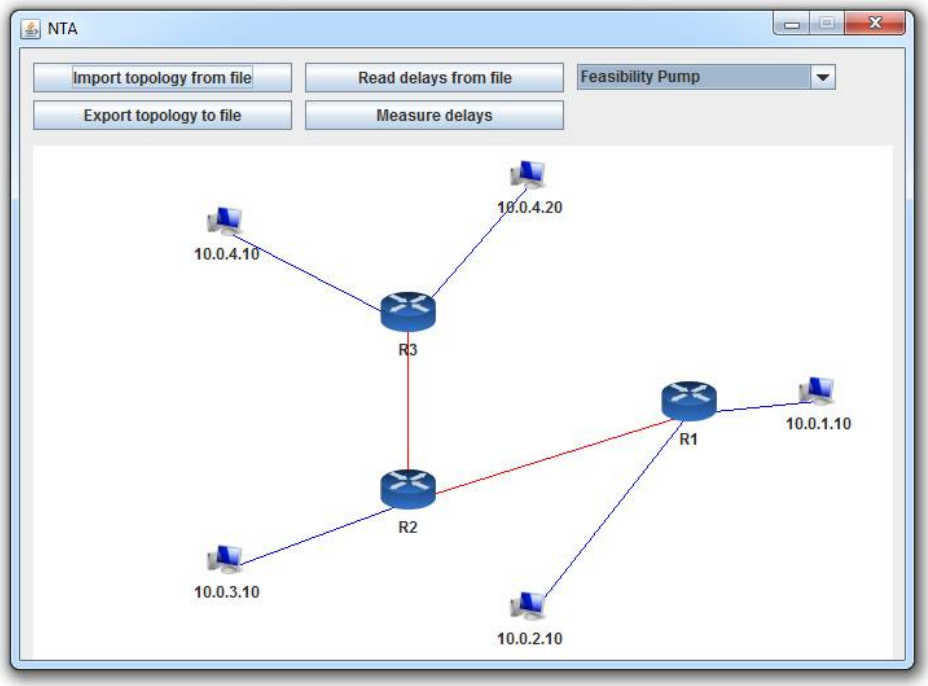


Figure 8.25: Logical network with five computers obtained using the Feasibility Pump Heuristic.

As we can see, also in this case the three formulations obtained the same logical topology and the logical topology is in accordance with the distribution of the computers in the physical network.

8.7.3 Network with six computers

The network we created with six computers is illustrated in Figure 8.26.

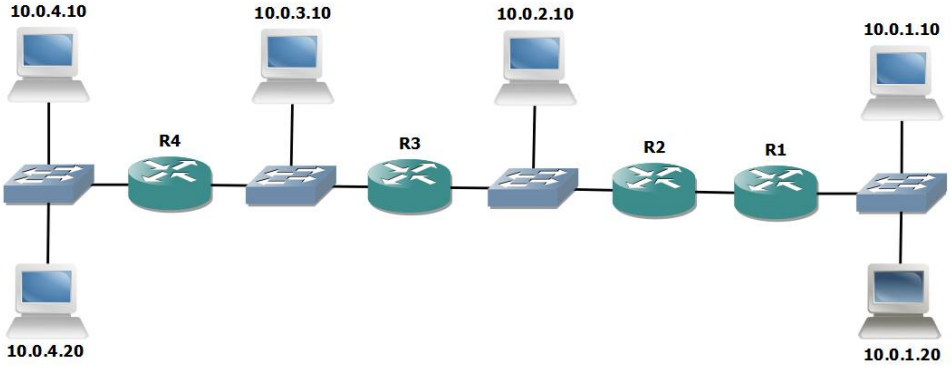


Figure 8.26: Physical network with six computers.

After we run the application, using the three formulations, we obtained the logical topologies that are shown in Figure 8.27, Figure 8.28 and Figure 8.29.

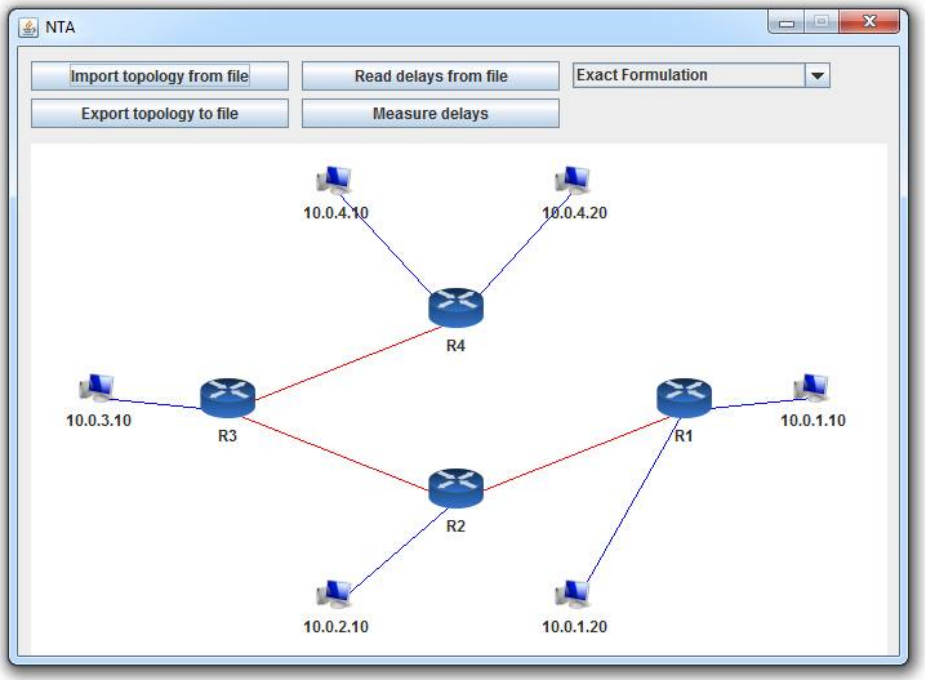


Figure 8.27: Logical network with six computers obtained using the Exact Formulation.

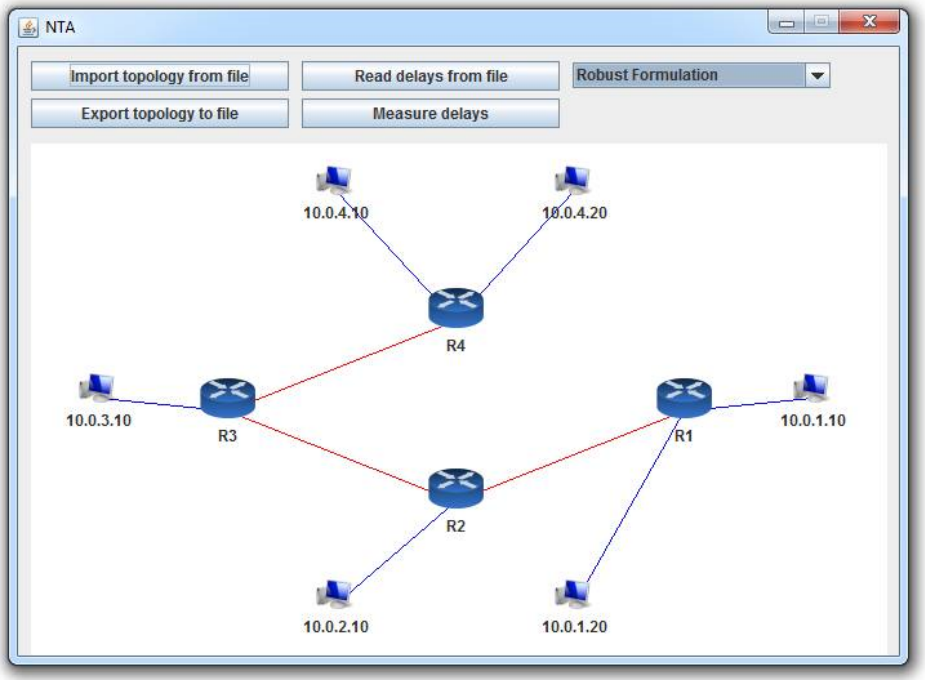


Figure 8.28: Logical network with six computers obtained using the Robust approach.

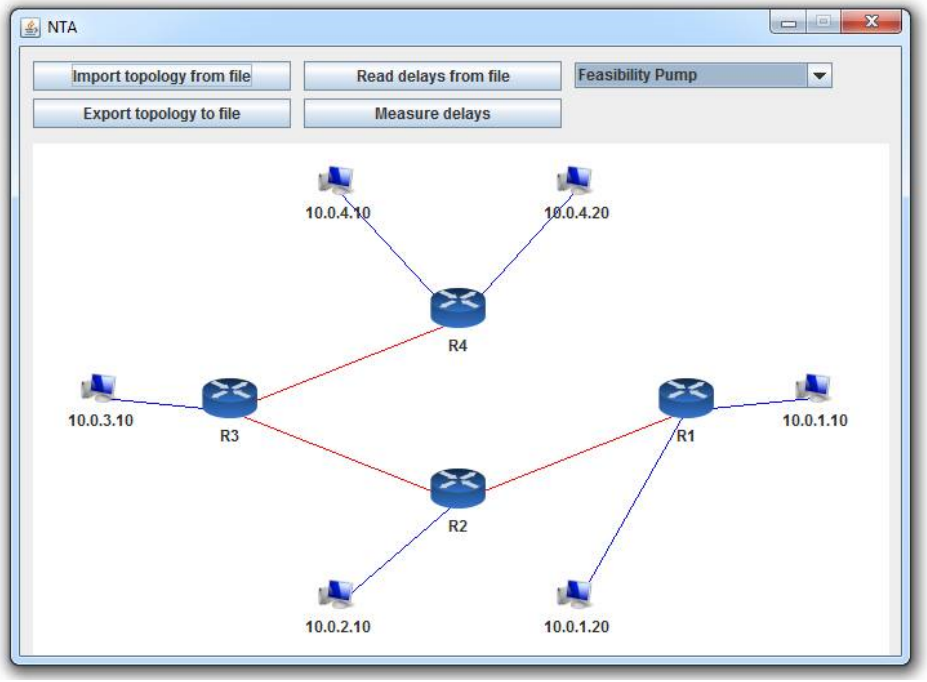


Figure 8.29: Logical network with six computers obtained using the Feasibility Pump Heuristic.

As we can see the three formulations obtained the same logical topology and the logical topology is in accordance with the distribution of the computers in the physical network.

8.7.4 Observations

The results of the tests we made show that the system is functional. We only regret the fact that we could not perform tests using a larger number of computers. It would have been interesting to see the system behavior in these cases, namely the processing time spent by the system.

Because the number of computers on the network is reduced, it took longer to determine the distances matrix than run the formulations. With a larger number of computers in the network, this would not be the case, namely using the exact formulation and the robust approach.

The reduced number of computers on the network is also the reason why the Feasibility

Pump heuristic obtained the same topology as the other two formulations. As the number of computers in the network grow, the topology obtained with the Feasibility Pump heuristic and the topology obtained with the other two formulations would be different, because in this case the heuristic only finds a feasible solution and not the optimal one.

Chapter 9

Conclusion

In this thesis we defined formally the Minimum Weighted Tree Reconstruction (MWTR) problem. We introduced the concepts related to the inference of a network topology and presented the instances of the problem we generated using NS-3, a network-level simulator. We also described some concepts related to the inference of the phylogenetic tree and present the instances of the phylogenetic area we used.

Subsequently, we presented two compact mixed integer linear programming formulations of the MWTR problem, the Path-weight formulation and the Path-edges formulation. By including valid equalities and inequalities to improve the performance of the second formulation we obtained the Path-edges^{+} formulation and by including only valid equalities we obtained the Path-edges^{+2} formulation. We also present the computational results obtained by running the several formulations when using data instances from networking application and phylogenetic application. The best results were obtained by the Path-edges^{+} formulation.

We also presented the methods Feasibility Pump and Local Branching and the two heuristics, the Feasibility Pump heuristic and the Local Branching heuristic, we developed applying the ideas of these methods. The computational results obtained by running the two heuristics when using the data instances from networking application and phylogenetic application were presented. The Feasibility Pump heuristic finds a feasible solution very

quickly but the quality of the solution is not the best. The Local Branching heuristic significantly improves the feasible solution obtained by the Feasibility Pump heuristic.

Then, we studied two robust approaches to solve the MWTR problem, one to control the maximum number of deviations and the other to reduce the risk of high cost. We presented the three formulations, the Robust-Deviation-Dual formulation, the Robust-Deviation formulation and the Robust-CVaR formulation, we derived from these two approaches and the computational results obtained by running the three formulations when using the data instances from the networking application and the phylogenetic application.

Finally, we presented the system we developed which displays the discovered topology of an unknown network, using only packet delay measurements between the end-devices of the network. The system consists of two independent applications that work cooperatively. The first application synchronizes the devices, determines the delays and compiles these delays in a distance matrix. The second application determines the topology of the network using the distance matrix compiled and displays the topology in a graphical way.

Bibliography

[1] *FICO Xpress Optimization Suite*. www.fico.com/xpress.

[2] Ipswitch whatsconnected. `http://www.whatsupgold.com/products/whatsup-gold-whatsconnected.aspx`. Accessed: 2015-06-06.

[3] *Network Simulator NS-3*. http://www.nsnam.org/.

[4] Networkview. `http://www.networkview.com`. Accessed: 2015-06-06.

[5] Pc and network downloads. `http://www.pcwdld.com/top-10-network-diagram-topology-and-mapping-software`. Accessed: 2015-09-19.

[6] Solarwinds network topology mapper. `http://solarwinds.com/network-topology-mapper.aspx`. Accessed: 2015-06-06.

[7] Trickman. `http://www.trickman.net`. Accessed: 2015-06-06.

[8] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[9] J. Almeida and O. Oliveira. The discovery of the network topology, 2015. Project in Informatics from the Degree in Computer Engineering.

[10] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[11] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of operations research*, 23(4):769–805, 1998.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations research letters*, 25(1):1–13, 1999.

[13] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424, 2000.

[14] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, 4(1):63–76, 2007.

[15] T. Berthold. Primal heuristics for mixed integer programs. Master's thesis, Techical University of Berlin, 2006.

[16] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, Ser. B 98:49–71, 2003.

[17] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[18] S. Bhamidi, R. Rajagopal, and S. Roch. Network delay inference from additive metrics. *Random Structures & Algorithms*, 37(2):176–203, 2010.

[19] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous IP networks: the netinventory system. *IEEE/ACM Transactions on Networking (TON)*, 12(3):401–414, 2004.

[20] P. Buneman. A note on the metric properties of trees. *Journal of Combinatorial Theory*, 17:48–50, 1974.

[21] S.-S. Byun and C. Yoo. Reducing delivery delay in HRM tree. In *International Conference on Computational Science and Its Applications*, pages 1189–1198. Springer, 2006.

[22] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network tomography: Recent developments. *Statistical Science*, 19:499–517, 2004.

[23] R. Castro, M. Coates, and R. Nowak. Maximum likelihood identification of network topology from end-to-end measurements. In *DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling*, 2002.

[24] D. Catanzaro. The minimum evolution problem: Overview and classification. *Networks*, 53(2):112–125, 2009.

[25] D. Catanzaro. Estimating phylogenies from molecular data. In *Mathematical approaches to polymer sequence analysis and related problems*, pages 149–176. Springer, 2011.

[26] D. Catanzaro, R. Aringhieri, M. Di Summa, and R. Pesenti. A branch-price-and-cut algorithm for the minimum evolution problem. *European Journal of Operational Research*, 244(3):753–765, 2015.

[27] D. Catanzaro, M. Labbé, R. Pesenti, and J.-J. Salazar-Gonzaléz. Mathematical models to reconstruct phylogenetic trees under the minimum evolution criterion. *Networks*, 53:126–140, 2009.

[28] D. Catanzaro, M. Labbé, R. Pesenti, and J.-J. Salazar-Gonzaléz. The balanced minimum evolution problem. *INFORMS Journal on Computing*, 24(2):276–294, 2012.

[29] D. Catanzaro, R. Pesenti, and M. C. Milinkovitch. A non-linear optimization procedure to estimate distances and instantaneous substitution rate matrices under the GTR model. *Bioinformatics*, 22(6):708–715, 2006.

[30] L. L. Cavalli-Sforza and A. W. Edwards. Phylogenetic analysis. Models and estimation procedures. *American Journal of Human Genetics*, 19:233–257, 1967.

[31] M. Choudhry. *An introduction to value-at-risk*. John Wiley & Sons, 2013.

[32] F. Chung, M. Garret, R. Graham, and D. Shallcross. Distance realization problems with applications to internet tomography. *Journal of Computer and System Sciences*, 63(3):432–448, 2001.

[33] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet mapping: from art to science. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 205–211. IEEE, 2009.

[34] M. Coates, R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. In *ACM SIGMETRICS Performance Evaluation Review*, volume 30, pages 11–20, 2002.

[35] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal processing magazine*, 19(3):47–65, 2002.

[36] M. Coates, M. Rabbat, and R. Nowak. Merging logical topologies using end-to-end measurements. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet Measurement*, pages 192–203. ACM, 2003.

[37] M. J. Coates and R. D. Nowak. Network loss inference using unicast end-to-end measurement. In *ITC Conference on IP Traffic, Modeling and Management*, pages 28–1, 2000.

[38] G. Cornuejols and R. Tütüncü. *Optimization methods in finance*, volume 5. Cambridge University Press, 2006.

[39] K. Correll, N. Barendt, and M. Branicky. Design considerations for software only implementations of the ieee 1588 precision time protocol. In *Conference on IEEE*, volume 1588, pages 11–15, 2005.

[40] J. E. Corter and A. Tversky. Extended similarity trees. *Psychometrika*, 51(3):429–451, 1986.

[41] J. P. Cunningham. Trees as memory representations for simple visual patterns. *Memory & Cognition*, 8(6):593–605, 1980.

[42] W. H. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of Mathematical Biology*, 49(4):461–467, 1987.

[43] G. De Soete. A least squares algorithm for fitting additive trees to proximity data. *Psychometrika*, 48(4):621–626, 1983.

[44] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 9(5):687–705, 2002.

[45] Z. Dias, A. Rocha, and S. Goldenstein. Image phylogeny by minimal spanning trees. *IEEE Transactions on Information Forensics and Security*, 7(2):774–788, 2012.

[46] B. Donnet. Internet topology discovery. In *Data Traffic Monitoring and Analysis*, pages 44–81. Springer, 2013.

[47] B. Donnet and T. Friedman. Internet topology discovery: A survey. *IEEE Communications Surveys*, 9(4):2–14, 2007.

[48] B. Donnet and T. Friedman. Internet topology discovery: a survey. *IEEE Communications Surveys & Tutorials*, 9(4):56–69, 2007.

[49] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Deployment of an algorithm for large-scale topology discovery. *IEEE Journal on Selected Areas in Communications*, 24(12):2210–2220, 2006.

[50] A. Dress. Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups: A note on combinatorial properties of metric spaces. *Advances in Mathematics*, 53:321–402, 1984.

[51] N. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Transactions on Information Theory*, 48(1):26–45, 2002.

[52] N. Duffield, J. Horowitz, and F. L. Prestis. Adaptive multicast topology inference. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1636–1645, 2001.

[53] N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Multicast topology inference from end-to-end measurements. *Advances in Performance Analysis*, 3:207–226, 2000.

[54] N. G. Duffield, F. L. Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 915–923. IEEE, 2001.

[55] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[56] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

[57] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak. Toward the practical use of network tomography for internet topology discovery. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[58] M. Farach, S. Kannan, and T. Warnow. A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1-2):155–179, 1995.

[59] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[60] J. Felsenstein. *Inferring phylogenies*, volume 2. Sinauer Associates Sunderland, 2004.

[61] S. Fiorini and G. Joret. Approximating the balanced minimum evolution problem. *Operations research letters*, 40(1):31–35, 2012.

[62] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[63] M. Fischetti and A. Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

[64] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computation*, 1(2-3):201–222, 2009.

[65] B. Fortz, O. Oliveira, and C. Requejo. Compact mixed integer linear programming models to the minimum weighted tree reconstruction problem. *European Journal of Operational Research*, 256(1):242–251, 2017.

[66] V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483, 2014.

[67] J. A. Gallian. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, 16(6):1–219, 2009.

[68] B. Gaschen, J. Taylor, K. Yusim, B. Foley, F. Gao, D. Lang, V. Novitsky, B. Haynes, B. H. Hahn, and T. Bhattacharya. Diversity considerations in HIV-1 vaccine selection. *Science*, 296(5577):2354–2360, 2002.

[69] G. Gilmore, H. Hersh, A. Caramazza, and J. Griffin. Multidimensional letter similarity derived from recognition errors. *Perception & Psychophysics*, 25(5):425–431, 1979.

[70] A. Goëffon, J.-M. Richer, and J.-K. Hao. Voisinage d'arbre évolutif appliqué au problème maximum parcimonie. In *Premières Journées Francophones de Programmation par Contraintes*, pages 443–446. Université d'Artois, 2005.

[71] J. Goh and M. Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.

[72] B. L. Gorissen, İ. Yanıkoğlu, and D. den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015.

[73] L. Gouveia. Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. *Computers and Operations Research*, 22(9):959–970, 1995.

[74] L. Gouveia. Multicommodity flow models for spanning trees with hop constraints. *European Journal of Operational Research*, 95:178–190, 1996.

[75] R. Govindan and H. Tangmunarunkit. Heuristic for internet map discovery. In *Proceeding of the* 19^{th} *annual joint conference of IEEE Computer and Communications Societies*, volume 3, pages 1371–1380. IEEE, 2000.

[76] G. I. A. M. H. Haddadi, M. Rio and R. Mortier. Network topologies: Inference, modelling and generation. *IEEE Communications Surveys*, 10(2):48–69, 2008.

[77] S. L. Hakimi and A. Patrinos. The distance matrix of a graph and its tree realization. *Quarterly of Applied Mathematics*, 30:255–269, 1972–73.

[78] S. L. Hakimi and S. S. Yau. Distance matrix of a graph and its realizability. *Quarterly of Applied Mathematics*, 22:305–317, 1965.

[79] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. Two-stage robust integer programming. *Optimization Online*, 2014.

[80] S. Herrmann, V. Moulton, and A. Spillner. Searching for realizations of finite metric spaces in tight spans. *Discrete Optimization*, 10(4):310–319, 2013.

[81] A. Hertz and S. Varone. A note on tree realizations of matrices. *RAIRO-Operations Research*, 41(4):361–366, 2007.

[82] D. M. Hillis, C. Moritz, B. K. Mable, and R. G. Olmstead. *Molecular systematics*, volume 23. Sinauer Associates Sunderland, MA, 1996.

[83] Hirschmann. White paper: Precision clock synchronization.the standard IEEE 1588. `http://www.industrialnetworking.com/pdf/Hirschmann_IEEE_1588.pdf`. Accessed: 2015-03-16.

[84] B. Huffaker, D. Plummer, D. Moore, and K. Claffy. Topology discovery by active probing. In *Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on*, pages 90–96. IEEE, 2002.

[85] D. Huson and C. Scornavacca. A survey of combinatorial methods for phylogenetic networks. *Genome Biology and Evolution*, 3:23–35, 2011.

[86] W. Imrich and C. Simões-Pereira, J.and Zamfirescu. On optimal embedding of metrics in graphs. *J. Combin. Theory Ser. B*, 36:1–15, 1984.

[87] A. L. J. Koolen and V. Moulton. Optimal realizations of generic five-point metric. *European Journal of Combinatorics*, 30:1164–1171, 2009.

[88] T. H. Jukes and C. R. Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 3(21):132, 1969.

[89] B. H. Junker and F. Schreiber. Analysis of biological networks. 2008.

[90] K. K. Kidd and L. A. Sgaramella-Zonta. Phylogenetic analysis: concepts and methods. *American journal of human genetics*, 23(3):235, 1971.

[91] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16(2):111–120, 1980.

[92] J. Kisiala. Conditional Value-at-Risk: Theory and Applications. Master's thesis, University of Edingburgh, 2015.

[93] P. Lemey. *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press, 2009.

[94] D. Lichtenberger. An extended local branching framework and its application to the multidimensional knapsack problem. Master's thesis, Vienna University of Technology, 2005.

[95] M. G. M. Edelberg and R. Graham. On the distance matrix of a tree. *Discrere Mathematics*, 14:23–39, 1976.

[96] T. Magnanti and L. Wolsey. Optimal trees. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Models*, Handbooks in Operations Research and Management Science, Vol. 7, pages 503–615. Elsevier Science Publishers, North-Holland, 1995.

[97] J. Matson. Choosing the correct time synchronization protocol and incorporating the 1756-time module into your application.

[98] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulations and travelling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960.

[99] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on communications*, 39(10):1482–1493, 1991.

[100] R. Motamedi, R. Rejaie, and W. Willinger. A survey of techniques for internet topology discovery. *IEEE Communications Surveys & Tutorials*, 17(2):1044–1065, 2015.

[101] D. Mount. *Bioinformatics. Sequence and genome analysis.* Cold Spring Harbor Laboratory Press, 2001.

[102] M. Nei and S. Kumar. *Molecular evolution and phylogenetics.* Oxford University Press, 2000.

[103] J. Ni and S. Tatikonda. Network tomography based on additive metrics. *IEEE Transactions on Information Theory*, 57(12):7798–7809, 2011.

[104] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang. Network routing topology inference from end-to-end measurements. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[105] S. Pandey, M.-J. Choi, S.-J. Lee, and J. W. Hong. IP network topology discovery using SNMP. In *2009 International Conference on Information Networking*, pages 1–5. IEEE, 2009.

[106] D. Parker and P. Ram. The construction of Huffman codes is a submodular ("convex") optimization problem over a lattice of binary trees. *SIAM Journal on Computing*, 28(5):1875–1905, 1996.

[107] Y. Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51:41–47, 2000.

[108] K. Pavlikov and S. Uryasev. Cvar norm and applications in optimization. *Optimization Letters*, 8(7):1999–2020, 2014.

[109] V. Paxson. End-to-end internet packet dynamics. In *ACM SIGCOMM Computer Communication Review*, volume 27, pages 139–152. ACM, 1997.

[110] W. R. Pearson, G. Robins, and T. Zhang. Generalized neighbor-joining: more reliable phylogenetic tree reconstruction. *Molecular Biology and Evolution*, 16(6):806–816, 1999.

[111] G. C. Pflug. Some remarks on the value-at-risk and the conditional value-at-risk. In *Probabilistic constrained optimization*, pages 272–281. Springer, 2000.

[112] O. Prado, F. J. Von Zuben, and S. F. dos Reis. Evolving phylogenetic trees: an alternative to black-box approaches. In *Proc. of Brazilian Workshop on Bioinformatics (WOB)*, pages 56–63, 2002.

[113] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *INFOCOM'99. Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 1, pages 353–360. IEEE, 1999.

[114] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[115] R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[116] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[117] E. M. M. Santos. *O problema da árvore de suporte de custo mínimo com restrições de peso*. PhD thesis, Universidade de Aveiro, 2014.

[118] S. Sattath and A. Tversky. Additive similarity trees. *Psychometrika*, 42(3):319–345, 1977.

[119] A. Shapiro. On duality theory of conic linear problems. In *Semi-infinite programming*, chapter 7, pages 135–165. Kluwer Academic Publishers, North-Holland, 2001.

[120] Y. Shavitt and E. Shir. Dimes: Let the internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, 2005.

[121] M.-F. Shih and A. Hero. Hierarchical inference of unicast network topologies based on end-to-end measurements. *IEEE Transactions on Signal Processing*, 55(5):1708–1718, 2007.

[122] M.-F. Shih and A. O. Hero. Hierarchical inference of unicast network topologies based on end-to-end measurements. *IEEE Transactions on Signal Processing*, 55(5):1708–1718, 2007.

[123] A. Shpiner, Y. Revah, and T. Mizrahi. Multi-path time protocols. In *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*, pages 1–6. IEEE, 2013.

[124] R. Siamwalla, R. Sharma, and S. Keshav. Discovering internet topology. *Unpublished manuscript*, 1998.

[125] J. Simões-Pereira. A note on the tree realizability of a distance matrix. *Journal of Combinatorial Theory*, 6(3):303–310, 1969.

[126] J. Simões-Pereira. A note on distance matrices with unicyclic graph realizations. *Discrete Mathematics*, 65:277–287, 1987.

[127] J. Simões-Pereira. An algorithm and its role in the study of optimal graph realizations of distance matrices. *Discrete Mathematics*, 79:299–312, 1990.

[128] J. Simões-Pereira. *Matemática Discreta: Grafos, Redes, Aplicações*. Luz da Vida, Coimbra, Portugal, 2009.

[129] J. Simões-Pereira and C. Zamfirescu. Submatrices of non-tree-realizable distance matrices. *Linear Algebra and its Applications*, 44:1–17, 1982.

[130] C. Singh. Convex programming with set-inclusive constraints and its applications to generalized linear and fractional programming. *Journal of Optimization Theory and Applications*, 38(1):33–42, 1982.

[131] M. Sion et al. On general minimax theorems. *Pacific J. Math*, 8(1):171–176, 1958.

[132] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on networking*, 12(1):2–16, 2004.

[133] W. Stallings. Snmp and snmpv2: the infrastructure for network management. *IEEE Communications Magazine*, 36(3):37–43, 1998.

[134] J. A. Studier and K. J. Keppler. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular biology and evolution*, 5(6):729–731, 1988.

[135] F. Tajima and M. Nei. Biases of the estimates of DNA divergence obtained by the restriction enzyme technique. *Journal of Molecular Evolution*, 18:115–120, 1982.

[136] S. Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on mathematics in the life sciences*, 17:57–86, 1986.

[137] E. Technologies. White paper: Precision time protocol (ptp/ieee-1588). `http://www.endruntechnologies.com/pdf/PTP-1588.pdf`. Accessed: 2015-03-16.

[138] Y. Vardi. Network tomography: estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91:365–377, 1996.

[139] S. C. Varone. Trees related to realizations of distance matrices. *Discrete Mathematics*, 192:337–346, 1998.

[140] S. C. Varone. A constructive algorithm for realizing a distance matrix. *European Journal of Operational Research*, 174(1):102–111, 2006.

[141] R. J. Wilson. *An introduction to graph theory*. Pearson Education India, 1970.

[142] P. Winkler. The complexity of metric realization. *SIAM Journal on Discrete Mathematics*, 1(4):552–559, 1988.