Alma Mater Studiorum Università di Bologna

DEI - Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione
"*Guglielmo Marconi*"

Dottorato di Ricerca in Automatica e Ricerca Operativa

Ciclo XVI

Settore concorsuale di afferenza: 01/A6 - RICERCA OPERATIVA

Settore scientifico disciplinare: MAT/09 - RICERCA OPERATIVA

Networks, Uncertainty, Applications and a Crusade for Optimality

Eduardo Álvarez-Miranda

Coordinatore
Prof. Daniele Vigo

Relatore
Prof. Paolo Toth

Esame Finale 2014

Contents

Keywords

- *Network Design*
- *Robust Optimization*
- *Uncertainty modeling*
- *Combinatorial Optimization*
- *Exact Algorithms*
- *Heuristics*
- *Telecommunications*
- *Bioinformatics*
- *Logistics*

To my parents, my brother and Stella;
the facets of my combinatorial heart.

Acknowledgements

First of all, I would like to express my utmost gratitude and appreciation to my advisers, Professors Paolo Toth and Ivana Ljubić, for their enormous personal and academic support, constant encouragement, rigorous guidance and insightful criticism. They were always willing to share their knowledge and to fully commit to our research in each of the subjects we worked on together.

I want to deeply thank my colleagues and "corridor" comrades, Paolo Tubertini and Tiziano Parriani, for their friendship during these years. They helped me to understand the Italian daily-life culture (including the culinary aspects), although it is their fault that I never learned Italian.

My gratitude also goes to other members of the OR Group. I want to thank Valentina Cacchiani, for her cooperation and advice; Andrea Lodi, for his patience, support and knowledge; and Dimitri Thomopulos, for his fraternity and good humor!

I wish to thank Jelena Loncarski, for her extraordinary friendship during these years and the endless talks about the ups and downs of the grad student life (*živeli*!).

I want to express my great gratitude to Rodrigo Linfati who helped me to get adjusted in Bologna during my first weeks here and, more importantly, who was always willing to give me support when I was fighting against my programming codes (no matter the time of the day or where he was).

I greatly appreciate the Institute of Advanced Studies of the Università di Bologna, *commanded* by Professors Barbara Cimatti and Patrizia Brigidi, for the honor I had of being one of their PhD Fellows.

Last, but most certainly not least, I thank Prof. S. Raghavan, Prof. Elena Fernández, Prof. Alfredo Candia, and Rolf Karner.

Chapter 1

Introduction

1.1. Networks, Uncertainty and Applications

Networks is an ubiquitous concept in a countless number of theoretical and practical areas such as combinatorics, optimization, computer science, transportation, telecommunications and bioinformatics, to mention few of them. Networks usually embody both the input and output of a generic tool called Network Design or Network Optimization. With input we not only mean an arrangement of binary relations (typically represented by *edges* or *arcs*) among discrete units (typically referred as *nodes*), but a wide spectrum of data such as installation costs, distances, travel times, capacities, hierarchies, revenues, commodities, demands, reliability measures, connectivity requirements, and a lot more.

Network Design (ND), one of the most prominent subjects in the disciplines of Operations Research (OR) and Management Sciences (MS), can be synthesized as the process of **designing**, under one or more *criteria of optimality* (objectives), a **network** and a corresponding operating regime that meets a certain set of topological and/or operative *requirements* (constraints). Usually, we are given an *existing* or *potential* network from which we want to find a target (or optimally designed) network. The topology of such target network can be, for instance, a path, a cycle, a collection of routes, a tree, a forest or a generic connected sub-network; and the operating regime can be represented, for instance, by the frequency in which a client is visited by a vehicle through a route, a dynamic allocation of bandwidth through a local-access network, or a dispatching scheduling of a commodity through a tree-like distribution network.

It is obvious that the quality or effectiveness of the designed network strongly relies on the data used in the decision making process. In a real-world application context, it is very unlikely that values such as the travel time between two cities, the future demand of a client, or the relevance of a protein-complex in a biological process can be estimated with absolute certainty. Likewise, the dynamism of real processes usually

bans the possibility of foreseeing with complete precision the classification of clients within different hierarchies, the availability of a place for hosting a facility, or whether a pair of genes actually interacts in a particular process. Therefore, uncertainty cannot be ignored and should be taken into account when (i) gathering the data of the problem, (ii) defining the corresponding mathematical model, (iii) designing the algorithmic tools to solve it, and (iv) assessing the obtained results.

The incorporation of uncertainty is not a new element in OR. The seminal work of G. Dantzig in the 50's [see Dantzig, 1955], provided the foundations for the development of optimization under uncertainty and, in particular, of Stochastic Programming (SP). SP is comprised by set of powerful modeling and algorithmic tools, in constant progress, that are broadly applied in different fields of OR and MS such as finance, transportation or energy optimization [see Ruszczyński and Shapiro, 2003, Birge and Louveaux, 2011]. A crucial element in the application of SP is the characterization of the uncertain data by means of probabilistic measures. Although such characterization can be suitable in some cases, there are situations in which problem parameters cannot be modeled by means of probability functions. Instead, the values of these parameters belong to known sets (e.g., ellipsoidal sets, polyhedral sets, closed intervals, or sets of discrete scenarios) from where they can deterministically realize as *any* element. This type of uncertainty, usually referred as *deterministic uncertainty*, cannot be tackled by SP and it is incorporated in the decision-making process through Robust Optimization (RO).

RO is an optimization framework consisting of different approaches sharing a common paradigm: "to hedge against deterministic uncertainty providing solutions that ensure to perform *reasonably well* (in terms of optimality and/or feasibility) for all possible realizations of the parameter values". This guarantee of good performance, *robustness*, is obtained by a sort of *immunization* against the effect of data uncertainty. Commonly, for a given optimization problem we refer to its deterministic version as the *nominal problem*, and to the version incorporating robustness as its *robust counterpart*. The origins of RO can be traced back to 50's when the *max-min* model proposed by A. Wald, a decade before, became a state-of-the-art decision making approach for tackling severe uncertainty [see Wald, 1945]. A couple of decades later, RO started to be regarded as an stream of OR [see Gupta and Rosenhead, 1968, Rosenhead et al., 1972, Soyster, 1973]. A revival of RO started in 90's when many efforts were independently devoted for the establishment of different concepts and models within the framework of optimization under deterministic uncertainty [see Mulvey et al., 1995, Kouvelis and Yu, 1997, El Ghaoui et al., 1998, Ben-Tal and Nemirovski, 2000]. In the last 15 years, RO approaches have been extensively considered in several areas of mathematical optimization leading to both practical and theoretical results. Moreover, new models are constantly developed along with different definitions of uncertainty and more sophisticated concepts of robustness [Ben-Tal et al., 2010, Gabrel et al., 2014].

The core of this thesis is mainly composed by the concepts described above. That is, we address ND problems, with a strong practical motivation, and in most cases we model them by means of RO due to the presence of deterministic uncertainty in some of their parameters. The problems we consider in this thesis are the Prize Collecting Steiner Tree Problem, the Two-Level Network Design Problem, the Uncapacited Facility Location problem, the Maximum Weight Connected Subgraph Problem, the Minimum Spanning Tree Problem, the Single Commodity Flow Problem and a variant of a Network Interdiction Problem. These problems naturally appear in applications such as telecommunications, supply chain, bioinformatics and humanitarian relief planning. For these problems, we recognize the presence of uncertainty in different parameters such as transportation times, customer revenues, clients' hierarchies or facility availability. Depending on the application and the nature of uncertainty, different RO approaches are used for formulating the corresponding robust optimization problem; in particular we use the Bertsimas & Sim [Bertsimas and Sim, 2003], the Recoverable Robust [Liebchen et al., 2009], the Minmax Regret [Kouvelis and Yu, 1997] and the Soyster [Soyster, 1973] approaches.

1.2. A Crusade for Optimality

In the previous section we motivated the need of formulating ND problems under the framework of RO. Although the process of devising a suitable mathematical optimization model could entail a scientific task in itself, it is not enough for providing a decision-making tool for more realistic applications.

Classical ND problems are typically represented as Mixed Integer Programming (MIP) formulations, and in most cases, their robust counterparts are represented as MIP models as well. This means that standard mathematical programming techniques such as branch-and-cut, Benders decomposition, heuristics, etc., can be applied to solve these new models. Therefore, the next natural step is the design, implementation and assessment of an algorithmic methodology that effectively solves a given robust ND problem. This process is what we refer to as a *Crusade for Optimality*.

In this thesis, we study both nominal and robust ND problems and, in most cases, we design algorithmic frameworks to solve them. Out of seven problems, five of them are tackled by means of specially-tailored exact algorithms, one by means of a novel hybrid heuristic approach, and one using a general-purpose MIP commercial solver. In particular, note that *designing* an exact algorithm such as branch-and-cut or Benders decomposition, refers to a more general process comprised by: (i) the definition of suitable (and *strong*) MIP formulation, possibly strengthen by non-obvious valid inequalities; (ii) the election of a basic algorithmic approach (e.g., branch-and-cut or Benders decomposition) that naturally tackles the defined formulation; (iii) the implementation of algorithmic enhancements such as complementary separation procedures,

sophisticated primal heuristics or branching strategies; and (iv) the evaluation of the proposed algorithmic framework on a sufficiently large set of meaningful instances. The previously described steps not only entail a methodological task, but also a theoretical one. For instance, proving that the studied problem is polynomially solvable for a particular class of instances, can be used in the design of an efficient primal heuristic or separation procedure. Likewise, discovering new facet-defining inequalities and including their separation in the algorithm, is also a theoretical result that it is used to enhance the algorithmic performance.

1.3. Content of the Thesis

The contribution of this thesis is contained in Chapters 2-10, where the main outcomes obtained during the three years of the PhD program are exposed. Each of the nine chapters corresponds entirely to a research article whose status (submitted, accepted or published) at the moment of writing this manuscript is described below. The reader should be aware that although the notation among the chapters is very similar, it might not be exactly the same. Moreover, she/he can regard each chapter as a self-contained manuscript whose comprehension does not strictly rely on any of the other chapters.

1.3.1 Chapter 2

This chapter is based on the article "Exact approaches for solving robust prize-collecting Steiner tree problems", co-authored with I. Ljubić (Universität Wien) and Prof. P. Toth. This article has been published in *European Journal of Operational Research* [Álvarez-Miranda et al., 2013e]. A preliminary version of this work appeared in [Álvarez-Miranda et al., 2011].

In this chapter we deal with Prize Collecting Steiner Tree problem (PCStT) under uncertainty. Typically in the PCStT, we are given a set of customers with potential revenues and a set of possible links connecting these customers with fixed installation costs. The goal is to decide which customers to connect into a tree structure so that the sum of the link costs plus the revenues of the customers that are left out is minimized. The problem, as well as some of its variants, is used to model a wide range of applications in telecommunications, gas distribution networks, protein-protein interaction networks, or image segmentation.

In many applications it is unrealistic to assume that the revenues or the installation costs are known in advance. In this work we consider the well-known Bertsimas and Sim (B&S) robust optimization approach, in which the input parameters are subject to interval uncertainty, and the level of robustness is controlled by introducing a control

parameter, which represents the perception of the decision maker regarding the number of uncertain elements that will present an adverse behavior.

We propose branch-and-cut approaches to solve the robust counterparts of the PCStT and the Budget Constraint variant and provide an extensive computational study on a set of benchmark instances that are adapted from the deterministic PCStT inputs. We show how the *Price of Robustness* influences the cost of the solutions and the algorithmic performance.

Finally, we adapt our theoretical results regarding algorithms for a general class of B&S robust optimization problems for the robust PCStT and its budget and quota constrained variants.

1.3.2 Chapter 3

This chapter is based on the article "The Recoverable Robust Two-Level Network Design Problem", co-authored with I. Ljubić, S. Raghavan (University of Maryland) and Prof. P. Toth. This paper is currently under the second round of reviews in *INFORMS Journal on Computing*.

We consider a network design application which is modeled as the two level network design problem under uncertainty. In this problem, one of the two available technologies can be installed on each edge and all customers of the network need to be served by at least the lower level (*secondary*) technology.

The decision maker is confronted with uncertainty regarding the set of *primary* customers, i.e., the set of nodes that need to be served by the higher level (*primary*) technology. A set of discrete scenarios associated with the possible realizations of primary customers is available. The network is built in two stages. In the first-stage the network topology must be determined. One may decide to install the primary technology on some of the edges in the first stage, or one can wait to see which scenario will be realized, in which case, edges with the installed secondary technology may be upgraded, if necessary to primary technology, but at higher *recovery* cost. The overall goal then is to build a "recoverable robust" spanning tree in the first stage that serves all customers by at least the lower level technology, and that minimizes the first stage installation cost plus the worst-case cost needed to upgrade the edges of the selected tree, so that the primary customers of each scenario can be served using the primary technology.

We discuss the complexity of the problem, provide mixed integer programming models and develop a branch-and-cut algorithm to solve it. Our extensive computational experiments demonstrate the efficacy of our approach.

1.3.3 Chapter 4

This chapter is based on the article "The Recoverable Robust Facility Location Problem", co-authored with E. Fernández (Universitat Politècnica de Catalunya) and I. Ljubić. This article is currently under the first round of reviews in *Operations Research*.

This work deals with a facility location problem in which location and allocation policy is defined in two stages such that a first-stage solution should be robust against the possible realizations (scenarios) of the input data that can only be revealed in a second stage. This solution should be robust enough so that it can be *recovered* promptly and at low cost in the second stage. In contrast to some related modeling approaches from the literature, this new *recoverable robust* model is more general in terms of the considered data uncertainty; it can address situations in which uncertainty may be present in any of the following four categories: *provider-side* uncertainty, *receiver-side* uncertainty, uncertainty *in-between*, and uncertainty with respect to the cost parameters.

For this novel problem, a sophisticated algorithmic framework based on a Benders decomposition approach is designed and complemented by several non-trivial enhancements, including dual lifting, branching priorities, matheuristics and zero-half cuts. Two large sets of realistic instances that incorporate spatial and demographic information of countries such as Germany and US (transportation) and Bangladesh and the Philippines (disaster management) are introduced. They are used to analyze in detail the characteristics of the proposed model and the obtained solutions as well as the effectiveness, behavior and limitations of the designed algorithm.

1.3.4 Chapter 5

This chapter is based on the article "Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions", co-authored with V. Cacchiani, A. Lodi, T. Parriani (Università di Bologna) and D. Schmidt (Universität zu Köln). This paper is currently under the second round of reviews in *European Journal of Operational Research*. A preliminary version of this work appeared in [Álvarez-Miranda et al., 2012].

In this work, we study a single-commodity Robust Network Design problem (RND) in which an undirected graph with edge costs is given together with a discrete set of balance matrices, representing different supply/demand scenarios. In each scenario, a subset of the nodes is exchanging flow. The goal is to determine the minimum cost installation of capacities on the edges such that the flow exchange is feasible for every scenario.

Previously conducted computational investigations on the problem motivated the study of the complexity of some special cases and we present complexity results on them, including hypercubes. In turn, these results lead to the definition of new instances (random graphs with {-1,0,1} balances) that are computationally hard for the natural flow formulation. These instances are then solved by means of a new heuristic algorithm for RND, which consists of three phases. In the first phase the graph representing the network is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase consists of a neighborhood search on the reduced graph based on a Mixed-Integer (Linear) Programming (MIP) flow model. Finally, the third phase applies a *proximity search* approach to further improve the solution, taking into account the original graph. The heuristic is tested on the new instances, and the comparison with the solutions obtained by CPLEX on a natural flow formulation shows the effectiveness of the proposed method.

Note This chapter appears in the PhD thesis of **Tiziano Parriani** (PhD Program in *Automatic Control and Operational Research*, Cycle XVI, Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione, Università di Bologna, 2014).

1.3.5 Chapter 6

This chapter is based on the article "On Exact Solutions for the Minmax Regret Spanning Tree Problem", co-authored with A. Candia, F. Pérez-Galarce (Universidad de Talca) and Prof. P. Toth. This paper has been accepted for publication in *Computers & OR*.

The Minmax Regret Spanning Tree problem is studied in this chapter. This is a generalization of the well known Minimum Spanning Tree problem, which considers uncertainty in the cost function. Particularly, it is assumed that the cost parameter associated with each edge is an interval whose lower and upper limits are known, and the Minmax Regret is the optimization criterion. The Minmax Regret Spanning Tree problem is an NP-Hard optimization problem for which exact and heuristic approaches have been proposed.

Several exact algorithms are designed and computationally compared with the most effective approaches of the literature. It is shown that a proposed branch-and-cut approach outperforms the previous approaches when considering several classes of instances from the literature.

1.3.6 Chapter 7

This chapter is based on the article "A Note on the Bertsimas & Sim Algorithm for Robust Combinatorial Optimization Problems", co-authored with I. Ljubić and Prof.

P. Toth. This article has been published in *4OR* [Álvarez-Miranda et al., 2013d].

In this chapter we propose some improvements and extensions to the algorithmic result presented in [Bertsimas and Sim, 2003]. For the case studied in their paper, we show that instead of solving $n+1$ deterministic problems, the robust counterpart can be computed by solving $n-\Gamma_X+2$ deterministic problems (Lemma 1); this improvement is particularly interesting for those cases for which a high level of conservatism, i.e., a large value of Γ_X, is suitable. Additionally, we show that if a knapsack-type constraint is part of a problem and m of its coefficients are affected by uncertainty, an equivalent algorithmic approach can be applied, and the robust counterpart can be computed by solving $m-\Gamma_Y+2$ deterministic problems (Lemma 2), for $0<\Gamma_Y\leq m$. Likewise, we show that if the uncertain coefficients in the objective function are associated with two disjoint sets of variables, of size n and m respectively, the robust problem can be computed by solving of $(n-\Gamma_X+2)(m-\Gamma_Y+2)$ deterministic problems (Lemma 3), giving to the decision maker the flexibility to define different levels of conservatism to different sets of uncertain parameters. A similar result is also shown for the case that uncertainty is present in a set of n objective function coefficients and in a set of m coefficients of a knapsack-type constraint (Lemma 4). Combining the previous results, we provide a more general result which considers the case in which the uncertain coefficients in the objective function are associated with K disjoint sets of variables and there are L knapsack-type constraints (each of them involving a different set of variables) with uncertain coefficients. For this type of problems, we show that the robust counterpart can be computed by solving a strongly-polynomial number of deterministic problems (Theorem 1), assuming that K and L are constant.

1.3.7 Chapter 8

This chapter is based on the article "Vulnerability Assessment of Spatial Networks: Models and Solutions", co-authored with A. Candia, F. Pérez-Galarce and E. Carrizosa (Universidad de Sevilla). This article has been accepted for publication in the proceedings of the *3rd International Symposium on Combinatorial Optimization* (March 5-8, 2014, Lisbon), edited by L. Gouveia and R. Mahjoub and published in the series LNCS by Springer.

Based on a well-known network interdiction model we formulate a framework of combinatorial optimization problems whose solutions can be used for assessing the vulnerability of spatial networks in the case of disruptions. We design a flexible model of network disruption based on the geometric characteristic of spatial networks. This model incorporates the nature of disruptions present in different situations such as military planning [Golden, 1978, Israeli and Wood, 2002], terrorist attacks [Salmeron et al., 2009] or emergency control of infectious disease spreading [Assimakopoulos, 1987] . The proposed problems, along with the model of disruption, span several realizations

of network interdiction providing a useful tool to characterize network vulnerability. In a way, our aim is to propose a methodology that uses network optimization problems to characterize the robustness of a network in the presence of multiple failures.

1.3.8 Chapter 9

This chapter is based on the article "The Maximum Weight Connected Subgraph Problem", co-authored with I. Ljubić and P. Mutzel (Technische Universität Dortmund). This article is the 11th chapter of the book *Facets of Combinatorial Optimization*, edited by M. Jünger and G. Reinelt in the occasion of the 65th birthday of Martin Grötschel [Álvarez-Miranda et al., 2013a].

The *Maximum (Node-) Weight Connected Subgraph Problem* (MWCS) searches for a connected subgraph with maximum total weight in a node-weighted (di)graph. It has various applications in systems biology, computer vision, communication network design, forestry, and wildlife preservation planning. In this work we introduce a new integer linear programming formulation built on node variables only, which uses new constraints based on node-separators. We theoretically compare its strength to previously used MIP models in the literature and study the connected subgraph polytope associated with our new formulation. In our computational study we compare branch-and-cut implementations of the new model with two models recently proposed in the literature: one of them using the transformation into the Prize-Collecting Steiner Tree problem, and the other one working on the space of node variables only. The obtained results indicate that the new formulation outperforms the previous ones in terms of the running time and in terms of the stability with respect to variations of node weights.

1.3.9 Chapter 10

This chapter is based on the article "The Rooted Maximum Node-Weight Connected Subgraph Problem", co-authored with I. Ljubić and P. Mutzel. This article has been published in the proceedings of the *10th Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) techniques in Constraint Programming* (May 18-22, 2013, Yorktown Heights), edited by C. Gomes and M. Sellman and published in the series LNCS by Springer [Álvarez-Miranda et al., 2013b].

Given a connected node-weighted (di)graph, with a root node r, and a (possibly empty) set of nodes R, the *Rooted Maximum Node-Weight Connected Subgraph Problem* (RMWCS) is the problem of finding a connected subgraph rooted at r that connects all nodes in R with maximum total weight. In this work we consider the RMWCS as well as its budget-constrained version, in which also non-negative costs of the nodes are given, and the solution is not allowed to exceed a given budget. The considered

problems belong to the class of network design problems and have applications in various different areas such as wildlife preservation planning, forestry, system biology and computer vision.

We present three new integer linear programming formulations for the problem and its variant which are based on node variables only. These new models rely on a different representation of connectivity than the one previously presented in the RMWCS literature that rely on a transformation into the Steiner Arborescence problem. We theoretically compare the strength of the proposed and the existing formulations, and show that one of our models preserves the tight LP bounds of the previously proposed cut set model of Dilkina and Gomes. Moreover, we study the rooted connected subgraph polytope in the natural space of node variables. We conduct a computational study and (empirically) compare the theoretically strongest one of our formulations with the one previously proposed using ad-hoc branch-and-cut implementations.

1.3.10 Chapter 11

General conclusions and final remarks are drawn in this chapter. The potential of extending the obtained results for other problems is also discussed.

Chapter 2

Exact Approaches for Solving Robust Prize-Collecting Steiner Tree Problems

2.1. Introduction

When defining an expansion plan of a fiber optic network in a given area and for a given planning horizon, a telecommunication company needs to decide to which subset of customers a service should be provided. Thereby, two elements need to be taken into account: potential gains in revenue (that will be referred to as *prizes*) of each customer, and infrastructure *costs* needed to connect them. This problem can be formulated as a network optimization problem called the *Prize-Collecting Steiner Tree Problem* (PCStT). In this chapter we will focus on the PCStT and the Budget and Quota constrained variants, under data uncertainty assumption.

When facing strategic decisions modeled by the PCStT, companies should consider the presence of uncertainty in problem parameters as an inevitable feature of the decision-making process. In our particular case, customer revenues and connection costs are uncertain parameters since they are affected by many external economic or even social factors. Consequently, uncertainty in both groups of parameters (or at least one of them) should be part of any decision model in order to obtain reliable and robust solutions from the economic point of view.

In our models, robustness can be seen as a guarantee of protection against data uncertainty. This guarantee is provided by the use of the Bertsimas and Sim (B&S) Robust Optimization (RO) approach [see Bertsimas and Sim, 2003], which entails the adoption of *protection functions* that are included in the objective function and/or constraints. Protection functions depend on both, the uncertainty present in the problem's input

parameters and the intuition of the decision maker. These protection functions are all in all what determines the *Price of Robustness* [see Bertsimas and Sim, 2004] which can be defined as the worsening of the economic performance of the solutions while ensuring higher level of robustness in presence of higher levels of uncertainty. The resulting model will be called *robust counterpart* of the original deterministic problem.

The PCStT arises as an important problem in Network Optimization from both the algorithmic and practical points of view (see Section 2.2). Therefore, we believe that studying the robust counterpart of the PCStT will help in solving and better understanding not only the robust PCStT itself, but also other related problems in the area of *robust network optimization*.

In this work we propose several RO variants of the PCStT and establish some connections between them. As main contribution, we propose three different strategies to exactly solve the Robust PCStT (RPCStT). These exact algorithms are all based on Branch-and-Cut techniques and the differences among them are implied by the underlying mathematical programming formulations and the different cutting-plane techniques. An extensive analysis of computational results is carried out in order to assess the performance of the proposed algorithms and their dependence on the problem parameters, and the nature and characteristics of the obtained solutions. This analysis concerns a qualitative study of the solutions in terms of the *Price of Robustness* and an interpretation and assessment of the different algorithmic performances. To complement this analysis, we also consider a budget-constrained variant of the PCStT and adapt the developed algorithms to solve its robust counterpart.

Structure of the Paper In Section 2.2, the PCStT is formally defined, a review of the main literature is presented, two important variants of the problem, i.e., Budget and Quota PCStT, are defined, and an integer programming formulation is provided. In Section 2.3, motivations and alternatives to consider parameter uncertainty are presented with an emphasis on the B&S robust optimization model. Subsequently, different Mixed Integer Programming (MIP) formulations for the robust counterpart of the PCStT and the Budget and Quota Constraint variants are presented. Branch-and-Cut algorithms are presented in Section 2.4. In Section 2.5 we present and analyze the computational results obtained for different sets of benchmark instances for the robust counterparts of the PCStT and its budget-constrained variant. In Section 2.6, our recent theoretical results regarding algorithms for a general class of B&S robust optimization problems [see Álvarez-Miranda et al., 2013d] are adapted for the robust PCStT and its variants. Finally, concluding remarks and paths for future work are presented in Section 2.7.

2.2. The Prize Collecting Steiner Tree Problem

The term *Prize Collecting* was used for the first time by Balas [see Balas, 1989], in the context of the traveling salesman problem. However, it was in [Bienstock et al., 1993] where the PCStT has been introduced. It is worth to mention that in [Segev, 1987], Segev studied for first time the closely related Steiner tree problem with node weights. A formal definition of the PCStT can be given as follows.

Given is an undirected graph $G = (V, E)$ with $n = |V|$, $m = |E|$, edge costs $c_e \in \mathbb{R}^{>0}$ for all $e \in E$, and node prizes $p_v \in \mathbb{R}^{\geq 0}$ for all $v \in V$. The PCStT consists of finding a tree $T = (V_T, E_T)$ of G, that minimizes the function

$$f(T) = \sum_{e \in E_T} c_e + \sum_{v \in V \setminus V_T} p_v. \tag{2.1}$$

For a feasible solution T, function (2.1) corresponds to the sum of the costs c_e of the edges in the tree, $e \in E_T$, plus the sum of the prizes p_v of the nodes that are *not spanned* by the tree, $v \in V \setminus V_T$; this definition of the PCStT is known as the Goemans and Williamson PCStT (GW-PCStT) [Bienstock et al., 1993]. In the context of the expansion of fiber optic networks mentioned above, graph $G = (V, E)$ is the potential network for which we want to find an expansion plan, so edges $e \in E$ are the possible links with construction costs c_e and nodes $v \in V$ represent customers or street intersections with potential revenues $p_v > 0$ or $p_v = 0$, respectively. By $V_{p_i>0}$ ($n' = |V_{p_i>0}|$) we will denote the set of *potential customers* and by $V_{p_i=0}$, the set of potential Steiner nodes.

The PCStT can be also defined as the problem of finding a tree T that minimizes

$$f_{NW}(T) = \sum_{e \in E_T} c_e - \sum_{v \in V_T} p_v. \tag{2.2}$$

Function (2.2) corresponds to the minimization version of the Net-Worth PCStT (NW-PCStT) which was introduced in [Johnson et al., 2000]. Although functions (2.1) and (2.2) are equivalent in the sense that both produce the same optimal solutions, they are not equivalent regarding approximation algorithms [see Johnson et al., 2000].

Approximation algorithms for the GW-PCStT are discussed in [Bienstock et al., 1993, Goemans and Williamson, 1997, Johnson et al., 2000] and recently in [Archer et al., 2011]. Heuristic procedures are implemented in [Canuto et al., 2001, Klau et al., 2004] and [Salles da Cunha et al., 2009]. The first published work on polyhedral studies for the PCStT is [Lucena and Resende, 2004], where a cutting plane algorithm is proposed. The cuts are efficiently generated when a violation of a generalized subtour elimination constraint (GSEC) is verified. In [Ljubić et al., 2006], a branch-and-cut algorithm based on a directed cut-set MIP formulation has been designed and implemented. Several state-of-the-art methods are combined and pre-processing techniques are used.

The proposed procedure has significantly improved the algorithm presented in [Lucena and Resende, 2004]. The same set of benchmark instances has been solved by two orders of magnitude smaller running times. Optimal solutions have also been achieved for large-scale real-world instances concerning the design of optical fiber networks. Another important algorithmic efforts for the PCStT and some of its variants have been presented in [Canuto et al., 2001, Haouari and Siala, 2006, Haouari et al., 2008] and [Haouari et al., 2010].

In [Ljubić et al., 2006] an application of the problem is approached for the first time; the exact algorithm developed in the paper is used to solve real world instances for the design of fiber optic networks of a German city where an existing subnetwork needed to be augmented in order to serve new customers in the most profitable way. Over the last few years various other applications have been studied in which the PCStT has shown to play a crucial role in the modeling process. These problems arise from very different industrial and scientific contexts, showing the potential and versatility of the PCStT as a modeling tool.

Relevant applications of the PCStT are found in Bioinformatics in the context of protein-protein interaction networks (PPIN). In [Dittrich et al., 2008, Bailly-Bechet et al., 2009, Huang and Fraenkel, 2009] and [Bailly-Bechet et al., 2010] the PCStT is applied to network optimization problems arising in the analysis of PPIN for different datasets of biological processes. The PCStT is used to model an "inference problem" in order to find, or rather "to infer", functional modules in PPIN. These networks represent signal pathways (constructed by edges) between proteins or protein complexes (represented by nodes). These biological networks are modeled as a graph $G = (V, E)$, where edge costs c_e represent the confidence of interaction between the source and the target of the given edge e, and node prizes p_v corresponds to the differential expression of node v in the network for a given biological process. In [Lasher et al., 2011], where a survey of models and algorithms for cellular response networks is provided, the PCStT and the algorithm studied in [Dittrich et al., 2008] (which is based on the exact approach developed in [Ljubić et al., 2006]) are presented as state-of-the-art tools for the detection of response networks in the context of analysis of gene expressions. Recently in [Huang, 2011], the author emphasizes the quality of the results obtained using the PCStT model compared with other modeling and algorithmic approaches for the analysis of signaling networks carried out over different gene databases.

The design of a leakage detection system using the PCStT is performed in [Prodon et al., 2010]. The problem consists of finding the optimal location of detectors in an urban water distribution network so that, given a budget constraint, a desired coverage is provided. The instance considered in the paper corresponds to the urban water distribution network of the city of Lausanne, Switzerland.

In [Vijayanarasimhan and Grauman, 2011], the PCStT is used to efficiently detect region-based objects in the context of image recognition. Nodes v represent superpixels and edges e connect pairs of superpixels that share a boundary. Node prizes p_v represent the contribution of the superpixel to the classifier score, and edge costs are a measure of the probability of two superpixels to belong to the same element. The objective is to find a best-scoring subregion identifying the most likely region of the object of interest. It is important to remark that in [Dittrich et al., 2008] and [Vijayanarasimhan and Grauman, 2011] the equivalence between the PCStT and the Maximum-weight connected subgraph problem (MWCS) is exploited to model the particular problem. For more details we refer the reader to [Ideker et al., 2002].

2.2.1 A Integer Programming Formulation for PCStT

To characterize the set of feasible solutions for the PCStT, i.e., subtrees of G, we consider a directed graph model and use connectivity inequalities to guarantee connectivity of the solution.

We transform the graph $G = (V, E)$ into the directed graph $G_{SA} = (V_{SA}, A_{SA})$. The vertex set $V_{SA} = V \cup \{r\}$ contains the nodes of the input graph G *and an artificial root* vertex r. The arc set A_{SA} is defined as $A_{SA} = \{(r, i) \mid i \in V_{p_i>0}\} \cup A$, where $A = \{(i, j), (j, i) \mid e = \{i, j\} \in E\}$. A subgraph T_{SA} of G_{SA} that forms a directed tree rooted at r such that for each node i in T_{SA} there is a directed path between r and i, is called a *Steiner arborescence* and is a feasible solution of the problem in case there is only one outgoing arc from r. We will use the following notation: A set of vertices $R \subset V_{SA}$ and its complement $\overline{R} = V_{SA} \backslash R$, $R \neq \emptyset$, induce two directed cuts: $\delta^+(R) = \big\{(i, j) \mid i \in R, j \in \overline{R}\big\}$ and $\delta^-(R) = \big\{(i, j) \mid i \in \overline{R}, j \in R\big\}$. Let z_{ij}, $\forall (i, j) \in A$, be a binary variable such that $z_{ij} = 1$ if arc (i, j) belongs to a feasible arborescence T_{SA} and $z_{ij} = 0$ otherwise. Let y_i, $\forall i \in V$, be a binary variable such that $y_i = 1$ if node i belongs to T_{SA} and $y_i = 0$ otherwise. The set of constraints that characterizes the set of feasible solutions of PCStT is given by:

$$\sum_{(j,i)\in\delta^-(i)} z_{ji} = y_i \qquad \forall i \in V_{SA} \backslash \{r\} \tag{2.3}$$

$$\sum_{(i,j)\in\delta^-(R)} z_{ij} \geq y_k, \quad k \in R, \quad \forall R \subseteq V_{SA} \setminus \{r\}, \quad R \neq \emptyset \tag{2.4}$$

$$\sum_{(r,i)\in\delta^+(r)} z_{ri} = 1 \tag{2.5}$$

Let x_e, $\forall e \in E$, be a binary variable such that $x_e = 1$ if edge e belongs to a feasible *subtree* T (induced by T_{SA}) and $x_e = 0$ otherwise. The connection between $\mathbf{x}$ and $\mathbf{z}$ variables is given by

$$x_e = z_{ij} + z_{ji} \quad \forall e = \{i, j\} \in E \tag{2.6}$$

The corresponding set of feasible solutions satisfying these inequalities is given as:

$$\mathscr{T} = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{|E|+|V|} \mid (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ satisfies } (2.3) - (2.6) \text{ and } \mathbf{z} \in \{0,1\}^{|A_{SA}|}\}.$$

Constraints (2.4), also known as *cut* or *connectivity inequalities*, are the directed counterpart of undirected GSECs used in [Lucena and Resende, 2004]. They ensure that there is a directed path from the root r to each customer k such that $y_k = 1$. In-degree constraints (2.3) guarantee that the in-degree of each vertex of the tree is equal to one. The root-out-degree constraint (2.5) makes sure that the artificial root is connected to exactly one of the terminals. In addition, the following inequalities are used to initialize the MIP model:

$$z_{rj} \leq 1 - y_i, \quad \forall i < j, \quad i, j \in V_{p_i>0} \tag{2.7}$$

$$\sum_{(j,i)\in\delta^-(i)} z_{ji} \leq \sum_{(i,j)\in\delta^+(i)} z_{ij}, \quad \forall i \in V_{p_i=0}. \tag{2.8}$$

Constraints (2.7), the so-called *asymmetry constraints*, ensure that for each feasible solution the customer vertex adjacent to the root is the one with the smallest index. Inequalities (2.8) are the *flow-balance constraints*, originally introduced for the Steiner tree problem [see Koch and Martin, 1998]. Constraints (2.7) cut off symmetric solutions, while constraints (2.8) improve the quality of lower bounds of the Linear Programming (LP) relaxation of the MIP model.

In the remainder, let $T = (V_T, E_T)$ denote the tree induced by a pair $(\mathbf{x}, \mathbf{y})$, such that $E_T = \{e \mid x_e = 1\}$ and $V_T = \{v \mid y_v = 1\}$. For simplicity of notation we state that $T \equiv (\mathbf{x}, \mathbf{y})$.

2.2.2 Variants of the PCStT: Budget and Quota PCStT

Two well-known variants of the PCStT are the Budget Constrained PCStT (B-PCStT) and the Quota Constrained PCStT (Q-PCStT), which are presented for the first time in [Johnson et al., 2000], where also approximation algorithms and computational studies have been provided.

Given a cost budget B, $B \in \mathbb{R}^{\geq 0}$, representing the maximum total cost allowed for the construction of the solution, the B-PCStT is defined as

$$f_B^*(T) = \min_{T\in\mathscr{T}} \left\{ \sum_{v\in V\setminus V_T} p_v \,\middle|\, \sum_{e\in E_T} c_e \leq B \right\}. \tag{2.9}$$

Given a prize quota Q, $Q \in \mathbb{R}^{>0}$, representing the maximum total prize allowed to be left out of a solution (or the total prize allowed to be lost), the Q-PCStT is defined as

$$f_Q^*(T) = \min_{T \in \mathscr{T}} \left\{ \sum_{e \in E_T} c_e \,\middle|\, \sum_{v \in V \setminus V_T} p_v \leq Q \right\}. \tag{2.10}$$

Problem (2.9) and (2.10) are natural extensions of the problem that appear in the bi-objective optimization framework. There are two conflicting goals, namely, minimization of the cost and maximization of the profit, and typically, one can solve these problems in iterative frameworks by e.g., the weighted sum approach or ϵ-constrained based approaches [see Ehrgott and Gandibleux, 2000].

2.3. Formulations for Robust PCStT and its Variants

2.3.1 Robust Optimization Approaches

In this work we consider decision-making environments with a lack of complete knowledge about the uncertain state of data and instead of dealing with probabilistic uncertainty (as in *stochastic optimization* [see Uryasev and Pardalos, 2001]) we actually deal with *deterministic uncertainty* [Bertsimas and Sim, 2003]. In contrast to probabilistic models, that treat the input parameters as random variables, in the deterministic uncertainty models we assume that the input parameters belong to a known deterministic set. This is in the core of many real world applications and it is the motivation supporting the robust optimization approaches, where the essential objective is to find solutions that will have a *reasonably good* performance (of optimality and/or feasibility) for all possible realizations of the parameter values.

In the last 20 years several RO models have been proposed, corresponding to different motivations and conceptual definitions; for a deep and extensive study on the RO we refer the reader to [Ben-Tal et al., 2010]. In our opinion there are three main characteristics that define the differences among RO models: **(1)** The nature of the input data; whether the data belong to e.g., an ellipsoidal set or polyhedral set, a closed interval, or a set of discrete scenarios; **(2)** If robustness is considered with respect to the value of the objective function (*robust solution*), to the feasibility of the solution (*robust model*) or both; **(3)** The definition of *reasonably good performance* of a solution, which is what determines the main features of the model.

In this work we consider the RO concept by Bertsimas and Sim (B&S) defined in [Bertsimas and Sim, 2003] and [Bertsimas and Sim, 2004]. This model is considered as one of the most important references in the field of RO. Regarding the first characteristic mentioned above, this approach tackles interval uncertainty. Regarding robustness, the B&S model allows to find solutions that are robust in terms of optimality and/or

feasibility of the solution. The definition of what is a *reasonably good performance* of a solution is given by the protection against a pre-defined number of parameters that might be subject to uncertainty.

In this work we consider interval uncertainty, which means that associated with each input parameter there is a closed interval with its lower and upper bounds. Formally, in the case of the PCStT, an interval $[c_e^-, c_e^+]$, such that $0 < c_e^- \leq c_e^+$, is associated with each edge $e \in E$, and an interval $[p_v^-, p_v^+]$, such that $0 \leq p_v^- \leq p_v^+$ is associated with each customer $v \in V_{p_i>0}$. To simplify the notation, we will define $0 \leq p_v^- \leq p_v^+$ for all nodes $v \in V$, where $p_v^- = p_v^+ = 0$ for potential Steiner nodes $v \in V_{p_i=0}$. Since we consider deterministic uncertainty, each input parameter can take any value from the corresponding interval without any specific (or known) behavior and independently of the values taken by the other parameters. The lower interval values c_e^- and p_v^- will be referred to as *nominal values*, i.e., they are the values to be considered if the deterministic PCStT is solved. *Deviations* from the nominal values are defined as: $d_e = c_e^+ - c_e^-$, for all $e \in E$ and $d_v = p_v^+ - p_v^-$, for all $v \in V$. In the following we will present two ways to derive mathematical programming formulations for the robust counterpart of the PCStT and its variants.

The PCStT under interval uncertainty has been considered before in [Álvarez-Miranda et al., 2010]. The authors used an alternative RO model based on a Risk/Cost trade-off concept defined in [Chen et al., 2009] and provided polynomial time algorithms for solving both the PCStT and its robust counterpart on 2-trees. In this context, our work is complementary since we consider a different RO model and we provide a more general algorithmic framework focusing on graphs with general structure. The PCStT under interval uncertainty with the B&S RO model has been introduced in our preliminary work [Álvarez-Miranda et al., 2011]. In that work, one of the three approaches studied in this work has been computationally tested; however, only for the robust version of the PCStT and on a subset of the instances that are considered here.

2.3.2 The B&S Robust PCStT

Suppose that a decision maker wants to solve the PCStT in which the input parameters, edge costs and node prizes, are subject to interval uncertainty. In many practical applications it is unlikely that all of edge costs and/or node prizes will present an uncertain behavior at the same time. Therefore, we assume that only a subset of input data is subject to uncertainty, while the remaining parameters are fixed to their *nominal* values. More precisely, the decision maker may assume that only Γ_E edges and Γ_V nodes ($\Gamma_E \in [0, m]$ and $\Gamma_V \in [0, n']$) will be subject to uncertainty, although she/he does not know exactly which they are. Without loss of generality, we will assume that the values of Γ_E and Γ_V are integral.

The essence of the model is to find a solution that is "robust" considering all scenarios in which Γ_E edges and Γ_V nodes present an adverse behavior. If $\Gamma_E = 0$ and $\Gamma_V = 0$, then uncertainty is ignored and the problem to solve is nothing but the nominal problem, whereas if $\Gamma_E = m$ and $\Gamma_V = n'$, i.e., full uncertainty is assumed, the most conservative robust solution is sought.
Considering the general mathematical programming formulation for combinatorial optimization problems with interval uncertainty presented in [Bertsimas and Sim, 2003], the B&S RPCStT can be formulated as

$$ROPT(\Gamma_E, \Gamma_V) = \min_{T \in \mathscr{T}} \left\{ \sum_{e \in E_T} c_e^- + \beta_E^*(\Gamma_E) + \sum_{v \in V \setminus V_T} p_v^- + \beta_V^*(\Gamma_V) \right\}, \tag{2.11}$$

where

$$\beta_E^*(\Gamma_E) = \max \left\{ \sum_{e \in \tilde{E} \cap E_T} d_e \mid \forall \tilde{E} \subseteq E, |\tilde{E}| \leq \Gamma_E \right\}$$

and

$$\beta_V^*(\Gamma_V) = \max \left\{ \sum_{v \in \tilde{V} \cap \{V \setminus V_T\}} d_v \mid \forall \tilde{V} \subseteq V, |\tilde{V}| \leq \Gamma_V \right\}.$$

These last two functions are the so-called protection functions and they provide robustness to the solutions in terms of protection of optimality in presence of a given level of data uncertainty, represented by Γ_E and Γ_V.

An optimal solution for (2.11) can be interpreted as the one that minimizes the total nominal cost plus the cost of the maximal Γ_E deviations in the cost of the edges of the solution plus the maximal Γ_V deviations in the prizes of the nodes that are *not* spanned by the solution. If $\Gamma_E = m$ and $\Gamma_V = n'$, the solution will obviously correspond to the optimal (worst-case) deterministic solution in which all edge costs and node prizes will be set to their upper bounds. The flexibility provided by Γ_E and Γ_V is the main advantage of the model from the practical point of view, because it allows the decision maker to include her/his preferences in order to control the level of conservatism of the solutions.

Formulation Based on Compact Robust Constraints: To find a mixed integer programming formulation for (2.11), it is necessary to rewrite protection functions $\beta_E^*(\Gamma_E)$ and $\beta_V^*(\Gamma_V)$ using auxiliary variables $u_e \in [0,1]$, $\forall e \in E$ and $u_v \in [0,1]$, $\forall v \in V$, which represent the portion of the corresponding deviation, d_e and d_v respectively, included into the protection function. We thus obtain

$$\beta_E^*(\Gamma_E) = \max \left\{ \sum_{e \in E_T} d_e u_e \mid u_e \in [0,1] \ \ \forall e \in E, \sum_{e \in E} u_e \leq \Gamma_E \right\} \tag{2.12}$$

and

$$\beta_V^*(\Gamma_V) = \max \left\{ \sum_{v \in V \setminus V_T} d_v u_v \mid u_v \in [0,1] \ \ \forall v \in V, \sum_{v \in V} u_v \leq \Gamma_V \right\}. \tag{2.13}$$

When considering (2.12) and (2.13) it is clear that the objective function of (2.11) contains two non-linear nested maximization problems. To overcome this, one can use strong duality. Let $T^* \equiv (\mathbf{x}^*, \mathbf{y}^*)$ be an optimal tree for (2.11). Objective functions of problems (2.12) and (2.13) can be written as $\sum_{e \in E} d_e x_e^* u_e$ and $\sum_{v \in V} d_v (1 - y_v^*) u_v$, respectively. By strong duality [see Bertsimas and Sim, 2003], we have:

$$\beta_E^*(\Gamma_E) = \min \left\{ \theta \Gamma_E + \sum_{e \in E} h_e \mid h_e + \theta \geq d_e x_e^* \text{ and } h_e \geq 0 \ \forall e \in E, \ \theta \geq 0 \right\} \tag{2.14}$$

and

$$\beta_V^*(\Gamma_V) = \min \left\{ \lambda \Gamma_V + \sum_{v \in V} k_v \mid k_v + \lambda \geq d_v \left(1 - y_v^*\right) \text{ and } k_v \geq 0 \ \forall v \in V, \ \lambda \geq 0 \right\}, \tag{2.15}$$

respectively.

Combining (2.11), (2.14) and (2.15), we can formulate the B&S RPCStT as the following Mixed Integer Programming (MIP) model:

$$ROPT(\Gamma_E, \Gamma_V) = \min \sum_{e \in E} c_e^- x_e + \theta \Gamma_E + \sum_{e \in E} h_e + \sum_{v \in V} p_v^- \left(1 - y_v\right) + \lambda \Gamma_V + \sum_{v \in V} k_v \tag{2.16}$$

s.t.

$$h_e + \theta \geq d_e x_e, \ \forall e \in E \tag{2.17}$$

$$k_v + \lambda \geq d_v \left(1 - y_v\right), \ \forall v \in V \tag{2.18}$$

$$h_e \geq 0 \ \forall e \in E, \ k_v \geq 0 \ \forall v \in V \text{ and } \theta, \lambda \geq 0 \tag{2.19}$$

$$(\mathbf{x}, \mathbf{y}) \in \mathscr{T}. \tag{2.20}$$

In this model, variables h_e, k_v, θ and λ are called "robust variables", while constraints (2.17) and (2.18) are called "compact robust-constraints" as their number is linear in m and n.

Formulation Based on Robustness Cuts: One can also use Benders decomposition to project out robust variables from the previous formulation. Since every solution $(\mathbf{x}, \mathbf{y}) \in \mathscr{T}$ is feasible for the robust counterpart of the problem, only Benders *optimality cuts* will be needed to describe the robustness of an optimal solution. These

optimality cuts are given by constraints (2.22) and (2.23) below:

$$ROPT(\Gamma_E, \Gamma_V) = \min \sum_{e \in E} c_e^- x_e + \Theta + \sum_{v \in V} p_v^- (1 - y_v) + \Lambda \tag{2.21}$$

s.t.

$$\Theta \geq \sum_{e \in S} d_e x_e, \ \forall S \subseteq E, \ |S| \leq \Gamma_E \tag{2.22}$$

$$\Lambda \geq \sum_{v \in R} d_v (1 - y_v), \ \forall R \subseteq V, \ |R| \leq \Gamma_V \tag{2.23}$$

$$\Theta, \Lambda \geq 0 \tag{2.24}$$

$$(\mathbf{x}, \mathbf{y}) \in \mathscr{T}. \tag{2.25}$$

In this model, additional variables Θ and Λ and constraints (2.22) and (2.23) allow to model the two nested maximization problems $\beta_E^*(\Gamma_E)$ and $\beta_V^*(\Gamma_V)$, respectively. Constraints (2.22) and (2.23) are called "robustness cuts". In this model we enforce robustness by working directly on the space of variables $(\mathbf{x}, \mathbf{y})$ at the expense of adding an exponential number of robustness constraints. In Section 2.4, we will show that these constraints can be separated in polynomial time. In Section 2.5 we will provide a computational study comparing the practical performance of the compact robust constraints versus these robustness cuts. In [Fischetti and Monaci, 2012], the authors have proposed to use robustness cuts for modeling robust linear optimization problems with uncertainty in the constraint parameters.

2.3.3 The B&S Robust NW-PCStT and Equivalences

It is known that for the deterministic case the connection between $f(T)$ and $f_{NW}(T)$ is given as

$$f_{NW}(T) = f(T) - \sum_{v \in V} p_v,$$

i.e., the two formulations of deterministic GW-PCStT and NW-PCStT find the same solution because the sum of node revenues is constant. However, when node revenues are subject to interval uncertainty and a B&S robust solution is sought, this sum is not constant anymore. In this case, the robust counterpart of the NW-PCStT is essentially solving a different problem. To better understand this difference, assume for a moment that edge costs are deterministic. Recall now that in the robust counterpart of the GW-PCStT, nominal values for node revenues are set to conservative lower bounds and, therefore *ROPT* corresponds to a *potential increase of revenues*, which a decision maker can miss. On the other hand, conservative setting for the node revenues in the robust NW-PCStT case is to assume the values are set to their upper bounds, p_v^+, for all $v \in V$.

By following the same ideas presented above for the GW-PCStT, the B&S Robust counterpart of the NW-PCStT is defined as:

$$ROPT_{NW}(\Gamma_E, \Gamma_V) = \min_{(\mathbf{x},\mathbf{y})\in\mathscr{T}} \left\{ \sum_{e\in E} c_e^- x_e + \beta_E^*(\Gamma_E) - \left(\sum_{v\in V} p_v^+ y_v - \eta_V^*(\Gamma_V) \right) \right\} \quad (2.26)$$

where

$$\eta_V^*(\Gamma_V) = \max \left\{ \sum_{v\in V} d_v u_v \mid \sum_{v\in V} u_v \leq \Gamma_V,\ u_v \in [0,1]\ \forall v \in V \right\}.$$

In other words, when assuming deterministic edge costs, $ROPT_{NW}$ corresponds to a *potential decrease of revenues*, that the decision maker can experience. It can be easily seen from (2.26) that larger values of Γ_V will increase the total value of the solution (i.e., decrease the total revenue) as it is expected in this RO model. A MIP formulation can be obtained accordingly by following the same procedure explained for the GW-PCStT.

Despite the fact that these two robust formulations essentially model different problems, the next result shows that in particular cases the two formulations are the same.

Observation 1. For a fixed value of $\tilde{\Gamma}_E \in [0, m]$, and $\Gamma_V \in \{0, n'\}$, the robust counterparts of the GW-PCStT and of the NW-PCStT are equivalent, i.e., they produce identical optimal subtrees. The following connection exists between the corresponding objective values:

$$ROPT_{NW}(\tilde{\Gamma}_E, 0) = ROPT(\tilde{\Gamma}_E, n') - \sum_{v\in V} p_v^+$$

and

$$ROPT_{NW}(\tilde{\Gamma}_E, n') = ROPT(\tilde{\Gamma}_E, 0) - \sum_{v\in V} p_v^-.$$

2.3.4 The B&S Robust B-PCStT and Q-PCStT

In the case of both the GW-PCStT and the NW-PCStT, uncertainty is present only in the coefficients of the objective function, which means that their robust counterparts provide protection with respect to the optimality of the solutions. However, in the case of the B-PCStT and if the Q-PCStT, the presence of uncertainty in edge costs and in node prizes affects not only their corresponding objective functions but also their budget and quota constraints, respectively. Therefore, for a given level of uncertainty, the robust counterpart of these problems should not only provide protection in terms of optimality but also in terms of feasibility.

Adopting the ideas presented in the previous sections, the Robust B&S Budget Constrained PCStT (B-PCStT), is defined as:

$$ROPT_B = \min_{(\mathbf{x},\mathbf{y})\in\mathscr{T}} \left\{ \sum_{v\in V} p_v^-(1-y_v) + \beta_V^*(\Gamma_V) \,\middle|\, \sum_{e\in E} c_e^- x_e + \beta_E^*(\Gamma_E) \leq B \right\}.$$

According to the previous section, for a given description $\mathscr{T}$ of the deterministic problem, one can consider four possible ways to derive a valid MIP model for this robust counterpart of the problem. The objective function can be modeled using compact or Benders robust constraints. But also the budget constraint can be modeled using one or the other variant. To model the budget constraint using Benders reformulation, we will need to insert the following family of inequalities into the MIP:

$$\sum_{e\in E} c_e^- x_e + \sum_{e\in S} d_e x_e \leq B \quad \forall S \subseteq E, \;\; |S| \leq \Gamma_E \tag{2.27}$$

These cuts are similar to (2.22) (see also [Fischetti and Monaci, 2012]).

Similarly, the Robust B&S Quota Constrained PCStT (Q-PCStT), is defined as:

$$ROPT_Q = \min_{(\mathbf{x},\mathbf{y})\in\mathscr{T}} \left\{ \sum_{e\in E} c_e^- x_e + \beta_E^*(\Gamma_E) \,\middle|\, \sum_{v\in V} p_v^-(1-y_v) + \beta_V^*(\Gamma_V) \leq Q \right\}$$

and again one can consider four ways of deriving a MIP model for this problem.

2.4. Branch-and-Cut Algorithms

The MIP formulations considered throughout this chapter cannot be solved directly, even for small instances, since there is an exponential number of connectivity constraints of type (2.4) and, in addition, if Benders cuts are used to model the protection functions, there is also an exponential number of robustness cuts to be considered. Consequently, more sophisticated and specific techniques should be designed and implemented to solve these models.

In this section we propose three ways to develop a branch-and-cut (B&C) algorithm for solving the robust PCStT and its budget and quota constrained variants. We will explain the main ideas for solving the RPCStT, and a similar scheme needs to be applied in order to solve the B-RPCStT or the Q-RPCStT.

B&C with Compact Robust Constraints (Compact): In this approach, we are solving the MIP model in which the deterministic model (2.3)-(2.8) is extended by a compact set of auxiliary variables and constraints (2.17)-(2.19) that model the protection functions (see Section 2.3.2). In this approach, only connectivity constraints will be separated within a B&C framework. The separation algorithm is an adaptation

of the exact approach presented in [Ljubić et al., 2006]. The MIP initially contains all variables and the constraints (2.3), (2.5)-(2.8). In addition, we explicitly insert the subtour elimination constraints of size 2:

$$x_{ij} + x_{ji} \leq y_i, \quad \forall i \in V_{SA} \setminus \{r\}, \quad j \in \delta^+(i)$$

to avoid too frequent calls of the maximum flow procedure. The connectivity constraints are separated within the B&C framework by means of the maximum flow algorithm given in [Cherkassky and Goldberg, 1995]. This separation randomly selects a terminal $i \in V_{p_i>0}$, calculates the maximum flow between an artificial root and i and inserts the corresponding (2.4), if violated. Instead of adding a single violated cut per iteration, we use *nested*, *back-flow* and *minimum cardinality* cuts to add as many violated cuts as possible [see for details Koch and Martin, 1998]. We restrict the number of inserted cuts within each separation callback to 25.

B&C with Separation of Robustness Cuts (R-Cuts): In this approach, we consider the MIP model in which protection functions are modeled by means of robustness cuts of type (2.22) and (2.23). We initialize the model using only the following bounds for Θ and Λ variables:

$$\Theta \leq \sum_{e \in S^*_{\Gamma_E}} d_e \quad \text{and} \quad \Lambda \leq \sum_{e \in S^*_{\Gamma_V}} d_v$$

where $S^*_{\Gamma_E}$ ($S^*_{\Gamma_V}$) is the subset of edges (nodes) containing Γ_E (Γ_V) edges (nodes) with largest deviations. The correctness of the bounds comes from the fact that both Θ and Λ accumulate the deviations of the nominal costs for the solution edges and for the nodes left out of the solution, respectively.

The separation problem for robustness cuts of type (2.22) is as follows: given the current LP solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\Theta}, \hat{\Lambda})$, find a set $S \subseteq E$ such that $|S| \leq \Gamma_E$ and $\sum_{e \in S} d_e \hat{x}_e$ is a maximum. Assume that a subset of edges S^* satisfies these properties. If $\sum_{e \in S^*} d_e \hat{x}_e > \hat{\Theta}$, the current LP solution violates constraint (2.22) and hence we insert the cut $\Theta \geq \sum_{e \in S^*} d_e x_e$ into the model. To determine the set S^*, we associate with each edge $e \in E$ a *weight* $w_e = d_e \hat{x}_e$. The separation problem consists in finding the subset of edges of size Γ_E with the maximum weight, which can be done in $O(|E|)$ time [see Fischetti and Monaci, 2012]. This idea was first implemented in [Fischetti and Monaci, 2012] in the context of robust optimization for linear and integer programming under uncertainty. The authors report a remarkable improvement in the running times when using these robustness cuts in the formulations and separation framework instead of a compact formulation.

Robustness cuts are added on the fly, within the B&C framework, i.e., we are not waiting to find an LP-solution that satisfies all the connectivity cuts. Instead, within

one separation callback, we insert all the violated connectivity cuts detected plus the (one or two) robustness cuts associated with (2.22) and (2.23).

B&C with Separation of Robust Compact Constraints (C-Cuts): We have observed that not all of compact constraints associated with a protection function are tight in an optimal solution. On the other hand, when the number of nodes and/or edges increases, the size of the compact block of constraints associated with $\beta_V^*(\Gamma_V)$ or $\beta_E^*(\Gamma_E)$ may become a bottleneck of the implementation. Therefore, instead of inserting all these constraints at once, we propose to separate them within a B&C framework. We start with an LP model in which there are no constraints associated with robust variables, except the following ones:

$$\sum_{e\in E} h_e + \theta \leq \sum_{e\in S^*_{\Gamma_E}} d_e \quad \text{and} \quad \sum_{v\in V} k_v + \lambda \leq \sum_{v\in S^*_{\Gamma_V}} d_v$$

and

$$\theta \leq d_{\Gamma_E} \text{ and } \lambda \leq d_{\Gamma_V}$$

where $S^*_{\Gamma_E}$ ($S^*_{\Gamma_V}$) has been described above, d_{Γ_E} is the Γ_E-th largest edge cost deviation and d_{Γ_V} the Γ_V-th largest node prize deviation (see Lemma 1 in Álvarez-Miranda et al. [2013d]).

The separation of constraints (2.17) can be stated as follows: given an LP-solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{h}}, \hat{\mathbf{k}}, \hat{\theta}, \hat{\lambda})$, find a set $\hat{E} \subseteq E$ of maximum cardinality for which $\hat{h}_e + \hat{\theta} < d_e \hat{x}_e$ $\forall e \in \hat{E}$ and insert the corresponding constraints of type (2.17). Of course, the separation of constraints (2.17) and (2.18) can be performed in $O(|E|)$ and $O(|V_{p_i>0}|)$ time, respectively.

Within the B&C framework we first separate all the connectivity constraints (2.4), and once we find an optimal LP solution, we find a subset of violated compact robust constraints, and insert all of them at once into the current LP.

2.5. Computational Results

Benchmark Instances In our computational experiments four sets of benchmark instances have been tested: `C`, `D`, `K` and `P`. These instances have been used in most of the papers discussing algorithm design for the PCStT [Lucena and Resende, 2004, Ljubić et al., 2006] and [Salles da Cunha et al., 2009]. Instances of group `P` were introduced in [Johnson et al., 2000] – they are unstructured and designed to have constant node degree and a constant prize/cost ratio. Group `K` are randomly generated geometric graphs designed to have a structure similar to street maps [Johnson et al., 2000]. Groups `C` and `D` were presented in [Canuto et al., 2001]. These two groups of

instances are derived from the instances of the Steiner tree problem provided in the OR-Library [Beasly, 1990].

Groups `C` and `D` are composed by 40 instances each with 500 and 1000 nodes, respectively, and the number of edges goes from 625 to 12500 and from 1250 to 25000, respectively. Group `P` is composed by 11 instances with 100, 200 and 400 nodes and the number of edges goes from 300 to 1185. Finally, group `K` is formed by 23 instances with 100, 200 and 400 nodes and the number of edges goes from 344 to 1493. For more details on the description of instances see the first four columns of Table 2.9.

Given an original instance `Prob` for the deterministic PCStT, the corresponding robust instance, named `Prob`-α-β, ($\alpha \in [0,1]$ and $\beta \in [0,1]$) is derived as follows: the number of nodes and edges are left unchanged. Lower limits, c_e^- and p_v^-, for intervals defining edge costs and node prizes are set to the corresponding deterministic values c_e and p_v, i.e., $c_e^- = c_e \ \forall e \in E$ and $p_v^- = p_v \ \forall v \in V$. The upper limit of edge costs, c_e^+, is set to $(1+\alpha)c_e \ \forall e \in E$. Similarly, the upper limit of node prizes, p_v^+, is set to $(1+\beta)p_v \ \forall v \in V$. Parameters α and β allow to control the width of the corresponding intervals and, consequently, the level of uncertainty of the problems. For most of our experiments we consider ($\alpha = 0.05, \beta = 0.05$) (unless mentioned otherwise), which means that both edge costs and node prizes present a deviation equal to the 5% of their corresponding nominal values. In preliminary experiments we also considered deviations of 1% and 2.5%, however, these instances did not allow to clearly show the impact of considering higher levels of uncertainty on both the solution structure and the algorithm performance. A deviation of 5% is in the middle of the values considered in most of the literature which range from 1% up to 10%; [see Bertsimas and Sim, 2003, 2004, Fischetti and Monaci, 2012], among other papers. Almost the same criterion to generate interval data instances is also used in [Ben-Tal et al., 2004, Klopfenstein and Nace, 2012, Lee et al., 2012] and [Solyali et al., 2012].

Machine and Implementation All the experiments were performed on a Intel Core2 Quad 2.33 GHz machine with 3.25 GB RAM, where each run was performed on a single processor. The Branch-and-cut algorithms were implemented using CPLEX 12.2 and Concert Technology. All CPLEX parameters were set to their default values, except the following ones: (i) Branching: we set the highest branching priorities to variables y_v, $v \in V_{p_i>0}$; (ii) Emphasis: this parameter was set to *optimality*. (iii) Maximum Running Time was set to 500 seconds.

In the following tables and figures, the running times are expressed in CPU seconds.

2.5.1 Results for the RPCStT

Reduction Tests Reduction tests for the deterministic PCStT have been implemented in [Canuto et al., 2001, Lucena and Resende, 2004, Ljubić et al., 2006] and [Uchoa,

2007]. It has been demonstrated that the utilization of some of these preprocessing procedures can lead to remarkable improvements of the algorithmic performance. For our interval data instances we have adapted one of these reduction tests, which is described in the following. ***Robust Least-cost Test*** Let $\mathrm{SP}_{ij}(\Gamma_E)$ be the cost of the B&S robust shortest path between a pair of nodes i and j calculated for Γ_E in G. If there is an edge e connecting i and j such that $\mathrm{SP}_{ij}(\Gamma_E) \leq c_e^-$, then edge e can be eliminated from G.

Since the calculation of $\mathrm{SP}_{ij}(\Gamma_E)$ requires $O(m)$ shortest path calculations [see Bertsimas and Sim, 2003], in our implementation we have used only a weaker variant of this test in which $\mathrm{SP}_{ij}(\Gamma_E)$ is replaced by $\mathrm{SP}_{ij}(|E|)$. Although somehow conservative, this reduction criterion provides a unique reduced graph valid for any value of $\Gamma_E < |E|$ when solving the RPCStT or any of its variants. For larger instances, the reduced graphs have less than 50% of the original number of edges. It is important to observe that applying this test requires only a few seconds even for large instances. It turned out that the other robust reduction tests cannot be easily derived from their deterministic counterparts – an illustrative example is a degree two test on a potential Steiner node. After merging two edges and two intervals into one, we basically obtain a new edge whose interval contains an extra break point that is needed to model 0, 1 or 2 deviations from the nominal edge costs.

2.5.1.1 The Price of Robustness

As mentioned above, the *Price of Robustness* corresponds to the increase of the cost of a robust solution with respect to the nominal cost when increasing the level of robustness, i.e., when increasing the values of Γ_E and Γ_V. For each group of instances, we report in Table 2.1 the minimum, mean and maximum values, computed over all the instances of the corresponding setting and group, of the relative increase of the cost of the solutions, $\Delta ROPT(\%)$, for different combinations of Γ_E and Γ_V. For each instance and setting, $\Delta ROPT(\%)$ is defined as $(ROPT - OPT) * 100/OPT$, where $ROPT$ and OPT are the corresponding optimal values[1] of the robust and of the nominal solution, respectively. For each instance, we consider 16 settings obtained by combining the 16 pairs of $\Gamma_V, \Gamma_E \in \{0, 5, 20, 50\}$. Since we chose $(\alpha = 0.05, \beta = 0.05)$ for generating the instances, we would expect $\Delta ROPT(\%)$ to be always not greater than 5%. The difference between 5% and $\Delta ROPT(\%)$ can be seen as the *level of protection* provided by the robust model and the chosen values of Γ_E and Γ_V. From the information reported in Table 2.1, two main observations can be made: (i) the B&S model seems to provide more protection against uncertainty to groups `C` and `D` than to groups `K` and `P`, and (ii) in the case of groups `C`, `D` and `P`, parameter Γ_E has a stronger impact on

[1] In case that none of the exact approaches was able to find an $ROPT$ optimal solution within the specified time limit, we used the best known upper bound to calculate $\Delta ROPT(\%)$, which is a good approximation considering the quality of the gaps.

the price of robustness than Γ_V, while in the case of group `K`, parameter Γ_V is the one with a stronger influence on the price of robustness.

Both observations can be explained by considering the relation between the particular values of Γ_E and Γ_V and the size (i.e., the number of edges and nodes) of the obtained solutions, whose statistics are given in the last 6 columns of Table 2.9. In the case of `C`, `D` and `P` instances, the average number of edges in the solutions is almost always greater than the chosen values of Γ_E, which means that in many cases the cost of some edges in the solution will remain within the corresponding lower limit. This explains why, for a given Γ_V, the average value of $\Delta ROPT(\%)$ does not reach 5% even when $\Gamma_E = 50$. When comparing the average number of all customers and the average number of customers connected by the solutions for groups `C` and `D` (see Table 2.9), it can be easily seen that many customers are taken into the solution. This means that the number of non-connected customers, i.e., those nodes whose prizes and deviations are added in the objective function, is generally smaller than $\Gamma_V = 50$. This explains why, for a given value of Γ_E, a variation of Γ_V does not strongly increase the value of $\Delta ROPT(\%)$. In the case of group `P`, particularly for instances `P400.`$\{0-4\}$, the ratio between the connected versus non-connected customers might be a little bit smaller than in the case of `C` and `D`, which explains why $\Delta ROPT(\%)$ can be as high as 4.44% for the maximum values of Γ_E and Γ_V.

In contrast to what happens for `C`, `D` and `P` groups, in the case of instances of group `K`, most of the solutions are on average relatively small, which explains why the mean value of $\Delta ROPT(\%)$ can reach almost 5% for large values of Γ_E and Γ_V. The particular Euclidean geometric topology of these instances might also give hints to understand these results; nodes are "locally connected" within a neighborhood, so despite the increase in the prize of non-connected nodes these are not reached because there are no direct connections between a given component and these attractive nodes, which increases the overall cost of the solution.

To look deeper into the impact of Γ_E and Γ_V on the structure of the solutions, Figures 2.1(a) and 2.1(b) show two optimal solutions obtained for the instance `K400.4-0.05-0.05` for $\Gamma_E = 0$, $\Gamma_V = 50$ and for $\Gamma_E = 50$, $\Gamma_V = 0$, respectively. The $ROPT$ value of the first solution is 403 036 while that of the second is 393 919, which represents a relative difference of only 2.3% although the structure of the solutions are quite different; just as a reference, the value of $ROPT$ for $\Gamma_E = 0$ and $\Gamma_V = 0$ is 389 451. These two figures put in evidence the capability of the B&S model to produce very different robust solutions for different levels of conservatism, and, at the same time, to provide a guarantee of protection in terms of the relative increase of the solution cost. This important feature of the model offers the possibility to choose a solution according to the perception of the uncertain state of the decision-making environment.

		$\Delta ROPT(\%)$											
		C			D			K			P		
Γ_E	Γ_V	min	mean	max	min	mean	max	min	mean	max	min	mean	max
0	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	5	0.00	0.71	5.00	0.00	0.62	5.00	0.67	2.02	3.74	0.36	0.66	1.74
0	20	0.00	0.95	5.00	0.00	0.81	5.00	1.79	3.54	5.00	0.45	0.94	1.89
0	50	0.00	1.11	5.00	0.00	0.93	5.00	1.79	4.07	5.00	0.45	0.98	1.89
5	0	0.00	0.78	3.00	0.00	0.60	3.10	0.00	0.39	1.73	0.46	0.96	1.92
5	5	0.15	1.49	5.00	0.03	1.22	5.00	0.98	2.42	4.41	0.89	1.65	2.87
5	20	0.15	1.72	5.00	0.05	1.41	5.00	2.50	3.99	5.00	1.25	1.93	3.03
5	50	0.15	1.89	5.00	0.06	1.54	5.00	2.67	4.52	5.00	1.25	1.96	3.03
20	0	0.00	1.82	5.00	0.00	1.46	5.00	0.00	0.65	1.98	1.38	2.45	4.00
20	5	0.43	2.53	5.00	0.19	2.16	5.00	0.98	2.72	4.84	1.83	3.18	4.74
20	20	0.43	2.77	5.00	0.20	2.34	5.00	3.05	4.30	5.00	2.23	3.47	5.00
20	50	0.43	2.93	5.00	0.19	2.48	5.00	3.78	4.88	5.00	2.23	3.50	5.00
50	0	0.00	2.39	5.00	0.00	1.88	5.00	0.00	0.73	2.51	2.50	3.34	4.44
50	5	0.72	3.12	5.00	0.47	2.62	5.00	0.98	2.81	4.84	3.01	4.10	5.00
50	20	0.97	3.36	5.00	0.47	2.81	5.00	3.05	4.40	5.00	3.49	4.40	5.00
50	50	0.98	3.53	5.00	0.47	2.95	5.00	4.92	4.99	5.00	3.49	4.44	5.00

TABLE 2.1: Basic statistics of $\Delta ROPT(\%)$ (*Price of Robustness*) for different values of Γ_E and Γ_V, groups C, D, K and P

(a) $\Gamma_E = 0$ and $\Gamma_V = 50$.

(b) $\Gamma_E = 50$ and $\Gamma_V = 0$.

FIGURE 2.1: Optimal solutions for the instance K400.4-0.05-0.05

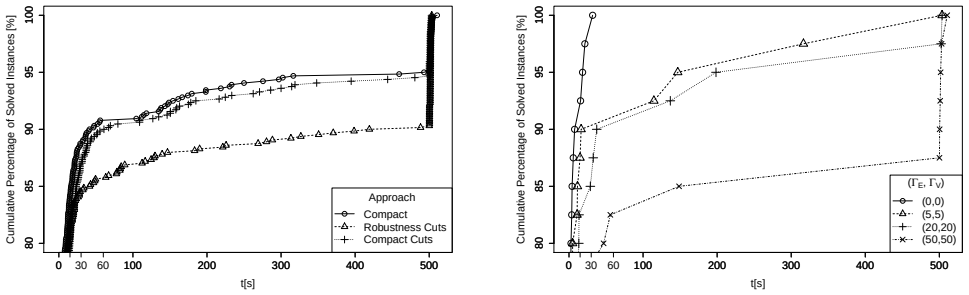
2.5.1.2 Algorithmic Performance

As mentioned before, to solve the RPCStT we used three different B&C strategies: *Compact*, *R-Cuts* and *C-Cuts*. The performance of these different approaches depends not only on the instance group, and the size of the instances therein, but also on the particular selection of the parameters Γ_E and Γ_V.

Figure 2.2(a) shows the cumulative percentage of instances of group C solved to optimality within a given time ranging from to 0 to 500 seconds. We compare the three different approaches for 16 settings across all values of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$ and across all 40 instances of group C. From this figure we conclude that *Compact* seems to be the best approach for this group since a larger percentage of instances can be solved within smaller running times than those of the other two approaches. However, we also observe that *C-Cuts* behaves similarly. To solve 90% of the instances, *Compact* requires less than 30 seconds, *C-Cuts* slightly more than 30 seconds, and *R-Cuts* more than 400 seconds. To solve an extra 5% of instances, *Compact* requires about 300 seconds, while both *C-Cuts* and *R-Cuts* reach the time limit (500 seconds). Overall, *R-Cuts* presents a performance clearly worse than that of *C-Cuts*, and *C-Cuts* is slightly outperformed by *Compact*.

To complement the previous analysis, Figure 2.2(b) shows the cumulative percentage of instances solved by *Compact* considering four different combinations of Γ_E and Γ_V, for the 40 instances of group C. We can see that for the nominal case ($\Gamma_E = 0$ and $\Gamma_V = 0$), *Compact* can solve to optimality all the instances within just a few seconds. However, when increasing the values of Γ_E and Γ_V, the running times begin to increase quickly, and even for $\Gamma_E = 5$ and $\Gamma_V = 5$ there are a few instances that cannot be solved to optimality within 500 seconds. Further increasing of the values of Γ_E and Γ_V produces a severe deterioration of the algorithmic performance. For example, when taking $\Gamma_E = 50$ and $\Gamma_V = 50$, almost 15% of the instances can not be solved to optimality within the given time limit. Hence, this is another aspect of the *price of robustness*: obtaining more robust solutions, in terms that they provide more protection against uncertainty, requires willingness to accept higher running times to calculate the optimal solutions.

Tables 2.2-2.5 provide more detailed statistics for the four groups of instances and the three algorithmic approaches. On the left hand side, for each of the approaches, we report the number of instances that are solved to optimality. On the center, statistics on the running times are reported considering only those instances that can be solved to optimality within 500 seconds by all three approaches. On the right hand side, we provide statistics for the remaining problems (i.e, for those that can not be solved to optimality by at least one of the approaches). For each approach, we report statistics on the final gap (calculated with respect to the corresponding lower bound) over these problems. These statistics indicate that, for the three approaches and across the four



(a) Comparing the three approaches over 16 combinations of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$.

(b) *Compact*, four (Γ_E, Γ_V) combinations.

FIGURE 2.2: Cumulative percentage of the total number of solved instances of group `C` within 500 seconds

		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	608/640	0.063	0.693	6.210	275.600	0.000	0.009	0.038	0.618
R-Cuts	577/640	0.031	0.773	12.380	419.200	0.000	0.223	0.287	0.817
C-Cuts	605/640	0.047	1.320	7.106	318.800	0.000	0.019	0.024	0.297

TABLE 2.2: Algorithmic performance statistics for group `C`

		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	553/640	0.156	5.719	23.690	407.200	0.000	0.029	0.048	0.327
R-Cuts	513/640	0.141	4.484	27.690	476.600	0.000	0.095	0.132	1.745
C-Cuts	544/640	0.141	8.297	30.430	402.600	0.000	0.047	0.060	0.609

TABLE 2.3: Algorithmic performance statistics for group `D`

		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	368/368	0.047	1.156	10.680	133.000	0.000	0.000	0.000	0.000
R-Cuts	365/368	0.047	0.766	13.270	476.500	0.020	0.059	0.064	0.114
C-Cuts	368/368	0.310	0.719	11.190	197.400	0.000	0.000	0.000	0.000

TABLE 2.4: Algorithmic performance statistics for group `K`

family of instances, there is a relatively small number of cases (given by a particular combination of Γ_E and Γ_V) that are intractable by the used algorithms. Although optimality is not always verified (especially by *R-Cuts*), the quality of the solutions obtained when reaching the time limit is remarkably good, as it can be seen from the statistics on the final gaps. The values of the median and the average of the gaps in Tables 2.2-2.5 indicate that the chosen formulations and approaches guarantee that solutions of a good quality can be obtained within a reasonable running time, in case that are not proven to be optimal. This observation complements the analysis of Figure 2.2(b).

Further information about algorithmic performances is presented in Tables 2.10-2.13.

		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	176/176	0.031	0.180	0.974	8.703	0.000	0.000	0.000	0.000
R-Cuts	176/176	0.031	0.297	4.622	84.200	0.000	0.000	0.000	0.000
C-Cuts	176/176	0.031	0.242	1.731	20.380	0.000	0.000	0.000	0.000

TABLE 2.5: Algorithmic performance statistics for group P

The evolution, over time, of the gap between lower and upper bounds in the B&C tree for a subset of the most difficult instances is also analyzed in Section 2.8 (see Figures 2.11 and 2.12).

The overall superiority of *Compact* might be explained by the fact that from the beginning of the optimization process the underlying LP contains complete information regarding the robustness of the solution. Although at the root node we obtain tight bounds even if we consider *R-Cuts* or *C-Cuts*, after starting the branching process, a large sequence of re-optimizations (each time that a set of cuts is inserted we need to solve the underlying LP) deteriorates the optimization process entailing higher running times. In particular, in the case of *R-Cuts*, the convergence of the values of Θ and Λ becomes slower, i.e., more cuts have to be added and more branch-and-bound nodes have to be enumerated in order to reach optimal values. The combination of these two elements is responsible for the poor performance of this approach with respect to the others. A similar observation is pointed out in [Fischetti and Monaci, 2012] when analyzing the performances of the compact formulation and robustness cuts to solve generic MIP problems.

2.5.1.3 Influence of α and β

As mentioned before, robust instances were created from original instances using ($\alpha = 0.05, \beta = 0.05$). In order to provide a more complete analysis of the robust model and the proposed approaches, we have also generated instances considering three additional combinations taken from $\alpha, \beta \in \{0.05, 0.10\}$. For these experiments, we have considered groups C and K. We first present results regarding the *Price of Robustness* and then results regarding the performance of the proposed approaches.

Price of Robustness It is clear that if the interval width is increased (by augmenting α and/or β), the presence of uncertainty also increases; therefore, the *price of robustness* paid for a given level of uncertainty will be greater.

In Table 2.6, similar to Table 2.1, we report statistics of the relative increase of the objective function value ($\Delta ROPT(\%)$), when solving the RPCStT on instances of group C, for different values of Γ_E and Γ_V and considering the four resulting combinations of α and β. As expected, the value of $\Delta ROPT(\%)$ increases when increasing the values of α and β. In the four cases, one can recognize a common pattern: the

value of Γ_E is more responsible for the increase of $ROPT$ than Γ_V. As explained before for the ($\alpha = 0.05, \beta = 0.05$) case, this is mainly due to the relation between the particular values of Γ_E and Γ_V and the size (number of edges and nodes) of the corresponding solutions; on average, the produced solutions have a quite similar size regardless of the values of α and β (see Table 2.14). Roughly speaking, the solutions are on average comprised by 100 edges and a few nodes with positive prize are left out of the tree, which means that increasing α (uncertainty on the edges) has more impact on the solution cost than increasing β (uncertainty on the nodes). On the contrary, for instances of group K, the value of β, along with the value of Γ_V, has more influence on $\Delta ROPT(\%)$ (see Table 2.15); this can be concluded by taking into account the same arguments presented before for the ($\alpha = 0.05$, $\beta = 0.05$) case.

In summary, we can see that the effect produced on the *price of robustness* by different values of α and β follows a common pattern determined by the ratio between the size of the produced solutions and the corresponding values of Γ_E and Γ_V.

		$\Delta ROPT(\%)$											
		$\alpha = 0.05, \beta = 0.05$			$\alpha = 0.05, \beta = 0.10$			$\alpha = 0.10, \beta = 0.05$			$\alpha = 0.10, \beta = 0.10$		
Γ_E	Γ_V	min	mean	max	min	mean	max	min	mean	max	min	mean	max
0	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	5	0.00	0.71	5.00	0.00	1.42	10.00	0.00	0.71	5.00	0.00	1.42	10.00
0	20	0.00	0.95	5.00	0.00	1.89	10.00	0.00	0.95	5.00	0.00	1.89	10.00
0	50	0.00	1.11	5.00	0.00	2.21	10.00	0.00	1.11	5.00	0.00	2.21	10.00
5	0	0.00	0.78	3.00	0.00	0.78	3.00	0.00	1.55	6.00	0.00	1.55	6.00
5	5	0.15	1.49	5.00	0.13	2.20	10.00	0.26	2.26	6.00	0.26	2.98	10.00
5	20	0.15	1.72	5.00	0.13	2.67	10.00	0.26	2.50	6.00	0.26	3.44	10.00
5	50	0.15	1.89	5.00	0.13	2.99	10.00	0.26	2.66	6.00	0.26	3.77	10.00
20	0	0.00	1.82	5.00	0.00	1.81	5.00	0.00	3.60	10.00	0.00	3.60	10.00
20	5	0.43	2.53	5.00	0.43	3.24	10.00	0.88	4.35	10.00	0.90	5.06	10.00
20	20	0.43	2.77	5.00	0.43	3.71	10.00	0.88	4.59	10.00	0.90	5.54	10.00
20	50	0.43	2.93	5.00	0.43	4.04	10.00	0.88	4.76	10.00	0.90	5.87	10.00
50	0	0.00	2.39	5.00	0.00	2.38	5.00	0.00	4.71	10.00	0.00	4.71	10.00
50	5	0.72	3.12	5.00	0.99	3.83	10.00	0.91	5.47	10.00	1.45	6.20	10.00
50	20	0.97	3.36	5.00	0.99	4.31	10.00	1.91	5.72	10.00	1.99	6.69	10.00
50	50	0.98	3.53	5.00	0.99	4.64	10.00	1.92	5.89	10.00	1.99	7.03	10.00

TABLE 2.6: Basic statistics of $\Delta ROPT(\%)$ (*Price of Robustness*) for different values of Γ_E and Γ_V, considering different values of α and β, group C

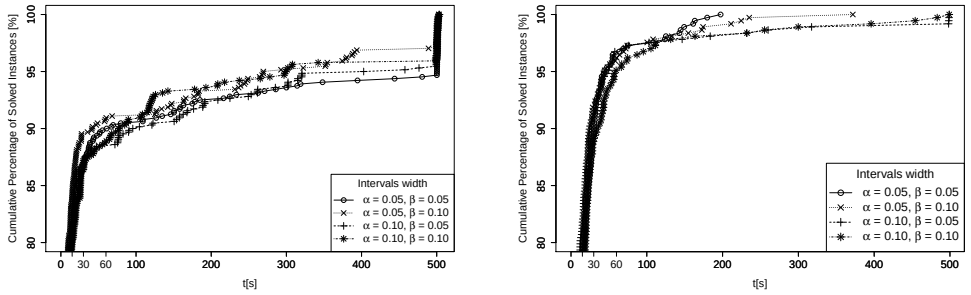
Algorithmic Performance In Table 2.7, similar to Tables 2.2-2.5, we report statistics for group C with $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$ and four combinations of α and β. The first observation is that the proposed approaches behave quite similarly for the four pairs of α and β values: the number of instances solved to optimality is similar in each case, the running times are comparable and also the attained gaps are alike. The second observation is that regardless of the values of α and β, *R-Cuts* is the the approach with the poorest performance. Regarding the other two algorithms, both are quite effective for all pairs (α, β), but looking at the number of instances solved to optimality we see that *C-Cuts* slightly outperforms *Compact*. Also in the case of group

K (see Table2.16), α and β only slightly influence the algorithmic performance of the considered approaches, in which case, *Compact* marginally beats *C-Cuts*.

			Running times (t $\leq$ 500 s)				Gaps (%) (t $>$ 500 s)			
α, β	Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
0.05, 0.05	*Compact*	608/640	0.063	0.693	6.210	275.600	0.000	0.009	0.038	0.618
	R-Cuts	577/640	0.031	0.773	12.380	419.200	0.000	0.223	0.287	0.817
	C-Cuts	605/640	0.047	1.320	7.106	318.800	0.000	0.019	0.024	0.297
0.05, 0.10	*Compact*	603/640	0.063	1.172	10.030	461.500	0.000	0.007	0.026	0.540
	R-Cuts	582/640	0.063	0.875	15.610	461.200	0.000	0.295	0.444	4.264
	C-Cuts	609/640	0.063	1.125	7.142	248.200	0.000	0.007	0.034	0.372
0.10, 0.05	*Compact*	592/640	0.063	1.195	6.667	145.000	0.000	0.037	0.122	0.967
	R-Cuts	572/640	0.047	1.016	13.800	465.900	0.000	0.456	0.528	1.243
	C-Cuts	601/640	0.063	1.297	7.459	402.800	0.000	0.008	0.115	1.054
0.10, 0.10	*Compact*	602/640	0.063	1.734	11.930	307.100	0.000	0.010	0.121	0.926
	R-Cuts	587/640	0.063	1.312	23.470	496.000	0.000	0.464	0.585	1.345
	C-Cuts	608/640	0.078	1.609	10.830	318.900	0.000	0.010	0.115	0.922

TABLE 2.7: Algorithmic performance statistics for different combinations of α and β, group C

More details regarding the influence of α and β on the algorithmic performance of *C-Cuts* are shown in Figures 2.3(a) and 2.3(b), where the cumulative percentage of solved instances within a given running time (that goes from 0 up to the time limit of 500 seconds) is shown for group C and K, respectively, for the four combinations of α and β. In Figure 2.3(a) one can see that the four curves are quite close to each other, reinforcing the conclusions obtained from Table 2.7 regarding the independence of the algorithms with respect to α and β when solving instances of group C. In the case of K instances, in Figure 2.3(b) we see that when increasing the values of α and β some outliers appear and very few problems, 4 out of 386 in the case of ($\alpha = 0.10$, $\beta = 0.05$) and 5 out of 386 in the case of ($\alpha = 0.10$, $\beta = 0.10$), cannot be solved to optimality within the time limit of 500 seconds (among these 9 problems, gaps of at most 0.5% are reached). Equivalent conclusions can be drawn for *Compact* when analyzing the reported results in Figures 2.8(a) and 2.8(b). Hence, α and β have both a very limited influence on the algorithmic performance for the considered instances.



(a) Comparing performance of *CCuts* for different α and β, group C.

(b) Comparing performance of *CCuts* for different α and β, group K.

FIGURE 2.3: Cumulative percentage of the total number of solved instances of groups C and K within 500 seconds for different values of α and β (*C-Cuts*)

2.5.2 Results for the Robust B-PCStT

In order to complement the analysis of the computational results presented for the RPCStT, we developed a similar experimental framework for the robust counterpart of the B-PCStT which, as we mentioned before, is an important variant of the PCStT. Because of the restriction on the length of the paper, we only present results obtained for group `P` and for eleven instances of group `K` (`K100.`$\{0-10\}$) considering ($\alpha = 0.05, \beta = 0.05$).

As part of the Robust B-PCStT model, it is necessary to provide a given *budget* B, which represents the maximum allowed sum of the edge costs, considering uncertainty, that the decision maker is willing to pay. Since different instances, even within the same group, have different cost structures, a given value of B might not be suitable for all of them, so it is necessary to establish a fair criterion to define appropriate values of B. In order to do so, we set the budget to be a percentage of a potential *maximum robust budget* value $B_{\max}$, associated with each particular instance. If the input graph is connected, $B_{\max}$ represents the cost of the optimal robust Steiner tree in which all the customers are connected and the cost of at most Γ_E edges is allowed to deviate from its nominal value. If the input graph is not connected, $B_{\max}$ is the cost of the robust Steiner sub-tree connecting as many customers as possible. To calculate the value of $B_{\max}$, we set the node prizes to a big-M-value and Γ_E to 50, and use one of the algorithms for the RPCStT proposed before. The selected value of $\Gamma_E = 50$ ensures feasibility for all the other values of Γ_E as long as they are not greater than 50, which is the maximum value we consider for this parameter in our experiments. We note that it was necessary to set the node prizes to a big-M-value, instead of simply adding the constraints $y_v = 1\ \forall v \in V_{p_v^- \geq 0}$ into the MIP model, because the considered instances are not necessarily connected.

B&C Variants Since there are more alternatives to formulate the Robust B-PCStT as a MIP, there are also more alternatives to design a B&C algorithm. Besides the separation of the connectivity inequalities, we have considered four alternatives to manage the different types of robust constraints: (i) B&C using the compact robust constraints of type (2.17) and (2.18) (*Compact*); (ii) B&C with separation of the robustness cuts of type (2.27) and type (2.23) including variable Λ in the objective function (*R-Cuts*); (iii) B&C with separation of the robust compact constraints of type (2.17) and (2.18) (*C-Cuts*); (iv) B&C with separation of the robustness cuts of type (2.27) but including all the compact constraints of type (2.18) (*R-Cuts+Compact*).

2.5.2.1 The Price of Robustness

In Figure 2.4 the value of $ROPT_B$ is reported for different values of the budget B and for four different combinations for Γ_E and Γ_V for instance `K100.10-0.05-0.05`. As

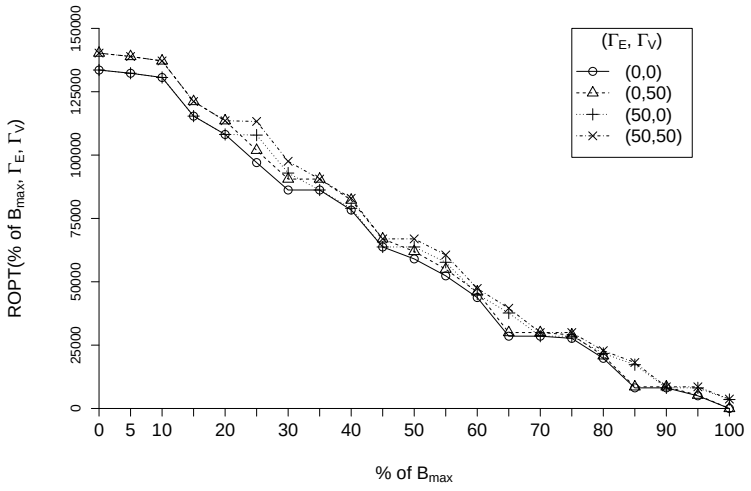
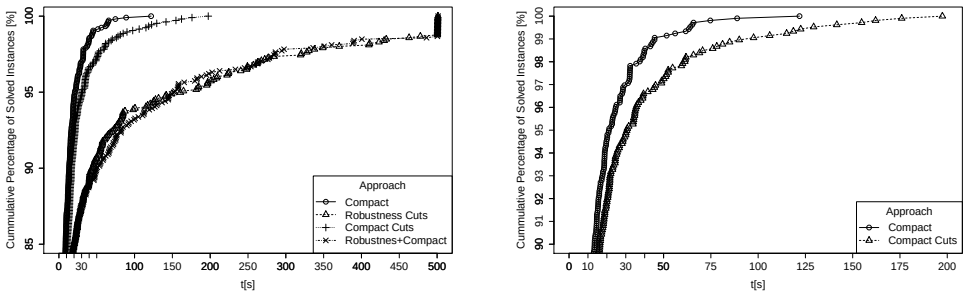


FIGURE 2.4: Values of $ROPT_B$ for different values of B, Γ_E and Γ_V, instance `K100.10-0.05-0.05`

expected, and independently of the values of Γ_E and Γ_V, there is a monotone decrease of the value of the objective function (recall that this is the sum of the prizes of the nodes that are not connected) when increasing the value of the available budget. When considering a particular value of B, we observe that the differences of $ROPT_B$, among different values of Γ_E and Γ_V, do not present a clear pattern as in the case of the RPCStT. This can be explained by the fact that Γ_E is not included in the objective function but in the budget constraint, so it has an *indirect influence* on the objective function value. For example, when considering a budget given by 25% of $B_{\max}$, the four considered combinations produce significantly different values of $ROPT_B$; while for a budget given by 90% of $B_{\max}$, the four values of $ROPT_B$ are almost the same. Another characteristic that we can observe, is that for tight budgets (0% - 20%) the value of Γ_V has more impact on the model than Γ_E, while for large budgets (80% - 100%) it is just the opposite.

As this was the case for the RPCStT, the latter behaviors are related to the size of the corresponding optimal solution and to its interaction with the problem parameters B, Γ_E and Γ_V. For a tight budget, an optimal solution is made up of only a few edges and many customer nodes are left unconnected, which explains why increasing the value of Γ_V strongly increases the value of the objective function, while increasing Γ_E barely produces changes since only a few edges can be taken into account. On the other hand, for a large budget, most of the customers are connected and an increase of Γ_V might not significantly affect the value of $ROPT_B$, but increasing Γ_E will indeed strongly influence the value of $ROPT_B$ because the budget feasibility will enforce a solution of a smaller cardinality, i.e., it will be necessary to "disconnect" some customers and consequently the value of $ROPT_B$ will be increased. An example that illustrates these dependencies is shown in Figure 2.14.

2.5.2.2 Algorithmic Performance



(a) Comparing all four approaches.

(b) Comparing only *Compact* and *C-Cuts*.

FIGURE 2.5: Cumulative percentage of the average number of solved instances of group P within $t = 500$ seconds considering different values of $B \in \{0, 5, 10, \ldots, 95, 100\}B_{\max}\%$, and $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$

Figure 2.5(a) shows the cumulative percentage of instances of group P solved to optimality within time t with a time limit of 500 seconds, comparing the four approaches described above. In a more detailed plot (see Figure 2.5(b)), only *Compact* and *C-Cuts* are compared. We observe that *Compact* and *C-Cuts* are substantially better than the other two approaches (which are both based on the utilization of robustness cuts). For example, to solve 95% percent of the instances of group P, *Compact* needs less than 20 seconds, *C-Cuts* less than 30 seconds, while the other two approaches need almost 150 seconds to solve the same percentage of instances. Moreover, in a small number of cases (less than 2%), *R-Cuts* and *R-Cuts+Compact* reach the time limit without being able to find optimal solutions within the given time limit.

More details about the running times needed to solve the instances, as well as the statistics on the gaps for those instances where at least one of the approaches failed to find an optimal solution, are reported in Table 2.8. For group P, *Compact* is the best in terms of average running times. However, *C-Cuts* has a similar performance and provides better minimum and median running times, but a few outliers (see Figure 2.5(b)) deteriorate the overall statistics. The same table shows that, in 14 out of 1056 cases, *R-Cuts* and *R-Cuts+Compact* do not solve all the instances to optimality, but provide very small final gaps.

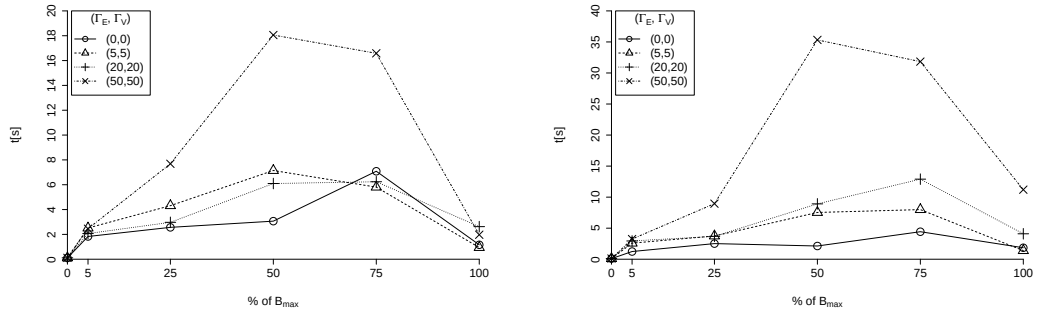
		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	1056/1056	0.031	0.500	3.621	65.831	0.000	0.000	0.000	0.000
R-Cuts	1042/1056	0.031	0.546	16.450	477.900	0.000	0.417	0.671	1.811
C-Cuts	1056/1056	0.015	0.453	4.791	162.100	0.000	0.000	0.000	0.000
R-Cuts+C	1042/1056	0.015	0.546	16.630	486.000	0.000	0.811	0.836	2.332

TABLE 2.8: Algorithmic performance statistics for group P (Robust B-PCStT)

Comparing the statistics for the RPCStT (see Table 2.5) with the results presented in Table 2.8 for the Robust B-PCStT, we may conclude that the Robust B-PCStT is

a considerably more complex problem. With an inclusion of a budget constraint the search for an optimal solution becomes a more difficult numeric task. The influence of the budget level on the algorithmic performance is shown in Figure 2.6, where the average running times over all the instances of group P are displayed for different budget levels and different values of Γ_E and Γ_V. Our first observation is that, independently of the values of the budget, increasing values of Γ_E and Γ_V directly influence the running times as it was the case of the RPCStT (see Figure 2.2). However, budgets levels set between $[0\%, 25\%]$ or $[75\%, 100\%]$ entail a better algorithmic performance than those taken from $[25\%, 75\%]$, and the influence of Γ_E and Γ_V is more accentuated in the latter case.

These relations between the running times and the budget levels can be explained by the way how different values of B reduce the space of feasible solutions. Tight budgets, let us say $[0\%, 25\%]$, strongly limit the set of feasible solutions, i.e., they usually correspond to small trees connecting a few customer nodes. Therefore, and considering that most of the P instances are sparse graphs (see Table 2.9), the optimal solutions can be quickly obtained. On the other hand, optimal solutions for large budgets, as those defined by $[75\%, 100\%]$, will be usually comprised by almost all the customer nodes; hence, solutions will be similar to the robust Steiner tree connecting those customers, which explains the decrease of the running times. On the contrary, for B chosen from $[25\%, 75\%]$ of $B_{\max}$, the combinatorial nature of the problem seems to have more influence on the algorithmic performance and there are more solutions, probably each of them with a very different structure, that might verify the optimality. Consequently, the computational effort to find an optimal solution is greater as illustrated in Figure 2.6.



(a) *Compact* approach. (b) *C-Cuts* approach.

FIGURE 2.6: Average running times for group P for different values of $B \in \{0, 5, 10, \ldots, 95, 100\}B_{\max}$, and the four selected combinations of (Γ_E, Γ_V)

A similar analysis of the results obtained for instances K100.$\{0-10\}$ (see Figure 2.13 and Table 2.17) lead us to conclude that *C-Cuts* is the best approach, both in terms of average and median running times. Once more, the robustness cuts based approaches do not seem to be competitive although they solve to optimality all instances within

the given time limit. Consequently, we may say that both *Compact* and *C-Cuts* are the most effective approaches for solving the robust B-PCStT for the considered instances.

2.6. Improved B&S Algorithms for the RPCStT and its Variants

Although in this work we presented MIP-based exact approaches for solving the robust counterparts of the PCStT and its variants, it is also possible to solve them by successively solving a finite number of classical instances of the corresponding problem. The next corollaries are derived from the more general results presented in [Álvarez-Miranda et al., 2013d]. To apply the results below, we assume that the customers and the edges are sorted in non-increasing order with respect to their deviations, i.e., $d_1 \geq d_2 \geq d_3 \ldots$ and the last deviations $d_{n'+1}$ (for the customers) and d_{m+1} (for the edges) are set to zero.

Lemma 1. Given $\Gamma_E \in \{0, \ldots, m\}$ and a given $\Gamma_V \in \{0, \ldots, n'\}$, the B&S Robust Counterpart of the GW-PCStT can be solved by solving $(n' - \Gamma_V + 2)(m - \Gamma_E + 2)$ nominal problems

$$ROPT(\Gamma_E, \Gamma_V) = \min_{\substack{a \in \{\Gamma_E, \ldots, m+1\} \\ b \in \{\Gamma_V, \ldots, n'+1\}}} G^{a,b},$$

where for $a \in \{\Gamma_E, \ldots, m+1\}$ and $b \in \{\Gamma_V, ..., n'+1\}$:

$$G^{a,b} = \Gamma_E d_a + \Gamma_V d_b + \min_{(\mathbf{x},\mathbf{y}) \in \mathscr{T}} \left(\sum_{e \in E} c_e^- x_e + \sum_{e=1}^{a} (d_e - d_a)\, x_e + \sum_{v \in V} p_v^- (1 - y_v) + \sum_{v=1}^{b} (d_v - d_b)\, (1 - y_v) \right).$$

Lemma 2. Given $\Gamma_E \in \{0, \ldots, m\}$ and $\Gamma_V \in \{0, \ldots, n'\}$, the Robust B&S B-PCStT can be solved by solving $(n' - \Gamma_V + 2)(m - \Gamma_E + 2)$ nominal problems

$$ROPT_B\,(\Gamma_E, \Gamma_V) = \min_{\substack{a \in \{\Gamma_E, \ldots, m+1\} \\ b \in \{\Gamma_V, \ldots, n'+1\}}} G_B^{a,b}, \tag{2.28}$$

where for $a \in \{\Gamma_E, ..., m+1\}$ and $b \in \{\Gamma_V, ..., n'+1\}$

$$G^{a,b} = \Gamma_V d_b + \min_{(\mathbf{x},\mathbf{y}) \in \mathscr{T}} \left(\sum_{v \in V} p_v^- (1 - y_v) + \sum_{v=1}^{b} (d_v - d_b)(1 - y_v) \,\middle|\, \sum_{e \in E} c_e^- x_e + \sum_{e=1}^{a} (d_e - d_a) x_e + \Gamma_E d_a \leq B. \right)$$

Lemma 3. Given $\Gamma_E \in \{0, \ldots, m\}$ and $\Gamma_V \in \{0, \ldots, n'\}$, the Robust B&S Q-PCStT can be solved by solving $(m - \Gamma_E + 2)(n' - \Gamma_V + 2)$ nominal problems

$$ROPT_Q\,(\Gamma_E, \Gamma_V) = \min_{\substack{a \in \{\Gamma_E, \ldots, m+1\} \\ b \in \{\Gamma_V, \ldots, n'+1\}}} G_Q^{a,b}, \tag{2.29}$$

where for $a \in \{\Gamma_E, ..., m+1\}$ and $b \in \{\Gamma_V, ..., n'+1\}$

$$G^{a,b} = \Gamma_E d_a + \min_{(\mathbf{x},\mathbf{y})\in\mathscr{T}} \left(\sum_{e\in E} c_e^- x_e + \sum_{e=1}^{a} (d_e - d_a) x_e \,\middle|\, \sum_{v\in V} p_v^- (1-y_v) + \sum_{v=1}^{b} (d_v - d_b)(1-y_v) + \Gamma_V d_b \leq Q. \right)$$

Lemmas 1-3 are particularly useful when polynomial-time algorithms are available for graphs with some special structures (see, e.g., Corollary 1). We want to point out that for the case of general graphs, where the deterministic counterparts are NP-Hard, branch-and-cut algorithms like the ones presented in this work remain a preferable option.

Corollary 1. If the input graph is a tree, a series-parallel graph or a 2-tree, the robust counterpart of the PCStT can be solved in $O(|V|^3)$ time.

Proof. The deterministic PCStT can be solved in $O(|V|)$ time on trees [see Klau et al., 2004]. A series-parallel graph can be completed in linear time into a 2-tree. In [Álvarez-Miranda et al., 2010] it has been shown that the PCStT can be solved in $O(|V|)$ time on 2-trees. We complete the proof by combining these results with the result of Lemma 1. □

2.7. Conclusions and Future Work

In this chapter we studied the PCStT and its budget and quota constrained variants assuming interval uncertainty associated with their input parameters. To include and handle this uncertainty we considered the B&S robust optimization (RO) approach, formulating the robust counterpart of the problems by means of different mixed integer programming formulations. Specific branch-and-cut algorithms were implemented to solve these problems. The algorithms were tested on a set of benchmark instances generated from state-of-the-art instances of the deterministic version of the problem. The obtained computational results suggest that: (1) the RO model allows to produce different robust solutions for different levels of conservatism. These solutions provide a protection in terms of the relatively small increase of the solution cost in presence of an increased uncertainty. This important feature of the model offers to the decision maker more flexibility to choose a solution according to her/his perception of the uncertain state of the decision-making environment. (2) The algorithmic performance strongly depends on the model parameters, Γ_E and Γ_V (and B in the case of the Robust B-PCStT). There is a strong correlation between the size of the optimal solution and the corresponding values for B, Γ_E and Γ_V. (3) Among three possibilities to deal with robustness in a MIP model, the addition of a compact set of constraints right at the beginning of the Branch-and-Bound process, outperforms the remaining two (cutting planes) approaches. This can be explained by the fact that from the

beginning of the optimization process the underlying LP contains complete information regarding the robustness of the solution, which allows CPLEX to exploit its powerful preprocessing, heuristics and MIP algorithms, while this is not possible for the cutting plane approaches.

As possible directions for future work, it would be interesting to develop algorithms for 2-trees (or, graphs with a bounded tree-width, in general) that improve the trivial running times obtained by running $O(|V||E|)$ iterations of the deterministic problem. In addition, a strategy combining the results described in Section 2.6 and the utilization of further polyhedral techniques might improve the results we obtained in terms of algorithmic performance.

2.8. Complementary Results

A more detailed analysis of the price of robustness can be done by observing Figure 2.7, where the value of $ROPT$ is shown for different values of Γ_E and Γ_V for instance `K400.4-0.05-0.05`. As expected, the value of $ROPT$ increases when increasing Γ_E and Γ_V. We can observe that different solution values are obtained, for a given value of Γ_E, for different values of Γ_V not greater than 50, but greater values of Γ_V do not produce different objective function values, i.e., different solutions. This behavior can be also explained by considering the relation between Γ_V and Γ_E and the size of the obtained solutions. In particular, we observed that when $\Gamma_E = 40$ and $\Gamma_V = 50$ the solution is made up of 32 edges and 51 customer nodes are not connected, which means that the cost of all the edges in the solution and the prize of almost all the non-connected nodes are set to their upper bounds. This explains why the solutions do not present further changes when increasing the values of Γ_E and Γ_V, since the worst-case solution has been already achieved.

				$\lvert E_T\rvert$			$\lvert V_{T_{p_v>o}}\rvert$		
Instances	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V_{p_v>0}\rvert$	min	mean	max	min	mean	max
`C-`$\{a,b\}$`-`$\{1-5\}$	500	625	65	0	91	315	1	60	236
`C-`$\{a,b\}$`-`$\{6-10\}$	500	1000	65	0	108	318	1	72	242
`C-`$\{a,b\}$`-`$\{11-15\}$	500	2500	65	0	121	306	1	87	248
`C-`$\{a,b\}$`-`$\{16-20\}$	500	12500	65	10	107	263	5	94	250
`D-`$\{a,b\}$`-`$\{1-5\}$	1000	1250	187	0	179	648	1	115	477
`D-`$\{a,b\}$`-`$\{6-10\}$	1000	2000	187	0	215	625	1	142	489
`D-`$\{a,b\}$`-`$\{11-15\}$	1000	5000	187	0	243	627	1	171	495
`D-`$\{a,b\}$`-`$\{16-20\}$	1000	25000	187	6	211	532	3	139	500
`P100.`$\{0-4\}$	100	300	30	20	33	42	14	23	32
`P200.0`	200	587	48	62	62	64	32	34	35
`P400.`$\{0-4\}$	400	1185	106	115	131	151	64	80	101
`K100.`$\{0-10\}$	100	344	13	0	2	12	1	2	8
`K200.0`	200	691	33	7	7	7	8	8	8
`K400.`$\{0-10\}$	400	1493	60	1	20	57	2	14	33

TABLE 2.9: Sizes of instances and their best-known solutions for ($\alpha = 0.05,\ \beta = 0.05$) (average values across all combinations of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$ are shown)

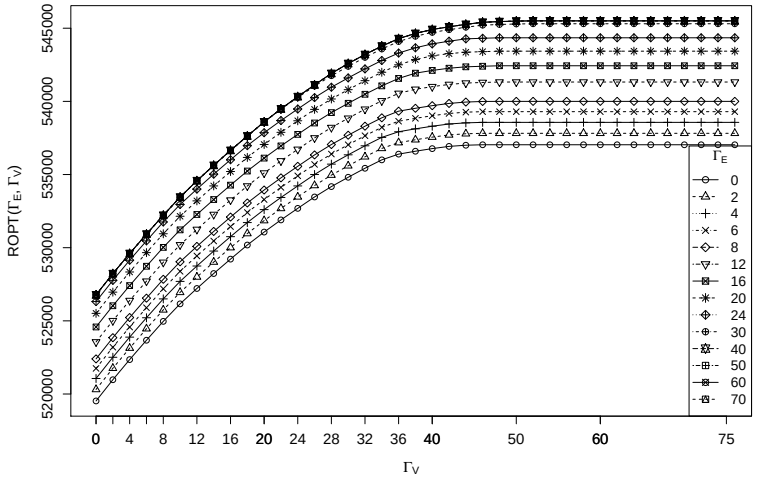


FIGURE 2.7: Values of $ROPT$ for different values of Γ_E and Γ_V, instance K400.4-0.05-0.05

TABLE 2.10: Algorithmic performance for C instances

		Compact				Robustness-Cuts					Compact-Cuts				
Γ_E	Γ_V	Gap(%)	t(s)	#ConCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#RCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#CCuts	#BBNs
0	0	0.000	3.44	552.40	0.73	0.000	2.39	492.33	0.00	0.00	0.000	**2.29**	373.25	0.00	0.00
0	5	0.000	3.32	472.55	0.30	0.000	3.24	537.53	1.18	0.95	0.000	**3.17**	431.78	6.33	1.08
0	20	0.000	3.09	424.95	0.70	0.000	**2.90**	630.13	1.15	0.18	0.000	3.41	480.75	11.05	0.43
0	50	0.000	**2.45**	320.03	0.28	0.000	3.28	524.60	1.08	1.03	0.000	3.31	469.93	15.05	0.95
5	0	0.000	**28.35**	673.73	33.35	0.000	37.95	1281.23	33.05	32.55	0.000	32.11	682.25	36.85	28.53
5	5	**0.000**	**29.54**	552.25	24.43	0.002	46.85	1314.10	38.35	24.40	0.001	36.57	910.95	42.75	32.00
5	20	**0.000**	**31.58**	557.25	29.83	0.001	39.29	1597.98	31.73	16.45	0.001	35.92	789.55	47.13	27.90
5	50	**0.000**	**29.12**	853.68	25.88	0.001	40.51	1170.10	36.18	20.53	0.000	32.39	677.98	52.10	26.23
20	0	**0.000**	**31.31**	816.53	59.10	0.037	71.45	1762.78	139.38	16.43	0.002	44.27	1005.63	64.18	82.68
20	5	0.001	36.92	910.00	40.20	0.037	71.59	1542.53	151.68	14.60	**0.000**	**36.50**	1069.13	70.88	51.48
20	20	0.001	38.40	1132.78	78.55	0.039	73.07	1458.78	204.68	31.08	**0.000**	**37.03**	826.90	74.83	78.53
20	50	0.001	38.76	881.58	46.98	0.033	70.51	2040.58	156.23	13.60	**0.000**	**34.42**	911.25	78.95	44.83
50	0	0.017	**80.95**	2056.60	296.50	0.078	123.12	1765.20	653.13	54.08	**0.014**	93.14	1631.78	107.50	394.43
50	5	0.010	**75.90**	1549.23	395.68	0.076	127.51	1837.75	702.20	79.33	**0.007**	86.94	1374.63	109.40	462.93
50	20	0.008	**84.19**	2092.70	626.65	0.075	131.15	1614.23	757.23	58.10	**0.006**	95.64	1939.50	117.30	716.03
50	50	0.020	**84.51**	2478.50	667.98	0.080	135.58	1870.48	847.75	82.23	**0.005**	89.34	1588.48	121.18	693.05

TABLE 2.11: Algorithmic performance for D instances

		Compact				Robustness-Cuts					Compact-Cuts				
Γ_E	Γ_V	Gap(%)	t(s)	#ConCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#RCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#CCuts	#BBNs
0	0	0.000	39.75	1295.40	0.05	0.000	**18.76**	745.95	0.00	0.03	0.000	23.15	720.10	0.00	0.00
0	5	0.000	32.94	867.03	0.30	0.000	**21.28**	740.85	2.05	0.10	0.000	30.88	973.15	9.50	0.33
0	20	0.000	28.90	789.20	0.68	0.000	**22.92**	818.35	2.08	0.05	0.000	30.49	865.75	16.53	0.38
0	50	0.001	35.73	1206.78	0.48	0.000	**17.96**	733.20	2.50	0.20	0.000	26.04	690.75	23.85	0.23
5	0	**0.003**	**74.28**	1372.08	7.18	0.006	84.22	1129.38	19.40	3.13	0.006	105.62	2092.73	52.93	8.30
5	5	0.008	**79.83**	2043.83	7.15	0.006	91.51	1241.18	24.23	4.48	**0.005**	99.33	1599.48	63.85	8.50
5	20	**0.004**	83.16	1899.65	7.56	0.011	**83.12**	918.90	25.25	4.68	0.007	94.61	1525.78	69.83	10.25
5	50	**0.004**	**79.76**	1269.30	8.35	0.007	88.14	1014.98	28.43	4.73	0.006	95.18	1278.33	78.40	9.48
20	0	**0.012**	**109.56**	1738.18	29.65	0.025	163.98	1372.65	74.90	10.13	0.012	137.22	1737.88	90.00	32.13
20	5	**0.011**	**119.90**	1204.00	38.78	0.020	172.99	1339.70	86.65	11.08	0.011	135.33	1568.70	98.30	40.13
20	20	**0.008**	**120.07**	1049.40	45.40	0.021	179.12	1280.50	117.90	24.38	0.011	139.24	1645.03	106.65	46.48
20	50	**0.009**	**122.25**	1123.38	45.25	0.021	169.16	1259.73	126.13	20.83	0.013	141.34	1573.78	113.50	48.70
50	0	0.026	**173.48**	1753.03	60.75	0.067	200.48	1481.73	118.78	6.25	**0.024**	184.34	1530.88	136.10	51.20
50	5	**0.032**	**173.50**	1589.38	65.68	0.101	202.67	1392.23	127.60	8.55	0.043	186.03	1758.83	141.73	49.68
50	20	**0.020**	**176.41**	1448.95	69.05	0.072	211.01	1476.03	176.70	14.30	0.025	182.73	1415.30	154.33	60.03
50	50	**0.018**	**192.68**	1462.35	170.40	0.070	235.17	1488.95	275.00	19.30	0.032	210.00	1754.75	160.68	203.75

Table 2.12: Algorithmic performance for K instances

		Compact				Robustness-Cuts					Compact-Cuts				
Γ_E	Γ_V	Gap(%)	t(s)	#ConCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#RCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#CCuts	#BBNs
0	0	0.000	5.45	1105.96	0.00	0.000	**4.14**	1082.74	0.00	0.13	0.000	4.20	1091.61	0.00	0.09
0	5	0.000	6.53	1177.78	0.09	0.000	**5.29**	1172.87	7.70	0.30	0.000	6.37	1146.39	28.74	0.30
0	20	0.000	7.93	1358.70	0.13	0.000	**6.96**	1363.26	10.35	0.65	0.000	7.33	1296.09	30.83	0.57
0	50	0.000	7.70	1377.30	0.17	0.000	7.63	1420.09	8.83	0.30	0.000	**7.15**	1425.13	29.13	0.26
5	0	0.000	9.17	1131.73	0.78	0.000	**7.12**	1129.35	11.96	1.91	0.000	7.85	1146.04	11.65	2.35
5	5	0.000	**8.02**	1164.78	0.61	0.000	9.22	1179.52	21.30	2.35	0.000	9.03	1252.70	41.09	2.04
5	20	0.000	12.17	1315.57	0.87	0.000	10.56	1386.04	25.48	2.26	0.000	**10.11**	1307.00	46.61	2.61
5	50	0.000	12.29	1460.52	0.96	0.000	**10.92**	1482.91	23.96	2.74	0.000	11.61	1477.26	53.17	2.96
20	0	0.000	**13.53**	1290.22	6.91	0.000	17.10	1171.78	121.44	19.74	0.000	14.82	1301.87	22.30	19.83
20	5	0.000	**12.79**	1291.48	5.09	0.000	18.43	1250.65	127.35	20.70	0.000	16.57	1392.78	52.13	20.44
20	20	0.000	**15.19**	1363.57	7.043	0.000	21.49	1356.09	139.26	22.00	0.000	16.85	1537.26	57.30	22.65
20	50	0.000	**17.36**	1533.39	8.83	0.000	22.89	1433.22	147.09	25.26	0.000	18.95	1633.39	60.57	25.74
50	0	0.000	**10.07**	1178.13	1.57	0.000	17.07	1194.65	66.35	2.78	0.000	10.94	1304.22	26.17	2.35
50	5	0.000	**11.52**	1258.96	1.35	0.000	17.24	1306.78	66.22	2.00	0.000	13.25	1347.78	56.22	3.83
50	20	**0.000**	**17.44**	1337.87	16.00	0.006	49.93	1416.45	304.39	13.13	**0.000**	18.50	1518.09	67.87	28.09
50	50	**0.000**	**17.34**	1454.65	21.57	0.003	49.86	1491.78	384.39	18.83	**0.000**	22.66	1676.74	66.74	53.44

Table 2.13: Algorithmic performance for P instances

		Compact				Robustness-Cuts					Compact-Cuts				
Γ_E	Γ_V	Gap(%)	t(s)	#ConCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#RCuts	#BBNs	Gap(%)	t(s)	#ConCuts	#CCuts	#BBNs
0	0	0.000	0.63	198.09	0.00	0.000	**0.60**	156.82	0.00	0.00	0.000	0.61	181.27	0.00	0.00
0	5	0.000	**0.60**	159.09	0.00	0.000	0.62	132.09	2.64	0.46	0.000	0.66	154.55	21.91	0.55
0	20	0.000	**0.57**	146.27	0.00	0.000	0.66	161.55	3.09	0.64	0.000	0.65	162.18	16.09	0.46
0	50	0.000	0.55	142.64	0.00	0.000	**0.51**	151.27	2.09	0.00	0.000	0.70	175.36	16.18	0.00
5	0	0.000	**0.74**	140.64	0.18	0.000	0.85	170.00	5.09	1.73	0.000	0.86	164.36	80.18	2.09
5	5	0.000	**0.58**	173.91	0.27	0.000	1.06	174.73	10.18	3.09	0.000	0.99	147.27	102.36	3.55
5	20	0.000	**0.62**	138.00	0.55	0.000	1.13	156.00	9.64	2.82	0.000	0.92	152.36	103.82	2.73
5	50	0.000	**0.65**	171.18	0.27	0.000	0.82	139.55	8.36	1.64	0.000	0.99	160.36	104.09	2.36
20	0	0.000	**0.83**	154.64	2.00	0.000	3.30	178.64	24.27	6.82	0.000	1.51	222.91	82.73	7.18
20	5	0.000	**0.81**	174.00	1.73	0.000	4.23	177.91	33.91	10.09	0.000	1.89	211.82	107.54	10.00
20	20	0.000	**0.79**	116.55	1.82	0.000	3.83	162.82	30.55	9.27	0.000	1.82	206.18	110.00	7.82
20	50	0.000	**0.78**	148.55	1.27	0.000	3.22	159.27	23.82	6.73	0.000	1.45	187.27	108.09	5.91
50	0	0.000	**1.79**	176.27	8.00	0.000	13.70	253.36	85.36	20.00	0.000	3.64	248.64	92.64	24.18
50	5	0.000	**2.11**	194.64	11.55	0.000	17.59	217.45	118.00	27.00	0.000	4.42	240.09	119.09	31.09
50	20	0.000	**1.96**	166.18	11.45	0.000	12.02	210.46	83.72	19.36	0.000	3.92	230.64	117.09	25.82
50	50	0.000	**1.58**	153.55	7.09	0.000	9.82	193.36	67.46	13.27	0.000	2.68	271.18	116.91	14.00

Complementary information about the algorithmic performances for the RPCStT is presented in Tables 2.10 - 2.13 for instances generated using ($\alpha = 0.05,\ \beta = 0.05$). These disaggregated statistics once again confirm that higher values of Γ_E and Γ_V produce a clear increase of the running times. The gain of the problem's complexity and the corresponding increment of the computational effort can be observed from the increasing values of the number of connectivity cuts (columns #ConCuts), robustness cuts (columns #RCuts), compact cuts (columns #CCuts) and Branch-and-Bound nodes (columns #BBNs). From these four extensive tables, in which each line report an average value over the whole group for a particular setting of Γ_V and Γ_E, we conclude that the Compact approach is the most effective one, although the Compact-Cuts strategy behaves quite similarly.

	$\alpha = 0.05,\ \beta = 0.10$						$\alpha = 0.10,\ \beta = 0.05$						$\alpha = 0.10,\ \beta = 0.10$					
	$\lvert E_T \rvert$			$\lvert V_{T_{p_u>o}} \rvert$			$\lvert E_T \rvert$			$\lvert V_{T_{p_u>o}} \rvert$			$\lvert E_T \rvert$			$\lvert V_{T_{p_u>o}} \rvert$		
Instance	min	mean	max	min	mean	max	min	mean	max	min	mean	max	min	mean	max	min	mean	max
C-{1 – 5}	0	92.11	315	1	60.57	236	0	90.63	316	1	59.61	236	0	90.85	314	1	59.72	236
C-{6 – 10}	0	107.7	318	1	71.22	242	0	107.9	318	1	71.42	242	0	108.1	318	1	71.64	242
C-{11 – 15}	0	120.8	304	1	85.97	248	0	121.4	305	1	86.76	248	0	121.4	304	1	86.81	248
C-{16 – 20}	10	106.7	263	5	93.06	250	10	107.7	263	5	93.98	250	10	107.6	263	5	93.94	250
K100.{0 – 10}	0	2.36	12	1	2.53	8	0	1.64	13	1	2.14	8	0	1.99	13	1	2.32	8
K200.0	7	7.00	7	8	8.00	8	7	7.00	7	8	8.00	8	7	7.00	7	8	8.00	8
K400.{0 – 10}	1	23.33	67	2	16.22	43	1	18.59	53	2	13.61	36	1	21.76	64	2	15.29	43

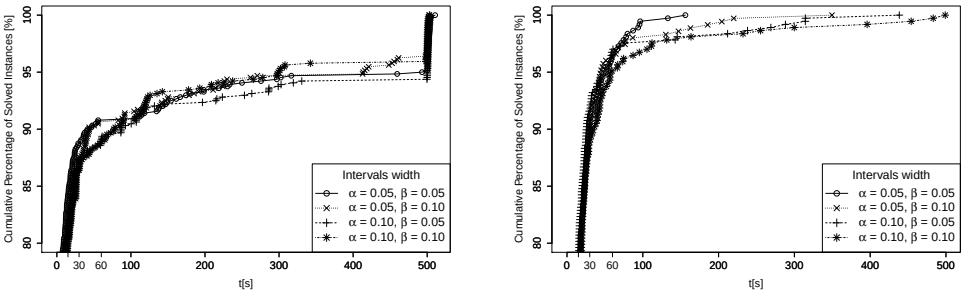
TABLE 2.14: Sizes of the obtained solutions (average values across all combinations of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$ are shown), groups C and K and different values of α, β

		$\Delta ROPT(\%)$											
		$\alpha = 0.05,\ \beta = 0.05$			$\alpha = 0.05,\ \beta = 0.10$			$\alpha = 0.10,\ \beta = 0.05$			$\alpha = 0.10,\ \beta = 0.10$		
Γ_E	Γ_V	min	mean	max	min	mean	max	min	mean	max	min	mean	max
0	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	5	0.67	2.02	3.74	1.35	4.00	7.48	0.67	2.02	3.74	1.35	4.00	7.48
0	20	1.79	3.54	5.00	3.57	6.96	10.00	1.79	3.54	5.00	3.57	6.96	10.00
0	50	1.79	4.07	5.00	3.57	7.71	10.00	1.79	4.07	5.00	3.57	7.71	10.00
5	0	0.00	0.39	1.73	0.00	0.39	1.73	0.00	0.74	3.18	0.00	0.74	3.18
5	5	0.98	2.42	4.41	1.71	4.44	7.79	1.00	2.79	5.20	1.86	4.83	8.83
5	20	2.50	3.99	5.00	4.35	7.48	10.00	2.91	4.37	6.10	4.76	7.93	10.00
5	50	2.67	4.52	5.00	4.38	8.26	10.00	3.09	4.91	6.10	4.79	8.73	10.00
20	0	0.00	0.65	1.98	0.00	0.65	1.98	0.00	1.19	3.40	0.00	1.19	3.40
20	5	0.98	2.72	4.84	1.85	4.74	7.79	1.00	3.28	6.04	1.86	5.37	8.90
20	20	3.05	4.30	5.00	5.45	7.83	10.00	3.07	4.91	7.01	5.99	8.54	10.00
20	50	3.78	4.88	5.00	5.49	8.76	10.00	5.00	5.53	7.01	7.01	9.60	10.00
50	0	0.00	0.73	2.51	0.00	0.73	2.51	0.00	1.27	4.43	0.00	1.27	4.43
50	5	0.98	2.81	4.84	1.85	4.84	7.79	1.00	3.37	6.04	1.86	5.46	8.90
50	20	3.05	4.40	5.00	5.73	7.95	10.00	3.07	5.03	7.04	5.99	8.70	10.00
50	50	4.92	4.99	5.00	6.79	9.06	10.00	5.00	5.73	7.56	9.09	9.93	10.00

TABLE 2.15: Basic statistics of $\Delta ROPT(\%)$ (*Price of Robustness*) for different values of Γ_E and Γ_V, considering different values of α and β, group K

			Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t > 500 s)			
α, β	Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
0.05, 0.05	Compact	368/368	0.047	1.156	10.680	133.000	0.000	0.000	0.000	0.000
	R-Cuts	365/368	0.047	0.766	13.270	476.500	0.020	0.059	0.064	0.114
	C-Cuts	368/368	0.310	0.719	11.190	197.400	0.000	0.000	0.000	0.000
0.05, 0.10	Compact	368/368	0.063	0.969	9.726	147.100	0.000	0.000	0.000	0.000
	R-Cuts	362/368	0.047	0.945	13.250	364.000	0.083	0.163	0.187	0.329
	C-Cuts	368/368	0.047	0.875	9.669	211.200	0.000	0.000	0.000	0.000
0.10, 0.05	Compact	368/368	0.047	0.938	8.175	75.250	0.000	0.000	0.000	0.000
	R-Cuts	359/368	0.047	0.969	11.000	273.700	0.246	0.762	1.605	8.423
	C-Cuts	364/368	0.047	0.734	7.606	102.900	0.000	0.010	0.046	0.154
0.10, 0.10	Compact	365/368	0.047	1.031	9.865	112.100	0.000	0.010	0.130	0.940
	R-Cuts	357/368	0.063	0.969	13.950	282.400	0.154	1.059	3.086	10.580
	C-Cuts	363/368	0.031	0.797	9.672	309.400	0.000	0.010	0.127	0.510

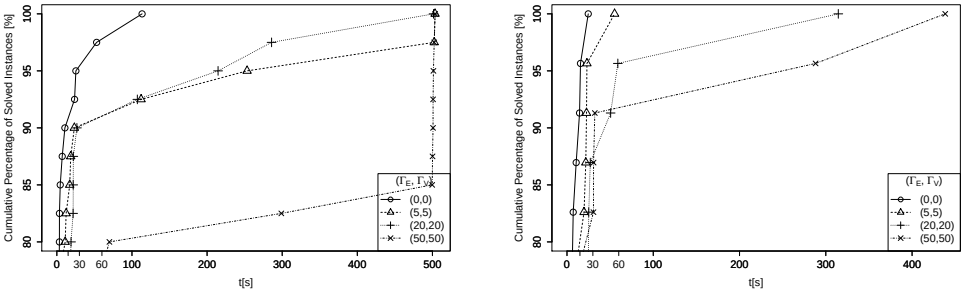
TABLE 2.16: Algorithmic performance statistics for different combinations of α and β, group K



(a) Comparing performance of *Compact* for different α and β, group C.

(b) Comparing performance of *Compact* for different α and β, group K.

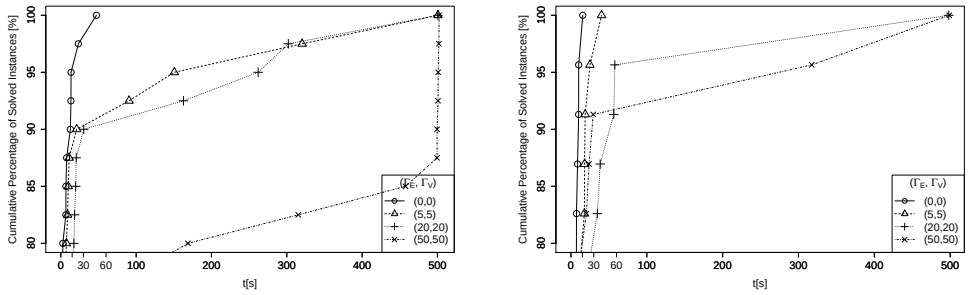
FIGURE 2.8: Cumulative percentage of the total number of solved instances of groups C and K within 500 seconds for different values of α and β (*Compact*)



(a) Comparing performance of *Compact* for different Γ_E and Γ_V, group C.

(b) Comparing performance of *Compact* for different Γ_E and Γ_V, group K.

FIGURE 2.9: Cumulative percentage of the total number of solved instances by *Compact* of groups C and K within 500 seconds for different values of Γ_E and Γ_V ($\alpha = 0.05$, $\beta = 0.10$)

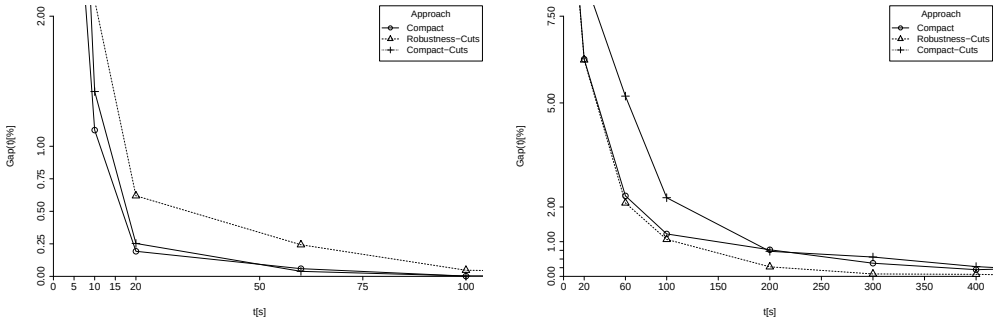


(a) Comparing performance of *C-Cuts* for different Γ_E and Γ_V, group `C`.

(b) Comparing performance of *C-Cuts* for different Γ_E and Γ_V, group `K`.

FIGURE 2.10: Cumulative percentage of the total number of solved instances by *C-Cuts* of groups `C` and `K` within 500 seconds for different values of Γ_E and Γ_V ($\alpha = 0.05$, $\beta = 0.10$)

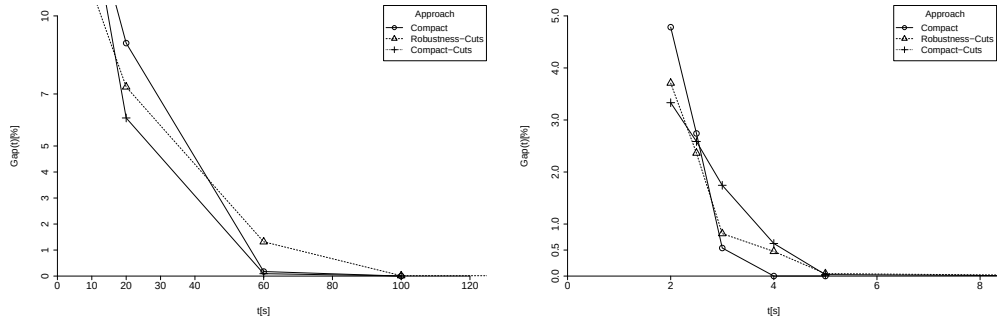
A deeper analysis of the algorithmic performance for the RPCStT can be carried out when studying the evolution, over time, of the gap between the global lower and upper bounds for each of the proposed strategies. For this more specific analysis, we consider only the most difficult instances from each group. From groups `C` and `D` we took the six largest instances and from groups `K` and `P` the five largest ones. In Figures 2.11 and 2.12 we show the evolution of the average gap over time for the subsets of `C` and `D` and the subsets of `K` and `P`, respectively. Figure 2.11(a) shows that *Compact* is the approach that reaches smaller gaps in less time and *R-Cuts* is approach that needs more time to obtain similar gaps. However, Figure 2.11(b) indicates that *R-Cuts* allows to obtain smaller gaps in smaller running times than those of the other approaches. For these two subsets of instances of groups `C` and `D` the time needed to obtain almost 0% of gap is around 100 and 400 seconds, respectively, but a gap of less than 1% (resp. 0.25%) can be achieved (by at least one approach) in less than the 25% (resp. 50%) of this time. Similarly, Figure 2.12(a) suggests that *C-Cuts* is the best approach in the case of the `K` instances; but for `P` instances, the *Compact* is the best approach.



(a) Instances `C-`$\{18-20\}$`-`$\{a,b\}$`-0.05-0.05`.

(b) Instances `D-`$\{18-20\}$`-`$\{a,b\}$`-0.05-0.05`.

FIGURE 2.11: Evolution of Gap (%) considering all combinations of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$



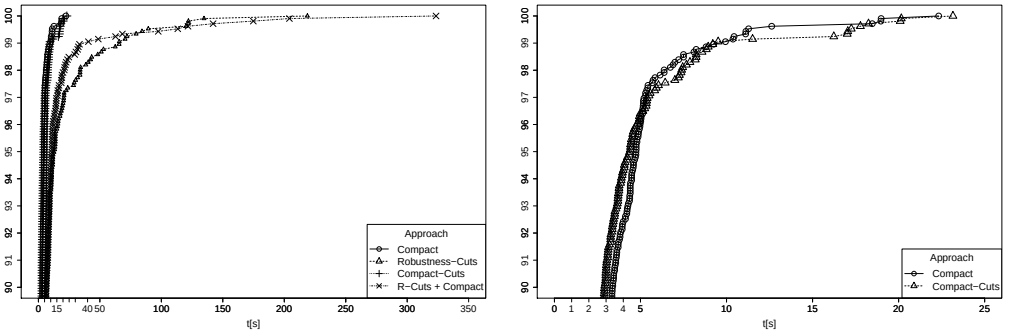
(a) Instances `K400-{6, 10}-0.05-0.05`.

(b) Instances `P400-{1, 5}-0.05-0.05`.

FIGURE 2.12: Evolution of Gap (%) considering all combinations of $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$

		Running time statistics (t $\leq$ 500 s)				Gaps (%) statistics (t $>$ 500 s)			
Approach	#Opt	Min	Median	Mean	Max	Min	Median	Mean	Max
Compact	1056/1056	0.031	1.000	1.474	22.330	0.000	0.000	0.000	0.000
R-Cuts	1056/1056	0.015	0.718	3.665	218.800	0.000	0.000	0.000	0.000
C-Cuts	1056/1056	0.015	0.703	1.272	23.160	0.000	0.000	0.000	0.000
R-Cuts+C	1056/1056	0.015	0.742	3.436	323.500	0.000	0.000	0.000	0.000

TABLE 2.17: Algorithmic performance statistics for group `K` (Robust B-PCStT)



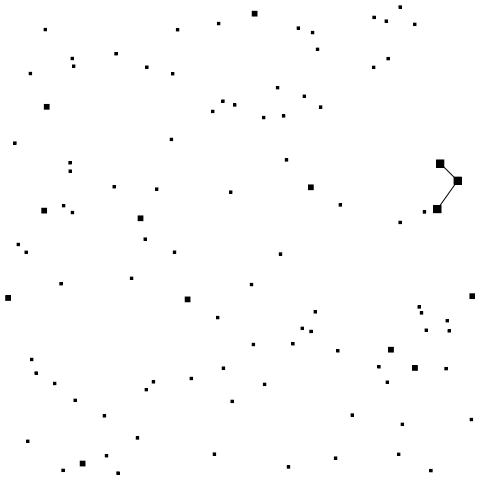
(a) Comparing all four approaches.

(b) Comparing only Compact and Compact-Cuts approaches.

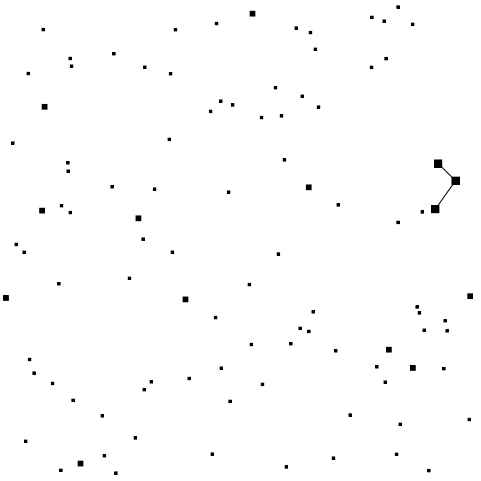
FIGURE 2.13: Cumulative percentage of the average number of solved instances of group `K` within $t = 500$ seconds considering different values of $B \in \{0, 5, 10, \ldots, 95, 100\}B_{\max}\%$, and $\Gamma_E, \Gamma_V \in \{0, 5, 20, 50\}$

For the Robust B-PCStT, interactions among the size of the optimal solution and the corresponding values of B, Γ_E and Γ_V are illustrated in Figure 2.14. Although solutions in Figures 2.14(a) and 2.14(b) are the same, the corresponding $ROPT_B$ values are different because of the difference between the values of Γ_V and their relation with the number of customer nodes that are not connected. Solutions shown in Figures 2.14(c) and 2.14(d) illustrate how large budgets allow to connect a large number of customers while Γ_E exhibits a greater influence on the solution structure. When taking $\Gamma_V = 50$ and $\Gamma_E = 0$ we see that all customer nodes are connected and that is why $ROPT_B = 0$,

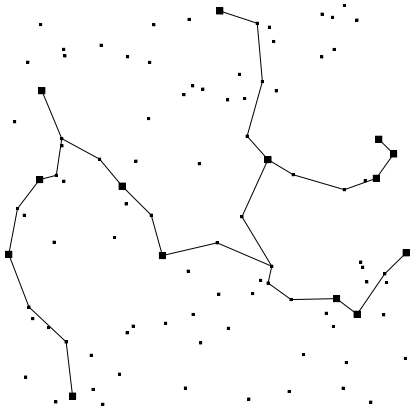
but when taking $\Gamma_E = 50$ two customers are disconnected with the corresponding increase in the value of $ROPT_B$.



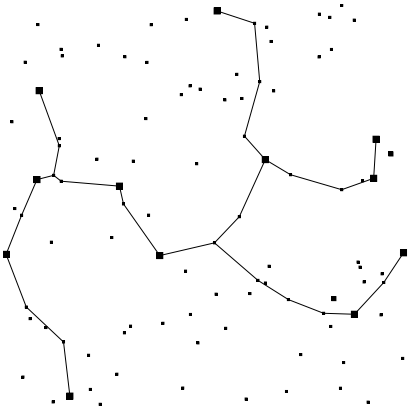
(a) $B = 5\%$, $\Gamma_E = 0$, $\Gamma_V = 50$, $ROPT_B = 138910$.



(b) $B = 5\%$, $\Gamma_E = 50$, $\Gamma_V = 0$, $ROPT_B = 132295$.



(c) $B = 100\%$, $\Gamma_E = 0$, $\Gamma_V = 50$, $ROPT_B = 0$.



(d) $B = 100\%$, $\Gamma_E = 50$, $\Gamma_V = 0$, $ROPT_B = 3531$.

FIGURE 2.14: Optimal solutions for the Robust B-PCStT for instance `K100.10-0.05-0.05` considering different values of B, Γ_E and Γ_V

Chapter 3

The Recoverable Robust Two-Level Network Design Problem

3.1. Introduction

In many real-world settings, when planning an expansion of a telecommunication or power distribution network, a network has to be built even before the set of customers is known with complete certainty. In addition, if different services are offered to customers, uncertainty could be present regarding the type of service that each of the customers needs. Usually, complete information regarding the underlying demand patterns becomes available much later in the planning process. In that case, applying standard deterministic optimization by considering only one of the possible realizations of the input data leads towards solutions that might not be optimal, or for that matter even feasible, for the final data realization. A wait-and-see approach might also be unacceptable from the economical perspective, since the infrastructure cost might significantly increase as time progresses.

Two-stage stochastic optimization and robust optimization (RO) are two possible approaches to deal with these kind of problems. In two-stage stochastic programming [Birge and Louveaux, 2011], the solution is built in two stages. In the first phase, a partial network is built which is later on completed, upon the realization of the uncertain data. The objective is to minimize the cost of the first-stage decisions plus the *expected* cost of the recourse (second-stage) decisions. However, this approach relies on the accuracy of the random representation of the parameter values (such as probability distributions) that allow one to estimate the second-stage expected cost. When such accuracy is not available, the use of *deterministic uncertainty models* arises as a suitable alternative [Kouvelis and Yu, 1997, Bertsimas and Sim, 2003, Ben-Tal et al.,

2010]. In these models no assumptions are made about the distribution of the uncertain input parameters. Consequently, in these RO approaches, single-stage decisions are made and solutions are sought that are immune in a certain sense to all possible realizations of the parameter values. Clearly, such solutions may be over conservative, since the networks constructed minimize the investment costs for the worst possible data realization.

Two-Stage Robust Optimization (2SRO) is a modeling approach that combines classical two-stage optimization with robust optimization. In this case, probability distributions are unknown, so the cost of the second-stage decisions is calculated by looking at the worst-case realization of data. The goal is to find a first-stage solution that minimizes the first-stage costs plus the worst-case second-stage costs across all possible data outcomes. For references on different models of 2SRO we refer the reader to [Ben-Tal et al., 2004, Atamtürk and Zhang, 2007, Thiele et al., 2009] and [Zhao and Zeng, 2012].

Recoverable Robustness is an approach that falls within the framework of 2SRO [see Liebchen et al., 2009]. Recalling our practical context, assume that the network is built in two stages and we are required to find a first-stage solution that should be *robust* against many possible realizations (*scenarios*) of the input data in a second-stage. *Robustness* in this context means that the first-stage solutions are expected to provide a reasonable performance in terms of optimality and/or feasibility, for any possible realization of the uncertain data. For this model, it is instructive to think there is a possibility to *recover* the solution constructed in the first stage in a second stage (i.e., to modify the previously defined network in order to make it feasible and/or cheaper) once the uncertainty is resolved. The set of allowed *recovery actions* and their cost may be known in advance for each of the possible data/scenario realizations. These *recovery actions* are *limited*, in the sense that the effort needed to recover a solution may be algorithmically (in terms of how a solution may be modified) and economically (in terms of the cost of recovery actions) limited. Therefore, instead of looking for a solution that is robust against all possible scenarios without allowing any kind of recovery [which is the case for many RO approaches, see Ben-Tal et al., 2010] we want a solution *robust enough* so that it can be "recovered" promptly and at low cost once the uncertainty is resolved. This balance between robustness and recoverability is what defines a *recoverable robust optimization* problem.

The Two-Level Network Design (TLND) problem [Balakrishnan et al., 1994a,b] models the design of telecommunication and power distribution networks, in which two types of customers (requiring two different levels of service) are taken into account. *Primary* customers require a higher level of service and are required to be connected using a higher level (*primary*) technology; *secondary* customers can be connected either by the primary or a *secondary*, and cheaper, technology. The difference between the cost of the primary and secondary technology is often called the *upgrade* cost.

In the deterministic version of the TLND problem the set of primary customers and its complement, the set of secondary customers, are known in advance. However, when long term decisions need to be made (i.e., when the topology of the network needs to determined), there is not always complete knowledge about the set of primary customers. The topology of the network needs to be determined well before a precise knowledge of the demand is available because of the long lead times involved in constructing physical links (edges) in telecommunications and power distribution networks (for example in telecommunications networks fiber cables need to be installed underground which can take a significant period of time). Further, even if some rough idea of demand is known, changes in demographic, socio-economic, or political factors can lead to changes in the structure of the demand during the planning horizon. Under these conditions, a decision maker needs to find a first-stage solution (a spanning tree comprised by secondary and primary technology edges) that can be *recovered* in the second stage, and turned into a feasible one, once the actual set of primary nodes becomes known. For this case the recovery action is the *late upgrade* of a given edge from secondary to primary technology. (In telecommunications networks once a fiber link is in place it can be relatively quick to upgrade the capacity/technology on a link by changing the equipment at the end points of the link.) For each possible scenario, this upgrade incurs an extra cost, *recovery cost*, defined as the sum of all late upgrades that are needed to ensure that all primary nodes are connected by the primary technology. The Recoverable Robust TLND (RRTLND) problem searches for a solution that minimizes the sum of the first-stage cost and the recovery cost of the second stage defined as the worst case recovery cost over all possible scenarios.

3.1.1 Our Contribution and Outline of the Paper

The RRTLND problem is a new problem not studied previously in the literature. We first study the problem on trees: we show that the RRTLND problem is NP-Hard even on a star (a star is a tree where all nodes except the central one have degree 1) with uniform upgrade and recovery costs; we then propose a preprocessing procedure and a Mixed Integer Programming (MIP) model with a linear number of variables for solving the RRTLND problem on trees to optimality. In the second part of the paper we propose a MIP formulation for the problem on general networks and develop a branch-and-cut algorithm to solve it. We develop problem-dependent techniques for efficiently separating the underlying inequalities within the branch-and-cut framework. In addition, we use a primal heuristic that relies upon the ideas of *matheuristics* and uses an embedded MIP for solving the problem on trees. Finally, an extensive set of computational experiments are carried out in order to assess (1) the performance of the proposed algorithm and its dependence on the problem parameters, and (2) the nature and characteristics of the solutions obtained. The analysis includes a qualitative study of the solutions in terms of Robustness and Recoverability and an assessment of the

algorithmic performance. To complement this analysis, we also consider a Steiner-tree variant of the TLND problem and adapt the algorithm to solve its recoverable robust counterpart.

In Section 3.1.2, the TLND problem is formally defined and a review of the main literature presented. In Section 3.2 the concept of Recoverable Robustness is discussed further, and the RRTLND problem is formally defined. Results regarding the computational complexity of the RRTLND problem on trees are discussed therein and a new MIP model is shown. A MIP formulation for the RRTLND problem on general graphs together with the elements of our branch-and-cut approach is described in Section 3.3. In Section 3.4 the Steiner tree variant of the TLND problem, the Two-Level Steiner Tree (TLStT) Problem, is defined and a MIP formulation is presented for its Recoverable Robust counterpart (RRTLStT). In Section 3.5 we present and analyze the computational results obtained for two sets of instances for the RRTLND problem and for the RRTLStT problem. Concluding remarks are provided in Section 3.6.

3.1.2 The Two-Level Network Design Problem

In this section we provide a formal definition of the TLND problem and give a review of the previous literature on this problem.

The Two-Level Network Design Problem We are given an undirected connected graph $G = (V, E)$, $|V| = n$, $|E| = m$, with a set $P \subseteq V$, which corresponds to the set of *primary nodes*. On each edge $e \in E$ one of two given technologies (*primary* or *secondary*) can be installed. Correspondingly, *primary* and *secondary* edge costs, a_e and b_e are associated with each edge $e \in E$, $a_e \geq b_e \geq 0$, where $u_e = a_e - b_e$; u_e is referred to as the *upgrade* cost as it can be viewed as the cost of *upgrading* a secondary edge to a primary one. Let $\mathbf{X} \in \{0,1\}^{|E|}$ be a binary vector such that $X_e = 1$ if edge $e \in E$ is used in the spanning tree and $X_e = 0$ otherwise; and let $\mathbf{Y} \in \{0,1\}^{|E|}$ be a binary vector such that $Y_e = 1$ if on edge $e \in E$ primary technology is installed and $Y_e = 0$ otherwise. Consequently, if $X_e = 1$ and $Y_e = 0$, secondary technology is installed in e. Let $E(\mathbf{X})$ and $E(\mathbf{Y})$ represent the subsets of edges associated with $\mathbf{X}$ and $\mathbf{Y}$, respectively. The TLND problem consists of finding $(\mathbf{X}^*, \mathbf{Y}^*)$ such that

$$f(\mathbf{X}^*, \mathbf{Y}^*) = \min_{(\mathbf{X},\mathbf{Y})\in\mathcal{D}} \left(\sum_{e\in E(\mathbf{X})} b_e + \sum_{e\in E(\mathbf{Y})} u_e \right) \tag{3.1}$$

where $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y}) \in \{0,1\}^{|E|} \times \{0,1\}^{|E|}\}$ such that $E(\mathbf{X})$ is a spanning tree in G, $E(\mathbf{Y})$ is a Steiner Tree connecting P, and $\mathbf{Y} \leq \mathbf{X}$.

Literature Review The history of the TLND problem begins with the introduction of the Hierarchical Network Design problem (HND) [see Current et al., 1986], which is a special case of the TLND problem with $|P| = 2$. In [Duin and Volgenant, 1989]

the author presents structural properties and reduction tests for the HND; in [Pirkul et al., 1991] a Lagrangian-relaxation based heuristic is developed; in [Sancho, 1995] the author proposes a dynamic programming procedure; and recently, branch-and-cut algorithm is presented in [Obreque et al., 2010].

The TLND problem was introduced in [Duin and Volgenant, 1991] where two heuristics and preprocessing procedures are proposed. Several network flow based models for the TLND problem have been proposed and compared in [Balakrishnan et al., 1994b]. The authors also propose a composite heuristic that provides an approximation ratio of $\frac{4}{4-\rho}$ if the embedded Steiner tree is solved with an approximation ratio of $\rho < 2$. In [Balakrishnan et al., 1994a], the authors provide a dual ascent method derived from a flow-based model presented in [Balakrishnan et al., 1994b]. More recently, [Gouveia and Telhada, 2008] discuss alternative MIP formulations for the problem and solve them using Lagrangian relaxation approaches. Some extensions of the TLND problem combine it with the *facility location* problem [see Current, 1988, Gollowitzer et al., 2013]. In addition to telecommunication applications the TLND problem appears in the design of Internet Protocol networks [Chamberland, 2010] and electrical power distribution systems [Costa et al., 2011].

The Multi-Level Network Design Problem (MLND) corresponds to the more general case in which L types of customers and L technologies are available, and the goal is to find a subtree that enables each node at level ℓ to communicate with other node of the same type, by using a tree built of edges of type at most ℓ, for each $1 \leq \ell \leq L$, $L \geq 2$. The problem has been defined by [Mirchandani, 1996], who called the problem the Multi-Tier Tree Problem and provided a heuristic based on the one proposed for the TLND problem in [Balakrishnan et al., 1994a]. In [Chopra and Tsai, 2002], a branch-and-cut approach derived on a layered graph formulation of the problem has been applied to problems with three to five levels.

3.2. The Recoverable Robust TLND (RRTLND) Problem

In this section we provide references to the recent applications of the recoverable robust optimization, define the RRTLND problem and study the properties of the problem on trees.

Recent Applications of Recoverable Robust Optimization In [Liebchen et al., 2009] the authors introduce the Recoverable Robust Optimization (RRO) concept and provide a general framework for optimization problems affected by uncertainty, while focusing on the applications arising in the railway scheduling. Recently, the concept of RRO has also been applied to other application areas as well. The recoverable robust knapsack problem considering different models of uncertainty is studied in [Büsing

et al., 2011]. Formulations and algorithms for different variants of the recoverable robust shortest path problem are given in [Büsing, 2012]. Finally, in [Cicerone et al., 2012] a more general framework of the RRO is studied in which multiple recovery stages are allowed. The authors apply the new model to timetabling and delay management applications.

As other robust optimization approaches the RRO approach allows different models of the uncertainty set, e.g., interval, polyhedral and discrete. In this work, as in [Büsing et al., 2011], we use the discrete set model of uncertainty.

3.2.1 The Recoverable Robust TLND Problem

Suppose that in a given application of the TLND problem it is not known exactly which elements comprise the set of primary customers P. Instead, we are given a finite set of scenarios K such that, for each $k \in K$, there is a set $P^k \subseteq V$ of nodes corresponding to the primary customers if scenario k is realized. Additionally, motivated by the practical applications, we are given a root node r such that $r \in P^k$ for all $k \in K$. We note that while the application typically has a root node (the root represents, for example, a central office, i.e., a connection to the backbone network), if this is not the case it is easy to modify the formulations and procedures described in this work to address the situation.[1]

The decision maker needs to determine the topology of the spanning tree connecting the nodes in the network in the first stage. He/she may decide to install the primary technology on edge $e \in E$ in the first stage, or to recover the edge in the second (recovery) stage by *upgrading* it from the secondary to the primary technology (in case scenario k is realized and set P^k requires it). Hence, for each edge e and for each scenario k, we also define the *late upgrade* (or *recovery*) cost $r_e^k \geq u_e = a_e - b_e$ that needs to be paid if secondary technology is upgraded on edge e in the second stage when *scenario* k is realized; as opposed to u_e being the *regular* (or *first-stage*) *upgrade* cost.

In our problem, for each scenario $k \in K$, each of the customers $v \in P^k$ is to be served by the primary technology, i.e., there exists a path between r and v along that tree, consisting of solely primary edges. These primary edges can either be installed in the first stage, or recovered in the second stage. Further, in the practical application (for administrative reasons generally) it is required that the primary edges form a connected network (i.e., there can be no isolated primary edges). Our goal is to find a spanning tree (along with a prescription of which edges should be installed as primary in the first

[1] Simply create a fictitious root node that has an edge to every node in the graph, and either (i) make these new edges have zero cost and add the requirement that the degree of the root node is 1, or (ii) give a sufficiently large cost to these new edges so that only one of them will be in the optimal solution.

stage) that minimizes the overall installation cost in the first stage (given as the sum of the costs of primary and secondary edges), plus the worst recovery costs, calculated over all scenarios $k \in K$.

More formally, let $\mathbf{X} \in \{0,1\}^{|E|}$ be a binary vector as defined in §3.1.2. Let $\mathbf{Y^0} \in \{0,1\}^{|E|}$ be a binary vector such that $Y_e^0 = 1$ if on edge e the primary technology is installed in the first-stage and $Y_e^0 = 0$ otherwise. Let $\mathbf{Y^k} \in \{0,1\}^{|E|\times|K|}$ be a binary vector such that $Y_e^k = 1$ if the secondary technology that was installed in the first-stage on edge e is upgraded into the primary one in scenario $k \in K$.

Given a scenario $k \in K$ and a first-stage solution $\left(\mathbf{X}, \mathbf{Y^0}\right)$ ($\mathbf{X}$ associated to a spanning tree of G and $\mathbf{Y^0} \leq \mathbf{X}$), the *recovery cost* is the minimum total upgrade cost needed to provide feasibility to $\left(\mathbf{X}, \mathbf{Y^0}\right)$ by recovery actions $\mathbf{Y^k}$. This cost can be expressed as

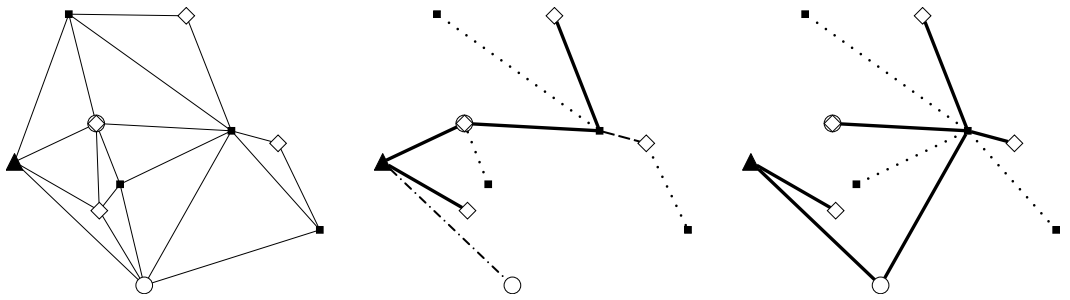
$$\min_{\mathbf{Y^k} \in \mathcal{Y}(\mathbf{X},\mathbf{Y^0},k)} \left\{ \sum_{e \in E(\mathbf{Y^k})} r_e^k \right\},$$

where $\mathcal{Y}\left(\mathbf{X}, \mathbf{Y^0}, k\right)$ is the set of all possible late upgrades for pair $(\mathbf{X},\mathbf{Y^0})$ and the set of primary customers P^k. In other words, vector $\mathbf{Y^k}$ expresses how to *recover* the solution $\left(\mathbf{X}, \mathbf{Y^0}\right)$ in scenario k in order to make it feasible. For each $k \in K$, the set of all feasible recoveries is given as:

$$\mathcal{Y}\left(\mathbf{X}, \mathbf{Y^0}, k\right) = \{\mathbf{Y^k} \in \{0,1\}^{|E|\times|K|} \mid E(\mathbf{Y^0}) \cup E(\mathbf{Y^k}) \text{ is a Steiner tree spanning } P^k, \\ \mathbf{Y^k} \leq \mathbf{X} - \mathbf{Y^0}\}.$$

Note that because of the requirement that the final network (after the uncertainty is resolved) does not allow for isolated primary edges (i.e., $E(\mathbf{Y^0}) \cup E(\mathbf{Y^k})$ is connected), it is easy to see that $E(\mathbf{Y^0})$ must be connected if $r_e^k \geq u_e$. Notice that, given the first stage decision, for each $k \in K$, the optimal recovery solution can be found in $O(n)$ time. The following second-stage objective function, $R(\mathbf{X}, \mathbf{Y^0})$, expresses the *robust recovery cost* (which is the maximum recovery cost over all scenarios $k \in K$):

$$R(\mathbf{X}, \mathbf{Y^0}) = \max_{k \in K} \min_{\mathbf{Y^k} \in \mathcal{Y}(\mathbf{X},\mathbf{Y^0},k)} \left\{ \sum_{e \in E(\mathbf{Y^k})} r_e^k \right\}.$$



(a) Instance of the RRTLND problem.

(b) Recovery costs are $r_e = 1.5\ \forall e \in E$.

(c) Recovery costs are $r_e = 3\ \forall e \in E$.

FIGURE 3.1: Instance and optimal solutions for the RRTLND problem; node with symbol ▲ corresponds to r, nodes denoted by ◇ are primary nodes in scenario $k = 1$ and nodes denoted by ◯ are primary nodes in scenario $k = 2$; for each $e \in E$, its primary and secondary costs are $a_e = 2$ and $b_e = 1$, respectively. Dotted, bold, dashed and dot-dashed edges correspond to $E(\mathbf{X})$, $E(\mathbf{Y^0})$, $E(\mathbf{Y^1})$ and $E(\mathbf{Y^2})$, respectively.

The Recoverable Robust TLND (RRTLND) problem is defined as follows

$$OPT_{RR} = \min\{ \sum_{e \in E(\mathbf{X})} b_e + \sum_{e \in E(\mathbf{Y^0})} u_e + R(\mathbf{X}, \mathbf{Y^0}) \mid (\mathbf{X}, \mathbf{Y^0}) \in \{0,1\}^{|E|} \times \{0,1\}^{|E|}, \tag{3.2}$$

$$E(\mathbf{X}) \text{ is a spanning tree on } G,$$
$$\mathbf{Y^0} \leq \mathbf{X} \text{ and } E(\mathbf{Y^0}) \text{ is connected } \}.$$

In Figure 3.1(a) an instance of the RRTLND problem with two scenarios is shown. In Figures 3.1(b) and 3.1(c) optimal solutions for different cost structures are presented. In the first case, recovery (i.e., late upgrade) costs are 50% more expensive than regular upgrade costs while in the second case the difference goes to 200%. This difference in the cost structure explains why in the solution shown in 3.1(b) there are edges that are recovered in a second stage for each of the scenarios, while in the solution shown in 3.1(c) no recovery is performed since it is cheaper to install primary edges in the first stage than recover edges in a second stage. The cost of the first solution is given by $OPT_{RR} = 1 \times 9 + 1 \times 4 + \max\{1 \times 1.5, 1 \times 1.5\} = 14.5$ and the cost of the second solution is given by $OPT_{RR} = 1 \times 9 + 1 \times 6 + \max\{\emptyset\} = 15$.

A first-stage solution $(\mathbf{X}, \mathbf{Y^0})$ is *robust* because, regardless of which scenario is realized, it ensures that the second-stage actions will be efficient (due to the minimization of the worst case) and easy to implement (because only upgrades are needed). In this sense, the more scenarios we take into account to find $(\mathbf{X}, \mathbf{Y^0})$, the more *robust* the solution is; because we are foreseeing more possible states of the future uncertainty. Along the same line, recoverability is the capability of a first-stage solution to become feasible by means of second-stage actions.

We wish to emphasize that in this two-stage setting the classical single-stage RO approaches such as those proposed in [Kouvelis and Yu, 1997] or [Ben-Tal and Nemirovski, 2000] are not good models, and can be overconservative. In intuitive terms, because the typical RO approaches are single-stage approaches without the possibility of recovery in the second stage, they require the constructed solutions to be feasible for all scenarios!

In RRO the first-stage solution lies between two extremes: an *absolute robust* (AR) solution and a *pure wait-and-see* (W&S) solution. The first case corresponds to a solution for which no recovery is allowed, i.e., $E(\mathbf{Y^0})$ spans $\check{P} = \bigcup_{k \in K} P^k$, so the solution is feasible for all scenarios (this solution in the first case can be viewed as one that would be obtained under the classical single-stage RO approach). On the contrary, the second case corresponds to a solution for which $E(\mathbf{Y^0}) = \emptyset$, so a complete primary Steiner tree should be constructed (but only the most expensive one is considered in the total cost) in the second-stage for each P^k, $k \in K$ (the solution in the second case can be viewed as one with maximum recovery as it requires each primary edge to be obtained via recovery). Both solutions can lead to very high total costs, either because unnecessarily many primary edges have to be installed in the first-stage or because the second-stage primary costs are considerably higher than those of the first stage. The *Gain of Recovery* (GoR) is defined as the relative difference between OPT_{RR} and the cost of these two solutions, i.e., $GoR(AR) = \frac{OPT_{AR} - OPT_{RR}}{OPT_{AR}} \times 100\%$ and $GoR(W\&S) = \frac{OPT_{W\&S} - OPT_{RR}}{OPT_{W\&S}} \times 100\%$, where OPT_{AR} and $OPT_{W\&S}$ are the costs of the optimal AR and W&S solutions respectively.

3.2.2 The RRTLND Problem on Trees

In this section we consider the RRTLND problem on trees.

3.2.2.1 Complexity of the RRTLND Problem on Trees

Theorem 1. Solving the RRTLND problem is NP-hard even if the input graph G is a tree, and all regular and late upgrade costs are uniform.

Proof. Because the input graph is a tree, every edge in the graph will have at least secondary technology installed. Therefore the optimization only needs to consider regular and late upgrade costs.

We will show the result by a transformation [the main idea in this transformation is similar to Garg et al., 1997] from the *minimum vertex cover problem*. Given a graph $H = (V_H, E_H)$, $V_H = \{v_1, \ldots, v_n\}$, a set of vertices such that each edge of the graph is incident to at least one vertex of the set is called a *vertex cover*. In the minimum vertex cover problem we wish to find a vertex cover of smallest cardinality. Given an instance

of a vertex cover on the graph H, construct an instance of the RRTLND problem with K scenarios as follows. First, construct a star graph $S = (V_S, E_S)$ from H as follows. Let $V_S = v_0 \cup V_H$, and $E_S = \bigcup_{i=1\ldots,n}\{v_0, v_i\}$. Let the upgrade costs in the first stage be $u_e = 1$, for all $e \in E_S$, and let $M = n/2$ be the uniform second-stage upgrade cost, i.e., $r_e^k = M$, for all $e \in E_S$, $k \in K$. For each edge $\{u_k, v_k\}$ in E_H, create a scenario $k \in K$ in S, by setting $P^k = \{v_0, u_k, v_k\}$.

We now show that the optimal solution of the RRTLND problem on S provides us the minimum vertex cover on H. Without loss of generality, assume that the value of the vertex cover, C, on H is such that $C \leq \frac{n-1}{2}$.[2] Consider the possible values for the maximum recovery cost R^*: (i) If there exists $k \in K$, such that the edges $\{u_k, v_0\}$ and $\{v_0, v_k\}$ were not purchased in the first stage, then the maximum recovery cost will be $R^* = 2M$. (ii) If for all $k \in K$ at least one of the two edges is purchased in the first stage, but there also exists $\hat{k}$ such that exactly one of the two edges is purchased, then $R^* = M$. Since for each scenario $k \in K$, at least one of the edges $\{u_k, v_0\}, \{v_0, v_k\}$ need to be installed in the first stage, the minimum cost first-stage solution that satisfies this property corresponds to the minimum vertex cover on H (edge $\{v_0, v_k\}$ installed in the first stage (on S) corresponds to node v_k in the vertex cover on H). Therefore, the total cost of such a constructed solution is upper bounded by $C+M$. (iii) Finally, if for all $k \in K$, both edges are purchased in the first stage, $R^* = 0$, but the first-stage cost is equal to n. It is not difficult to see that the second solution will be the optimal one (recall that we chose H such that $C < n/2$) since: $C + M < 2M$ and $C + M < n$. □

3.2.2.2 A MIP Model for the RRTLND Problem on Trees

We now provide a MIP formulation for the RRTLND problem on trees for which it is necessary to perform an $O(nK)$ preprocessing. For $K = const$, this formulation is of compact size. Furthermore, it involves only binary variables associated with the installation of the primary technology in the first stage. Due to the preprocessing, this model does not involve the variables associated with the second-stage decisions.

Preprocessing: Given G that has a tree structure, for each scenario $k \in K$ we first solve the Steiner tree problem with the set P^k being the terminal nodes of that tree. We assume that on all edges in G secondary technology is installed in the first stage, so that all edges of the Steiner Tree need to be recovered in the second stage. Therefore, to find the optimal Steiner tree, we consider the edge cost defined by r_e^k, for each $e \in E$, for each $k \in K$. Let $\mathcal{P}_k$ be the set of edges corresponding to the optimal Steiner tree, for $k \in K$, and let $\omega_k = \sum_{e\in\mathcal{P}_k} r_e^k$ be the recovery cost for that tree,

[2] Given a graph H we can create another graph H' by duplicating the set of nodes and add an additional node (i.e., $V_{H'} = V_H \cup v_0 \cup \{v_{n+1}, \ldots, v_{2n}\}$). The set of edges $E_{H'}$ in this new graph includes the previous edges and an edge from v_0 to each node v_i, $i = 1, \ldots, 2n$. It is easy to see that the minimum vertex cover on $V_{H'}$ is the union of v_0 and the minimum vertex cover on V_H, and satisfies our assumption on the size of the vertex cover.

assuming that there were no primary edges in the first stage. Obviously, finding the optimal Steiner trees can be done in $O(n)$ time, for each $k \in K$. We now state a useful property concerning the structure of RRTLND problem solutions on trees.

Property 1. Let $\mathcal{P} = \bigcap_{k \in K} \mathcal{P}_k \neq \emptyset$ denote the set of edges for which the recovery is needed over all scenarios $k \in K$. Given that for all $e \in E$, $k \in K$ we have $r_e^k \geq u_e$, there always exists an optimal solution to the RRTLND problem on trees such that the primary edges are installed in the first stage along all edges in $\mathcal{P}$. Further, the subgraph induced by $\mathcal{P}$ is connected.

Hence, the optimal primary subtree of the first stage is a rooted subtree of G which is a superset of $\mathcal{P}$ and a subset of E. Therefore, if $\mathcal{P} \neq \emptyset$, we can shrink all the edges of $\mathcal{P}$ into the root node and continue solving the problem on the shrunken tree. Consequently, we can assume w.l.o.g. that $\mathcal{P} = \emptyset$. Given that G is a tree with a pre-specified root node, for each edge $e : \{u, v\} \in E$ $(u, v \neq r)$, we can uniquely determine the *predecessor* edge e' on the path between r and e. Let $\mathbf{s} \in \{0,1\}^{|E|}$ be a binary vector such that $s_e = 1$ if a primary technology is installed in the first stage and $s_e = 0$ otherwise. The following formulation allows us to solve the RRTLND problem on trees:

$$f(\mathbf{s}^*) = \min \sum_{e \in E} u_e s_e + \lambda \tag{T.1}$$

$$\text{s.t.} \quad s_{e'} \geq s_e \qquad \forall e \in E,\ e' \text{ is predecessor of } e : \{u, v\}, u, v \neq r \tag{T.2}$$

$$\lambda \geq \omega_k - \sum_{e \in \mathcal{P}_k} r_e^k s_e \qquad \forall k \in K \tag{T.3}$$

$$\mathbf{s} \in \{0,1\}^{|E|},\ \lambda \geq 0 \tag{T.4}$$

In the formulation (T.1)-(T.4) we only have one set of binary variables, $\mathbf{s}$, and $O(n + K)$ constraints. Therefore, for a constant number of scenarios, this is a compact formulation. Constraints (T.2) force first-stage primary edges to form a connected component rooted at r. Inequalities (T.3) model the nested maximization problem associated with the robust recovery cost; if primary technology is installed on edge e in the first stage, then its recovery cost is subtracted from ω_k for those sets for which e is supposed to be upgraded in the second stage (i.e., for $e \in \mathcal{P}_k$). This MIP model will be used in a matheuristic fashion for finding feasible solutions of the RRTLND problem in general graphs. This will be the core of the primal-heuristic embedded into a branch-and-cut approach framework that we discuss in §3.3.2.

3.3. MIP Model and Branch-and-Cut Algorithm

Before we provide a MIP model for the RRTLND problem, we observe that for every feasible solution of the problem, we can associate a rooted spanning arborescence consisting of a rooted primary sub-arborescence embedded into the secondary one. In

addition, for each $k \in K$, edges from $E(\mathbf{Y^k})$ can uniquely be oriented, so that the set of directed primary edges from the first-stage solution, plus the set of directed edges from $E(\mathbf{Y^k})$ builds a Steiner arborescence spanning P^k. Henceforth, instead of dealing with MIP models containing binary variables associated with edges of the graph G, we will consider its bidirected counterpart, $G_A = (V, A)$, where $A = \{(r,i) \mid e : \{r,i\} \in E\} \cup \{(i,j),(j,i) \mid e : \{i,j\} \in E,\ i,j \neq r\}$.

3.3.1 MIP formulation for the RRTLND Problem

The MIP formulation investigated in this work is based on directed cut-set inequalities. The LP-relaxation of this model usually accomplishes good quality lower bounds, since many facet-defining inequalities can be projected out of the directed model for tree problems [see Grötschel et al., 1992].

Let $\mathbf{x} \in \{0,1\}^{|A|}$ be a binary vector such that $x_{ij} = 1$ if arc $(i,j) \in A$ belongs to the spanning arborescence and $x_{ij} = 0$ otherwise, let $\mathbf{y^0} \in \{0,1\}^{|A|}$ be a binary vector such that $y^0_{ij} = 1$ if primary technology is installed in arc $(i,j) \in A$ in the first stage and $y^0_{ij} = 0$ otherwise. Let $\mathbf{y^k} \in \{0,1\}^{|A|\times|K|}$ be a binary vector such that $y^k_{ij} = 1$ if the secondary technology installed on arc $(i,j) \in A$ is upgraded into the primary one in scenario $k \in K$ and $y^k_{ij} = 0$ otherwise. We use the following notation: A set of vertices $S \subseteq V$ $(S \neq \emptyset)$ and its complement $\overline{S} = V \backslash S$, induce two directed cuts: $\delta^+(S) = \left\{(i,j) \mid i \in S, j \in \overline{S}\right\}$ and $\delta^-(S) = \left\{(i,j) \mid i \in \overline{S}, j \in S\right\}$; we write $\mathbf{x}(A') = \sum_{(i,j)\in A'} x_{ij}$ for any subset $A' \subset A$.

Vector $\mathbf{x}$ is associated with a directed spanning tree of G_A (spanning arborescence) rooted at r if it satisfies the following set of inequalities

$$\mathbf{x}\left(\delta^-(S)\right) \geq 1 \quad \forall S \subseteq V \backslash \{r\},\ S \neq \emptyset, \tag{3.3}$$

a vector $\mathbf{y^0}$ is associated with a directed arborescence of G_A rooted at r if it satisfies

$$\mathbf{y^0}\left(\delta^-(S)\right) \geq \mathbf{y^0}\left(\delta^-(i)\right) \quad \forall i \in S,\ \forall S \subseteq V \backslash \{r\},\ S \neq \emptyset, \tag{3.4}$$

and a vector of recovery actions $\mathbf{y^k}$ along with a vector $\mathbf{y^0}$ are associated with a directed Steiner arboresence of P^k for all scenarios $k \in K$ if they fulfil

$$\left(\mathbf{y^0} + \mathbf{y^k}\right)\left(\delta^-(S)\right) \geq 1 \quad \forall S \subseteq V \backslash \{r\},\ S \cap P^k \neq \emptyset,\ \forall k \in K. \tag{3.5}$$

Constraints (3.3), (3.4) and (3.5) are called $\mathbf{x}$-cuts, $\mathbf{y^0}$-cuts and *scenario*-cuts, respectively. As we will describe in detail later, our branch-and-cut performs at a given node of the branch-and-bound tree a separation procedure of $\mathbf{x}$-cuts, $\mathbf{y^0}$-cuts and *scenario*-cuts by means of (i) the resolution of a max-flow problem on a *support graph* induced

by the Linear Programming (LP) relaxation and (ii) a combinatorial enumeration of those cuts on a *support tree* also induced by the current LP relaxation.

The MIP model for the RRTLND problem reads then as follows:

$$\min \sum_{e \in E} b_e X_e + \sum_{e \in E} u_e Y_e^0 + \omega$$

$$\text{s.t.} \quad \omega \geq \sum_{e \in E} r_e^k Y_e^k \qquad \forall k \in K \tag{3.6}$$

$$(3.3),\ (3.4),\ (3.5)$$

$$X_e = x_{ij} + x_{ji} \quad Y_e^0 = y_{ij}^0 + y_{ji}^0 \quad Y_e^k = y_{ij}^k + y_{ji}^k \qquad \forall e : \{i,j\} \in E, \forall k \in K \tag{3.7}$$

$$X_e, Y_e^0, Y_e^k \in \{0,1\} \qquad \forall e \in E, k \in K \tag{3.8}$$

Before concluding this section we note that it is possible to also write a compact formulation using three sets of flow variables—to model the three sets of connectivities imposed by constraints (3.3), (3.4) and (3.5). However, given the number of scenarios, this model blows up rapidly and is not a computationally viable approach for the problem.

3.3.2 Branch-and-Cut Algorithm

The MIP formulation based on cut-set inequalities for the RRTLND problem cannot be solved directly, even for small instances, since there are an exponential number of **x**-, $\mathbf{y^0}$- and *scenario*-cuts. In this section we describe a branch-and-cut approach used for solving the problem. We first explain different schemes designed to separate the directed cut-set constraints (i.e., (3.3), (3.4) and (3.5)). Next, the initialization performed to improve the quality of the lower bounds of the initial MIP model is described. Finally, we provide a description of the primal heuristic embedded within the branch-and-cut framework that helps in establishing high-quality upper bounds early in the search process.

3.3.3 Separation of Cut-set Inequalities

Cut-set inequalities are usually separated using maximum-flow algorithms. Basic ideas of this separation for the RRTLND problem are provided below. In addition, we also explain two advanced separation mechanisms that are called *mixed* and *combinatorial cuts separation*. The latter approach uses the problem-specific structure to speed-up the separation process and improves lower bounds in the earlier phase of the search process.

Basic Separation Procedures (Max-Flow Based Cuts) Violated cut-set inequalities can be found in polynomial time using a maximum-flow algorithm on the

support graph with arc-capacities given by the current fractional solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}^{\mathbf{0}}, \tilde{\mathbf{y}}^{\mathbf{k}})$. When separating $\mathbf{x}$-, $\mathbf{y}^{\mathbf{0}}$- and *scenario*-cuts, the capacities of the support graph are set to be equal to the values of $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}^{\mathbf{0}}$ and $(\tilde{\mathbf{y}}^{\mathbf{0}} + \tilde{\mathbf{y}}^{\mathbf{k}})$, respectively. For finding the maximum flow in a directed graph, we used an adaptation of [Cherkassky and Goldberg, 1995] maximum flow algorithm.

The separation is performed in the following order: First, we randomly select a node from $V \setminus \{r\}$ and if there is a violated $\mathbf{x}$-cut separating v from r, we insert it into the LP (together with the corresponding set of nested and backward cuts, see the explanation below). We resolve the LP and continue as long as such violated cuts are found. After that, we attempt to find violated $\mathbf{y}^{\mathbf{0}}$-cuts. This time, we perform the maximum-flow calculation between r and each $i \in V \setminus \{r\}$, such that $\mathbf{y}^{\mathbf{0}}(\delta^-(i)) > 0$. In the final phase, when no more violated $\mathbf{x}$-cuts and $\mathbf{y}^{\mathbf{0}}$-cuts can be found, we search for violated *scenario*-cuts. For each scenario $k \in K$, we perform the maximum-flow calculation between r and each $i \in P^k$.

By following this order in the separation procedure, we avoid inserting cuts that have a greater likelihood of being weak (i.e., dominated by others) and thus reduce the computational effort of the separation. For example, for a given set S and $i \in S$ such that $y^0(\delta^-(i)) = 1$, the corresponding y^0-cut dominates all scenario-cuts associated with the same S.

Mixed Separation Because $\mathbf{y}^{\mathbf{0}} \leq \mathbf{x}$, if a set $S \subseteq V \setminus \{r\}$ induces a violated $\mathbf{x}$-cut then it might also induce a violated $\mathbf{y}^{\mathbf{0}}$-cut, if there exist $i \in S$ such that $\mathbf{y}^{\mathbf{0}}(\delta^-(S)) < \mathbf{y}^{\mathbf{0}}(\delta^-(i))$. Because $\mathbf{y}^{\mathbf{k}} \leq \mathbf{x} - \mathbf{y}^{\mathbf{0}}$, if there exists a scenario $k \in K$, such that $S \cap P^k \neq \emptyset$, the same set S also induces a violated *scenario*-cut. Hence, within the separation process applied to $\mathbf{x}$-cuts we can also separate $\mathbf{y}^{\mathbf{0}}$-cuts and *scenario*-cuts without solving another max-flow problem. We use these facts to develop a separation procedure that we refer to as *mixed separation*. The outline of this procedure is given in Algorithm 1. In this procedure, we call the maximum-flow algorithm `MaxFlow` $(G_A, \tilde{\mathbf{x}}', r, v, S_r, S_v)$ that, for a given directed graph G_A, calculates the maximum flow between r and v with capacities $\tilde{\mathbf{x}}'$. The algorithm returns two subsets of nodes: S_r, $r \in S_r$ and S_v, $v \in S_v$, such that the edges of the cut $\delta^+(S_r)$ and $\delta^-(S_v)$ induce the maximum flow. Inequalities associated with the set S_r and S_v are called *forward* and *backward cuts*, respectively. Then, we continue recalculating maximum flows on the same graph G_A, on which the capacities of the edges from the two previously found cut-sets $\delta^+(S_r)$ and $\delta^-(S_v)$ are set to one. That way, we detect disjoint cuts and we reuse the previous maximum flow computation to speed up the overall separation. The latter strategy is known as the *nested cuts* approach [see Ljubić et al., 2006]. Variable *MAX-CUTS* denotes the number of cuts to be inserted before the LP is resolved. In our implementation *MAX-CUTS* was set to 25.

Finally, we apply two variants of the mixed cut separation. The first one is described in Algorithm 1: whenever we detect a violated $\mathbf{x}$-cut, we also add corresponding violated

Algorithm 1 Mixed Separation

Input: Graph $G_A = (V, A)$, fractional solution $\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}^{\mathbf{0}}, \tilde{\mathbf{y}}^{\mathbf{k}}\right)$
Choose a random node v from $V \setminus \{r\}$
$\tilde{\mathbf{x}}' = \tilde{\mathbf{x}}$
repeat
  $f = \texttt{MaxFlow}(G_A, \tilde{\mathbf{x}}', r, v, S_r, S_v)$
  Detect the cut $\delta^+(S_r)$ such that $\tilde{\mathbf{x}}'\left(\delta^+(S_r)\right) = f$, $r \in S_r$
  if $f < 1$ **then**
    Insert violated cut $\mathbf{x}\left(\delta^+(S_r)\right) \geq 1$ into the LP
    $\tilde{i_r} = \arg\max_{i \notin S_r} \tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^-(i)\right)$
    if $\tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^+(S_r)\right) < \tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^-\left(\tilde{i_r}\right)\right)$ **then**
      Insert violated cut $\mathbf{y}^{\mathbf{0}}\left(\delta^+(S_r)\right) \geq \mathbf{y}^{\mathbf{0}}\left(\delta^-\left(\tilde{i_r}\right)\right)$ into the LP
    for all $k \in K$, $\overline{S_r} \cap P^k \neq \emptyset$ **do**
      Insert the violated cut $\left(\mathbf{y}^{\mathbf{0}} + \mathbf{y}^{\mathbf{k}}\right)\left(\delta^+(S_r)\right) \geq 1$ into the LP
    $\tilde{x}'_{ij} = 1$, $\forall (i,j) \in \delta^+(S_r)$
    Detect the cut $\delta^-(S_v)$ such that $\tilde{\mathbf{x}}'\left(\delta^-(S_v)\right) = f$, $v \in S_v$
    if $S_v \neq \overline{S_r}$ **then**
      Insert the violated cut $\mathbf{x}\left(\delta^-(S_v)\right) \geq 1$ into the LP
      $\tilde{i_v} = \arg\max_{i \in S_v} \tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^-(i)\right)$
      if $\tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^-(S_v)\right) < \tilde{\mathbf{y}}^{\mathbf{0}}\left(\delta^-\left(\tilde{i_v}\right)\right)$ **then**
        Insert the violated cut $\mathbf{y}^{\mathbf{0}}\left(\delta^-(S_v)\right) \geq \mathbf{y}^{\mathbf{0}}\left(\delta^-\left(\tilde{i_v}\right)\right)$ into the LP
      for all $k \in K$, $S_v \cap P^k \neq \emptyset$ **do**
        Insert the violated cut $\left(\mathbf{y}^{\mathbf{0}} + \mathbf{y}^{\mathbf{k}}\right)\left(\delta^-(S_v)\right) \geq 1$ into the LP
      $\tilde{x}'_{ij} = 1$, $\forall (i,j) \in \delta^-(S_v)$
until $f \geq 1$ or *MAX-CUTS* constraints added
Resolve the LP

$\mathbf{y^0}$-cuts and *scenario*-cuts. On the other hand, when performing the separation of $\mathbf{y^0}$-cuts in a later phase, we basically use the same idea to add violated *scenario*-cuts, whenever a violated $\mathbf{y^0}$-cut is detected.

Combinatorial Cuts The separation of *combinatorial cuts* relies on the following idea: if we knew the structure of the optimal spanning tree built in the first stage, to find the optimal recoverable robust solution it is sufficient to consider the cut-sets associated with the edges of that tree. Let $\tilde{T} = (\tilde{V}, \tilde{A})$ denote the rooted spanning arborescence associated with $\mathbf{x}$-variables of the optimal solution. Observe that the removal of an arc $(j, \ell) \in \tilde{A}$ separates $\tilde{T}$ into two components. Let V_ℓ be the set of nodes of the sub-arborescence rooted at ℓ, and K_ℓ be the set of *relevant scenarios*, i.e., $K_\ell = \{k \in K \mid V_\ell \cap P^k \neq \emptyset\}$. The values of the variables $\mathbf{y^0}$ and $\mathbf{y^k}$ could then be determined by solving the following Integer Program (IP):

$$\min \sum_{(i,j)\in\tilde{A}} u_{ij} y^0_{ij} + \max_{k\in K} \min \sum_{(i,j)\in\tilde{A}} r^k_{ij} y^k_{ij} \tag{3.9}$$

$$\text{s.t.} \quad (\mathbf{y^0} + \mathbf{y^k})(\delta^-(V_\ell)) \geq 1 \qquad \forall (j,\ell) \in \tilde{A},\ \forall k \in K_\ell \tag{3.10}$$

$$\mathbf{y^0}(\delta^-(V_\ell)) \geq \mathbf{y^0}(\delta^-(i)) \qquad \forall i \in V_\ell,\ \forall (j,\ell) \in \tilde{A} \tag{3.11}$$

$$\mathbf{y^0} \in \{0,1\}^{|\tilde{A}|}, \mathbf{y^k} \in \{0,1\}^{|\tilde{A}|} \qquad \forall k \in K \tag{3.12}$$

Obviously, in this model there are only $O(nK)$ constraints, and the associated sets V_ℓ

Algorithm 2 Combinatorial Cuts

Input: Graph $G_A = (V, A)$, $\tilde{T} = \left(\tilde{V}, \tilde{A}\right)$ a spanning arborescence of G_A, fractional solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}^{\mathbf{0}}, \tilde{\mathbf{y}}^{\mathbf{k}})$

$\mathcal{L} = \left\{v \in \tilde{V} \mid \delta^+(v) = \emptyset\right\}$
for all $\ell \in \mathcal{L}$ **do**
  $V_\ell = \{\ell\}$
  $K_\ell = \left\{k \in K \mid \ell \in P^k\right\}$
repeat
  Chose $\ell \in \mathcal{L}$
  Let j be the parent of ℓ in $\tilde{T}$, i.e., $(j, \ell) \in \tilde{A}$
  if $\tilde{y}^0_{j\ell} < 1$ **then**
    for all $k \in K_\ell$ **do**
      if $\left(\tilde{\mathbf{y}}^{\mathbf{0}} + \tilde{\mathbf{y}}^{\mathbf{k}}\right)(\delta^-(V_\ell))\big) < 1$ **then**
        Insert the violated cut $\left(\mathbf{y}^{\mathbf{0}} + \mathbf{y}^{\mathbf{k}}\right)(\delta^-(V_\ell))\big) \geq 1$ into the LP
  $K_j = K_\ell \cup \{k \in K \mid j \in P^k\}$
  $V_j = V_\ell \cup \{j\}$; $\mathcal{L} = \mathcal{L} \backslash \{\ell\}$; $\mathcal{L} = \mathcal{L} \cup \{j\}$
until $\mathcal{L} = \{r\}$

can be determined in $O(n)$ time using a dynamic programming procedure. Furthermore, formulation (3.9)-(3.12) is equivalent to formulation (T.1)-(T.4).

Since we do not know the structure of the optimal arborescence in the first stage, we try to heuristically approximate it and generate cut-sets of type (3.10) and (3.11) on graph G (with the heuristic tree) and insert them into the model. Thus, we are able to insert $O(nK)$ cuts into the LP, in $O(mK)$ running time. For good approximations of $\tilde{T}$, these combinatorial cuts can bring a significant speed-up to the separation procedure, especially in the early stages of the cutting plane algorithm. In Algorithm 2 the outline of the procedure is presented. The main idea of the algorithm is to recursively generate sets V_ℓ and K_ℓ and insert the violated cuts into the current LP. We start with the leaf nodes of $\tilde{T}$ and process the arborescence in a bottom-up fashion until reaching the root node. Whenever we process an arc of $\tilde{T}$, we insert violated cuts into the current LP. In total, each edge from G is "visited" at most twice and therefore, the total running time of this procedure is at most $O(mK)$.

Combinatorial cuts are separated together with $\mathbf{y}^{\mathbf{0}}$-cuts and before the (more time consuming) separation of *scenario*-cuts is performed. To approximate the tree $\tilde{T}$, we run the minimum spanning tree algorithm on G with edge weights set to

$$w_e = b_e \min\{(1 - \tilde{x}_{ij}), (1 - \tilde{x}_{ji})\} \text{ for each } e : \{i, j\} \in A, \tag{3.13}$$

where $\tilde{\mathbf{x}}$ is the value of the current fractional solution. Combinatorial cuts are also added, whenever in the current LP, $\mathbf{x}$ is a binary vector.

3.3.4 MIP Initialization

In our branch-and-cut approach we first drop all **x**-, $\mathbf{y^0}$- and *scenario*-cuts, and add them in an iterative fashion only when violated. However, to improve the quality of the lower bounds we incorporate additional constraints to the initial model. Since for the RRTLND problem the **x** variables should construct a spanning arborescence of G, the following in-degree constraints

$$\mathbf{x}\left(\delta^{-}(i)\right) = 1,\ \forall i \in V, \tag{3.14}$$

are valid inequalities that stress the tree-like topology of the corresponding solution. We also include the constraints

$$\left(y^0_{ij} + y^k_{ij}\right) + \left(y^0_{ji} + y^k_{ji}\right) \leq 1,\ \forall (i,j) \in A,\ \forall k \in K, \tag{3.15}$$

that correspond to subtour elimination constraints of size 2 for arcs with primary technology.

Finally, we also use combinatorial cuts described above as part of the initialization of the MIP model. The arborescence $\tilde{T}$ is approximated by the minimum spanning tree considering edge weights b_e, $\forall e \in E$. This initialization provides good initial lower bounds since many important cut-sets are inserted into the model at an early stage of the cutting plane procedure without the resolution of a maximum flow problem.

3.3.5 Primal Heuristic

An important component of our branch-and-cut is the embedded Primal Heuristic, whose pseudo-code is given in Algorithm 3. The core of the heuristic is to solve an instance of the RRTLND problem on an induced spanning arborescence $\tilde{T}$ of G_A to optimality. For constructing the spanning arborescence $\tilde{T}$ we use LP-values of **x** variables from the current LP relaxation. We run the minimum spanning tree algorithm on G with edge weights defined by (3.13) (Step 1 of the algorithm).

In the loop (2-4) the preprocessing described in §3.2.2.2 is applied. We find the optimal Steiner Tree (constructed by recovered edges) on $\tilde{T}$ considering terminal set P^k; ω_k denotes the corresponding total recovery cost for each scenario. The main step of the algorithm is Step 5, where the MIP problem (T.1)-(T.4) is solved. The feasible primal solution $\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}^{\mathbf{0}}, \hat{\mathbf{y}}^{\mathbf{k}}\right)$ of our problem is obtained by mapping the solution $\mathbf{s}^*$ and the structure of $\tilde{T}$ as shown in Step 6. All arcs in $\tilde{T}$ define the spanning arborescence associated with $\hat{\mathbf{x}}$. The values of $\hat{\mathbf{y}}^{\mathbf{0}}$ correspond to the values of $\mathbf{s}^*$ and the values of $\hat{\mathbf{y}}^{\mathbf{k}}$ are calculated by a simple inspection using the information contained in $\mathcal{P}_k$, $\forall k \in K$, and $\mathbf{s}^*$.

Algorithm 3 Primal Heuristic

Input: Graph $G_A = (V, A)$, fractional solution $\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}^{\mathbf{0}}, \tilde{\mathbf{y}}^{\mathbf{k}}\right)$, cost vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{u} = \mathbf{a} - \mathbf{b}$ and $\mathbf{r}$.
Output: A feasible solution $\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}^{\mathbf{0}}, \hat{\mathbf{y}}^{\mathbf{k}}\right)$ for the RRTLND problem

1: $\tilde{T} = (V_{\tilde{T}}, E(\tilde{\mathbf{x}})) =$`spanningTree`$(G, \mathbf{w})$, where $\mathbf{w}$ is defined by (3.13).
2: **for all** $k \in K$ **do**
3: $\quad \mathcal{P}_k =$ `steinerTree`$\left(P^k, \tilde{T}\right)$
4: $\quad \omega_k = \sum_{(i,j)\in\mathcal{P}_k} r^k_{ij}$
5: Solve problem (T.1)-(T.4) with $\tilde{T}$ as input graph, cost vectors $\mathbf{u}$ and $\mathbf{r}$, and vectors $\mathcal{P}_k$ and ω_k.
6: Let $\mathbf{s}^*$ be an optimal solution for (T.1)-(T.4) and $A(\tilde{\mathbf{x}})$ be the arcs of $E(\tilde{\mathbf{x}})$ oriented away from r. A feasible solution $\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}^{\mathbf{0}}, \hat{\mathbf{y}}^{\mathbf{k}}\right)$ for the RRTLND problem is defined by $\hat{x}_{ij} = 1$ if $(i,j) \in A(\tilde{\mathbf{x}})$ and $\hat{x}_{ij} = 0$ otherwise, $\hat{y}^0_{ij} = 1$ if $s^*_{ij} = 1$ and $\hat{y}^0_{ij} = 0$ otherwise, $\hat{y}^k_{ij} = 1$ if $(i,j) \in \mathcal{P}_k$ and $s^*_{ij} = 0$ and $\hat{y}^k_{ij} = 0$ otherwise.

Although we know that the problem is NP-Hard, in practice the computational effort to solve the problem is remarkably little (usually only a fraction of a second or a couple of seconds). This makes the primal heuristic very effecive since feasible solutions are quickly computed.

3.4. The RR Two-Level Steiner Tree Problem

In some real-world instances of the TLND problem, in addition to the customer nodes, there are additional nodes in the network (corresponding to street intersections, for example) that do not require any service. The definition of the TLND problem can be extended correspondingly. In this variant of the TLND problem that we refer to as the *Two-Level Steiner Tree* (TLStT) problem, we are given a set $R \subset V$ representing the customers that have to be served either by primary or secondary technology. The set of primary customers, P, is such that $P \subseteq R$. The goal is to find a minimum-cost Steiner tree in G spanning all nodes from R and such that all nodes from P are connected with each other using the primary technology. Using the notation presented before, binary vector $\mathbf{X}$ instead of being associated with a spanning tree of G is now associated with a Steiner tree connecting nodes from R. The remaining conditions remain the same ($\mathbf{Y}$ is associated with a Steiner Tree connecting P, $\mathbf{Y} \leq \mathbf{X}$, and the objective function is given by (3.1)). Those nodes that do not belong to R but that are spanned by a solution given by a pair $(\mathbf{Y}, \mathbf{X})$ are called *Steiner-nodes*.

The RRTLStT Problem For the Recoverable Robust counterpart of the TLStT problem (RRTLStT) the set $R \subset V$ is given at the outset, while the set of primary customers is only determined after the uncertainty is resolved (i.e., the scenarios are such that $P^k \subseteq R$, $\forall k \in K$).

The MIP formulation provided for the RRTLND problem can be easily adapted for the RRTLStT problem by imposing that feasible values of vector $\mathbf{x}$ instead of being associated with a spanning arborescence of G_A, have to instead be associated with a

Steiner arborescence of set R. This is expressed by replacing $\mathbf{x}$-cuts by

$$\mathbf{x}\left(\delta^{-}(S)\right) \geq 1, \forall S \subseteq V \setminus \{r\}, S \cap R \neq \emptyset. \tag{3.16}$$

This set of constraints, which we call $\mathbf{x_R}$-cuts, ensures that there is a directed path from r to every node in $R\backslash\{r\}$.

In the algorithmic framework outlined in §3.3.2 some procedures should be adapted for solving the RRTLStT problem. In the MIP initialization, the "=" sign in (3.14) should be replaced by "≤". In the separation described in Algorithm 1, $\mathbf{x_R}$-cuts are separated instead of $\mathbf{x}$-cuts; in this case instead of selecting a random node v in $V\backslash\{r\}$ and performing the separation from r to v, the separation is performed from r to every node in $v \in R\backslash\{r\}$. When applying Combinatorial-Cuts, instead of giving as input a spanning arborescence $\tilde{T}$ of G_A, we give as input a Steiner arborescence which spans all nodes in R; this arborescence is found by means of an algorithm that succesively solves shortest-path problems from r to $v \in R\backslash\{r\}$ with arc costs given by (3.13) and merges these paths to conform an arborescence of G_A spanning R. The same idea is used in our primal heuristic, in which instead of finding an spanning arborescence of G_A we find a Steiner arborescence connecting nodes in R.

3.5. Computational Results

In this section we report on our computational experience on two sets of instances that are used to test the branch-and-cut algorithm for both, the RRTLND problem and the RRTLStT problem.

All the experiments were performed on an Intel Core$^{\text{TM}}$ i7 (2600) 3.4GHz machine with 16 GB RAM, where each run was performed on a single processor. The branch-and-cut was implemented using CPLEX$^{\text{TM}}$ 12.3 and Concert Technology framework. All CPLEX parameters were set to their default values, except the following ones: (i) All cuts were turned off, (ii) heuristics were turned off, (iii) preprocessing was turned off, (iv) time limit was set to 1800 seconds, and (v) higher branching priorities were given to $\mathbf{y^0}$ variables. We have turned these CPLEX features off in order to make a fair assessment of the performance of the techniques described in §3.3.2.

Interestingly, and somewhat to our surprise, it turns out that turning on CPLEX cuts and heuristics actually slows down the performance of our algorithm. With regards to the branching priorities, notice that whenever a variable, say y^0_ℓ, is fixed to one, variables x_ℓ and y^k_ℓ, $\forall k \in K$ can be immediately fixed as well ($x_\ell = 1$ and $y^k_\ell = 0$ $\forall k \in K$). Furthermore, we know that only a few $\mathbf{y^0}$ variables (at most $n-1$) will, at the end of the optimization, take the value of 1. Therefore, we give higher branching priorities to these variables, as they mostly influence the overall solution structure and reduce the underlying search space.

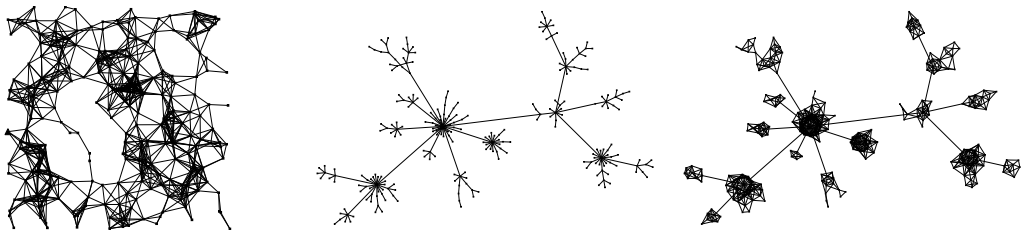
3.5.1 Instances

We consider two classes of randomly generated instances, that we refer to as `G` and `SC` instances. Their topologies resemble different geographic local structures of communication and distribution networks.

`G` Instances This group of instances is generated following a similar scheme as [Johnson et al., 2000] (where the authors intended to generate instances that coincide with the street maps of real-world instances). Here n nodes are randomly located in a unit Euclidean square. An edge e between two nodes is created if the Euclidean distance between them is no more than $\alpha/\sqrt{n}$, for a fixed $\alpha > 0$. Coordinates are generated with five significant digits. The secondary cost of an edge b_e corresponds to the Euclidean distance between its end points multiplied by 10^4 and rounded to the closest integer; the primary cost a_e is calculated as $(1+\beta)\, b_e$, where $\beta \in [0,1]$ is a pre-defined parameter and the recovery cost $r_e = r_e^k$ is assumed to be equal for all $k \in K$ and is set to $(\epsilon+\beta)\, b_e$, for a fixed $\epsilon \in [0,1]$. Both, primary and recovery costs are rounded to the nearest integer value. A single node is randomly selected and chosen to be the root node r. For the RRTLND problem, in each scenario $k \in K$, $\pi\%$ of nodes are uniformly randomly selected from V to constitute the set of primary nodes P^k. For the RRTLStT problem, the set R of *all potential customers* is constructed by uniformly randomly selecting $\varphi\%$ of all nodes from V. Similarly as for the RRTLND problem, $\pi\%$ of all nodes from R are then uniformly randomly selected to build the set P^k, for each $k \in K$.

In our experiments we consider the following parameter settings: $\beta \in \{0.5, 1.0, 2.0, 3.0\}$, $\epsilon \in \{0.5, 1.0, 2.0, 3.0\}$ (which produces $r_e/u_e \in \{7/6, \ldots, 7\}$), $\pi \in \{10\%, 20\%, 30\%\}$, and $\varphi = 50\%$. Four instances were generated for each combination of those parameters. Graphs of different size are considered as well. We choose $n \in \{50, 75, 100, 250\}$ and set $\alpha = 0.6$. The value of α is incremented in steps of 0.001 until a connected graph is obtained (in only one case, for $n = 250$, 0.6 was not enough to define a connected graph and the real value of α was 0.613). This leads to 192 instances for a given n. Figure 3.2(a) illustrates an example of a graph with 250 nodes and $\alpha = 0.6$ (which produces 1134 edges).

`SC` Instances These instances are generated on the basis of the well-known *scale-free* networks [see Barabasi and Albert, 1999]. *Scale-free* networks frequently appear in the context of complex systems, including the World Wide Web, the internet backbone, infrastructure networks, airline connections, cellular networks, wireless networks, electric-power grids and many other contexts. Using the `igraph` library package [see `igraph` Project, 2012] a *scale-free* graph of n nodes is created using default settings. This actually produces a tree since linear preferential attachment (*power-law* equal 1) is the default parameter for the generation. The resulting graph is simply an array of binary relations. We then use the yEd Graph Editor software [see yWorks, 2012]



(a) `G` instance with $\alpha = 0.6$. (b) *Scale-free* tree with $n = 250$. (c) `SC` instance with $\alpha = 0.2$.

FIGURE 3.2: Examples of the generated instances.

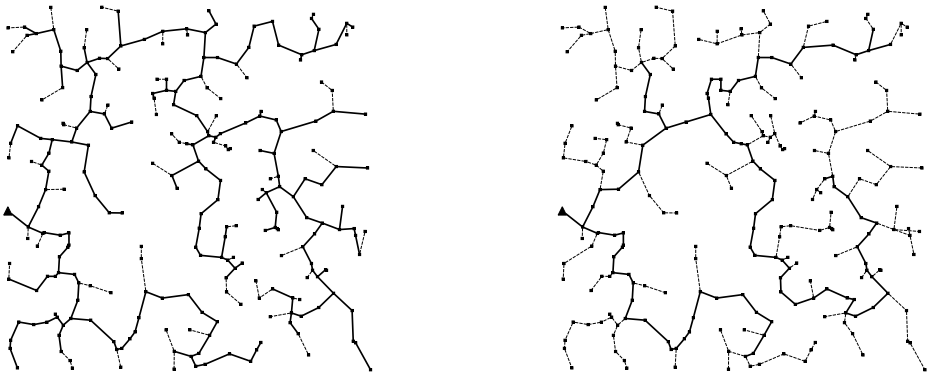
and draw the tree using the "organic" layout. This layout determines node coordinates which are then used to add additional edges and augment the tree. A new edge between two nodes is added if its Euclidean distance is no more than $\alpha/\sqrt{n}$. The root node corresponds to the node with label 0 in the *scale-free* tree. Edge costs (b_a, a_e, r_e) and scenarios for both the RRTLND problem and the RRTLStT problem are generated identically as the `G` instances.

In Figure 3.2(b) we show a *scale-free* tree with a layout fixed by yEd and in Figure 3.2(c) the same instance augmented with a set of complementary edges (922 in total). For $n = 50, 75, 100, 250, 500, 750, 1000$ we use $\alpha = 0.1, 0125, 0.15, 0.2, 0.3, 0.35, 0.4$ respectively. The other parameters were set as in the case of the `G` instances. Four instances were generated for each combination of the parameters n, π, β and ϵ. This leads to 192 instances for a given n.

3.5.2 Robustness and Recoverability

In our computations we consider up to 30 scenarios which are created in advance. By doing this, when considering problems with 10 scenarios, we simply use the first 10 scenarios out of those 30. The same applies when considering 20 scenarios. The scenarios are identical for the different values of β and ϵ. By proceeding in this way, it is easier to measure the impact of considering a larger number of scenarios.

The way that *robust* first-stage solutions and the corresponding recovery actions are calculated depends not only on the scenario structure but also on the cost structure; the relations between a_e, b_e and r_e. If the recovery costs r_e are high compared to the first-stage upgrade costs $u_e = a_e - b_e$, then the solutions of the RRTLND problem are more likely to have a larger first-stage primary tree. On the contrary, if recovery is relatively cheap, then the optimal solutions will be comprised by a smaller first-stage primary tree and more recovery actions will be performed (as in a wait-and-see approach). This can be seen when comparing the solutions in Figures 3.3(a) and 3.3(b) of a 250 nodes `G` instance with 20 scenarios. In the first case, recovering an edge in the second stage is seven times more expensive than installing a primary technology in the first stage (which is 50% more expensive than secondary technology), consequently the



(a) $\beta = 0.5$, $\epsilon = 3$, $OPT_{RR} = 50982$, $\left|E(\mathbf{y^0})\right| = 187$, $\left|E(\mathbf{y^k})\right| / |K| = 4$.

(b) $\beta = 3$, $\epsilon = 0.5$, $OPT_{RR} = 96795$, $\left|E(\mathbf{y^0})\right| = 95$, $\left|E(\mathbf{y^k})\right| / |K| = 34$.

FIGURE 3.3: Examples of the solution of the RRTLND problem for a `G` instance with 250 nodes and $\alpha = 0.6$, $|K| = 20$, with different values of β and ϵ. Bold edges correspond to first-stage primary edges, dashed edges are secondary edges that might be recovered in some scenarios.

first-stage primary tree (bold edges, $E\left(\mathbf{y^0}\right)$), spans a large portion of the graph (186 nodes) and only a few recovery actions are needed per scenario ($\left|E(\mathbf{y^k})\right| / |K| = 4$). The opposite occurs in the second case, when recovery cost is slightly more expensive than the upgrade cost (which is four times more expensive than secondary cost); in this case, the $E\left(\mathbf{y^0}\right)$ component is smaller, spanning only 94 nodes, and much more recovery actions take place in each scenario ($\left|E(\mathbf{y^k})\right| / |K| = 37$). The differences in the value of the objective functions, OPT_{RR}, can be explained similarly.

In Table 3.1 we report average values of the experimental results obtained for the RRTLND problem for classes `G` and `SC` considering different number of nodes and different number of scenarios (columns Class, n and $|K|$ respectively). The presented statistics concern the solution characteristics as well as indicators of the algorithmic performance. Column m corresponds to the average number of edges among the instances created for each value of n. Column Gap(%) shows the average gap obtained after the time limit of 1800 seconds is reached. This average is calculated over 64 instances per each group. The corresponding average running times are shown in seconds in column Time(s). The average size of the first-stage primary subtree of the optimal, or best known feasible solution, is indicated in column $\left|E(\mathbf{y^0})\right|$. The mean number of recovery actions performed in each scenario can be expressed by $\left|E(\mathbf{y^k})\right|$ divided by $|K|$; the average values of this measure, for the optimal or best known solution, are reported in column $\left|E(\mathbf{y^k})\right| / |K|$. In column #Opt the number of problems that can be solved to optimality (out of 64 for each row) is shown.

A first-stage solution is expected to be more robust with respect to data perturbations if more scenarios (possible data realizations) are taken into account. However, this robustness is not for *free*. On the one hand the difficulty of the problem increases since a larger search space should be considered; while on the other hand, the cost of the

Class	n	m	$\|K\|$	Gap(%)	Time(s)	$\|E(\mathbf{y^0})\|$	$\|E(\mathbf{y^k})\| / \|K\|$	PH(%)	#BBN's	#(3.3)	#(3.4)	#(3.5)	#Opt
G	50	163	10	0.01	31.27	17	5	7.62	367	25	131	1459	64
			20	0.01	222.60	17	5	6.79	837	30	207	4314	63
			30	0.01	230.31	18	5	7.5	642	28	181	5554	64
	75	257	10	0.01	53.11	24	6	7.91	471	50	151	1986	64
			20	0.09	540.85	25	6	7.02	1197	54	272	6407	56
			30	0.24	1004.85	25	6	6.78	1416	57	264	9292	40
	100	356	10	0.01	458.67	35	9	7.54	1491	105	292	4136	62
			20	0.36	1470.20	35	10	6.61	1056	105	344	9066	23
			30	0.83	1780.54	36	10	6.95	434	103	271	11426	2
	250	1114	10	0.86	†	90	23	9.81	237	64	172	6861	0
			20	6.10	†	111	23	11.26	15	36	37	7497	0
			30	10.67	†	119	23	15.28	5	24	13	6995	0
SC	50	175	10	0.00	32.31	11	6	5.93	198	2	38	409	64
			20	0.01	81.68	11	6	7.48	439	1	63	1162	64
			30	0.01	150.98	12	6	6.45	769	1	84	2167	64
	75	287	10	0.01	196.53	17	8	7.04	4016	15	109	1202	63
			20	0.02	470.32	18	9	7.05	1460	15	151	3126	61
			30	0.08	820.36	18	9	7.23	1486	15	185	5446	49
	100	410	10	0.01	452.42	23	11	7.45	2490	13	149	1540	61
			20	0.08	878.75	25	11	7.75	2731	11	179	3559	42
			30	0.14	1177.93	24	12	7.7	1779	10	212	6096	32
	250	932	10	0.07	1778.68	60	29	5.76	1313	59	288	3826	1
			20	0.17	†	62	32	5.63	518	55	217	6342	0
			30	0.26	†	63	32	5.54	228	53	121	6896	0
	500	2345	10	0.06	†	124	58	5.44	385	36	243	5689	0
			20	0.26	†	126	63	5.26	16	20	93	7706	0
			30	1.53	†	142	65	5.36	1	15	68	9289	0
	750	3460	10	0.08	†	189	87	5.39	132	38	210	7095	0
			20	0.95	†	209	94	5.39	4	20	94	10051	0
			30	3.50	†	241	94	6.38	1	11	50	10622	0
	1000	4658	10	0.16	†	261	114	5.58	65	36	201	8792	0
			20	2.34	†	308	125	5.83	1	16	88	11970	0
			30	6.64	†	367	124	8.34	0	7	33	9911	0

TABLE 3.1: Solution characteristics and algorithm performance averages for different values of $|K|$ for classes G and SC (RRTLND problem, $\pi = 10\%$, $\beta, \epsilon \in \{0.5, 1, 2, 3\}$). †: Time limit (1800 s).

solutions, OPT_{RR}, increases due to a possible enlargement of the first-stage primary component or because a new worst-case scenario induces a higher robust recovery cost. The first phenomenon is what we call the *Effort for Robustness*. Table 3.1 demonstrates this phenomenon. Increasing the number of scenarios results in a deterioration of the algorithmic performance for both classes of instances: (i) the average running times increase (this is more apparent in the case of small instances, which could be solved to optimality within the time limit); (ii) the average gap of the obtained solutions deteriorates; and, therefore, (iii) the number of solution for which the proof of optimality is obtained decreases. From the perspective of the solutions structure and the corresponding cost, from columns $\left|E(\mathbf{y^0})\right|$ and $\left|E(\mathbf{y^k})\right| / |K|$ we can see the size of the first-stage primary tree is almost constant for a given n, as well as the average number of recovery actions performed by scenario. The fact that the average values are almost constant for a given n means that our recoverable robust solutions are protected against data perturbation and are able to balance robustness and recoverability: the robust first-stage solutions and their corresponding recovery actions depend more on the cost structure (as shown in the Figure 3.3 example) than on the level of uncertainty. Nevertheless, the absolute number of recovery actions ($\left|E(\mathbf{y^k})\right|$) increases proportionally to $|K|$, which means that the cost of the corresponding solutions is also

		Running times statistics ($t \leq 1800s$)					Gap (%) statistics ($t > 1800s$)				
Class	$\|K\|$	#	Min	Median	Mean	Max	#	Min	Median	Mean	Max
G	10	190	0.30	40.11	164.00	1690.00	66	0.02	0.25	0.84	12.39
	20	142	2.56	189.40	372.80	1605.00	114	0.07	0.62	3.68	22.98
	30	106	6.79	216.80	360.00	1594.00	150	0.07	0.78	5.01	29.60
SC	10	189	0.72	60.00	194.20	1787.00	259	0.01	0.05	0.10	1.14
	20	167	4.60	125.60	278.70	1758.00	281	0.03	0.22	0.87	6.39
	30	145	5.59	222.20	364.80	1535.00	303	0.03	0.90	2.56	13.86

TABLE 3.2: Running times and gap statistics of all instances of classes G and SC for different values of $|K|$ ($\pi = 10\%$, $\beta, \epsilon \in \{0.5, 1, 2, 3\}$, RRTLND problem)

likely to increase due to the augmentation of the worst-case recovery cost induced by a new scenario.

Table 3.2 provides further analysis on the the impact of $|K|$ on the algorithmic performance. We report the statistics (the number of instances (#), min, median, mean and max values) of the running times of those problems that are solved to optimality and the statistics of the gaps of those problems that cannot be solved within 1800 seconds; these statistics are summarized for all values of n, β, ϵ and π, for the two classes of instances. Hence, each row summarizes statistics over 256 instances of each group. As observed before, increasing the number of scenarios, $|K|$, clearly deteriorates the performance of the algorithm: the median and mean running times of those problems that are solved to optimality increase notably; while the median, mean and maximum gaps of those problems that cannot be solved within the time limit, and their quantity also increases.

In Table 3.3 we report basic statistics (Min, Median, Mean, Max) of the values of the *Gain of Recovery* of the recoverable robust solutions with respect to the AR and W&S approaches for a subset of instances of classes G and SC. The economical advantage of the RR solutions is clearly shown by the reported values: both AR and W&S solutions are, in general, more than 15% more expensive than the recoverable robust ones. Moreover, the AR solutions can be 39% more expensive, while the W&S solutions 49% more expensive! This means that the recoverable robustness approach is able to provide economically robust solutions by means of balancing first-stage and second-stage actions depending on the cost and scenario structure.

3.5.3 Algorithmic Performance

More specific performance measures are presented in the remaining columns of Table 3.1. In column PH(%) we report the average gap between the *initial upper bound* (obtained by running Algorithm 3 in which $\mathbf{w} = \mathbf{b}$ in Step 1) and the optimal, if known, or the best lower bound attained within the time limit. The average number of nodes of the branch-and-bound tree is shown in column #BBN's. In columns #(3.3),

			GoR with respect to AR				*GoR* with respect to W&S			
Class	n	$\|K\|$	Min	Median	Mean	Max	Min	Median	Mean	Max
G	75	10	0.00	13.56	13.45	29.89	3.58	16.58	18.48	44.04
		20	1.01	19.35	18.77	37.15	3.52	17.09	19.27	45.16
		30	0.84	20.89	19.84	39.39	3.62	17.46	18.98	44.82
	100	10	0.00	13.15	13.57	31.44	2.39	16.97	18.54	45.35
		20	0.79	19.28	18.77	36.68	3.18	18.66	19.92	45.27
		30	0.96	19.79	19.50	37.58	4.85	19.26	20.29	47.55
SC	75	10	0.00	13.23	13.65	33.71	3.25	15.37	17.32	43.53
		20	0.00	17.56	17.38	38.08	3.28	15.60	17.60	44.46
		30	1.01	20.22	19.23	38.43	1.70	15.85	17.73	43.56
	100	10	0.39	13.81	13.86	31.18	4.31	19.04	21.14	49.02
		20	0.71	16.57	16.14	32.26	3.44	19.88	21.36	48.09
		30	1.03	17.85	17.44	34.24	3.00	19.31	20.87	47.44

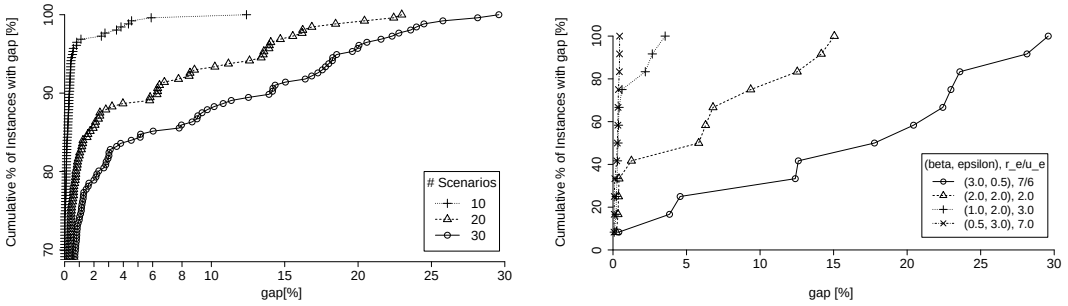
TABLE 3.3: Gain of Recovery with respect to the Absolute Robustness and the Wait-and-See approaches for instances of classes G and SC (RRTLND problem, $\pi = 10\%$, $\beta, \epsilon \in \{0.5, 1, 2, 3\}$).

#(3.4) and #(3.5) we summarize the average number of **x**-, $\mathbf{y^0}$- and *scenario*-cuts, respectively, that are added during the optimization process.

As discussed in §3.3.2, one of the main features of our branch-and-cut is the embedded primal heuristic. From column PH(%) we observe that, in most cases, this average value is below 10%, which reinforces our conviction that this procedure is crucial as part of the algorithmic approach. These initial upper bounds can be obtained in a couple of seconds or even fractions of a second for small instances.

For small instances (50 and 75 nodes for G instances, and 50 nodes for SC instances), we notice that the number of nodes of the branch-and-bound tree increases with the number of scenarios. However, for larger instances the situation is the opposite: an increased number of scenarios implies a reduced value of #BBN's. The more scenarios we consider, the more complex the problem is. In some cases, especially for the largest instances, only a few nodes are explored or, even worse, no branching is performed and the optimization terminates while cutting planes are still being added at the root node.

With respect to the separation of **x**-, $\mathbf{y^0}$- and *scenario*-cuts, the first observation is that, for small and medium size instances, when increasing the number of scenarios the number of **x**- and $\mathbf{y^0}$-cuts that are added is approximately constant and, the number of *scenario*-cuts increases proportionally. Additionally, as might be expected, for a given n and a given $|K|$, more *scenario*-cuts are added than $\mathbf{y^0}$-cuts, and more $\mathbf{y^0}$-cuts are added than **x**-cuts. These behaviors are not verified for larger instances, which is probably due to the fact that in this case the separation is mainly performed at the root node, while it is actually during branching that the separation reaches a more stable behavior. Despite the differences, it is interesting to notice that, in general, not many cutting planes are needed to obtain strong lower bounds, which is the case for a large percentage of instances.



(a) All instances of group G for different values of $|K|$ ($\pi = 10\%$, $\beta, \epsilon \in \{0.5, 1, 2, 3\}$, RRTLND problem)).

(b) Influence of β and ϵ in the algorithm performance for the 250 nodes group of class G ($|K| = \{10, 20, 30\}$, $\pi = 10\%$, RRTLND problem).

FIGURE 3.4: Cumulative percentage of instances with a given gap (%) obtained within the time limit for the RRTLND problem

In Figure 3.4(a), we show the cumulative percentage of problems of class G, for different values of $|K|$, for which we reach less than a given gap (%) within the time limit (for each number of scenarios there are 256 problems to be solved in class G). This complements the information presented in Tables 3.1 and 3.2 about the average gap in relation to the number of scenarios. For 10 scenarios, we notice that more than 95% of problems can be solved with less than a 2% gap within the time limit, and only a few outliers present gaps greater than 5%. When considering problems with 20 scenarios approximately 85% of the instances are solved to within a 2% gap in the time limit. In this case, almost 10% of the instances present a gap larger than 10%, which can be even higher than 20% for a few cases (less than 2% of the problems). However, when considering $|K| = 30$, the quality of the solutions significantly deteriorates. More than 15% of the instances present gaps greater than 10% when reaching the time limit, and these gaps are even higher than 25% for a few problems.

Figure 3.4(b) considers the group of instances with 250 nodes of class G (considering $|K| = \{10, 20, 30\}$ and $\pi = 10\%$) and provides the cumulative percentage of problems (%), for four combinations of β and ϵ, for which we reach less than a given gap (%) within the time limit. It follows that when the recovery costs are significantly higher than the upgrade costs the problem turns out to be easier to solve. This can be explained by the fact that if recovery costs are expensive, then the induced solutions tend to be comprised of a larger first-stage primary component (reducing the number of recovery actions, see Fig. 3.3(a)). These solutions have a closer resemblance to the easier deterministic TLND problem with $P = \bigcup_{k \in K} P^k$. On the other hand, when recovery costs are more "comparable" to first-stage upgrade costs the structure of solutions has more of a "wait-and-see" flavor: the first-stage primary component is smaller and a large number of recovery actions is performed in the second stage; this emphasizes the combinatorial nature of the problem and it makes the optimization task harder.

		Running times statistics ($t \leq 1800s$)					Gap (%) statistics ($t > 1800s$)				
Class	π	#	Min	Median	Mean	Max	#	Min	Median	Mean	Max
G	10%	87	6.01	368.80	555.80	1690.00	105	0.07	0.48	0.73	5.18
	20%	57	5.85	486.80	605.80	1630.00	135	0.02	0.47	0.60	3.95
	30%	64	6.13	506.30	649.10	1790.00	128	0.01	0.37	0.43	1.41
SC	10%	135	12.01	263.60	429.00	1787.00	57	0.03	0.21	0.23	0.67
	20%	71	1.23	486.60	520.20	1785.00	121	0.01	0.23	0.32	3.22
	30%	62	0.97	369.10	524.00	1744.00	130	0.03	0.20	0.22	0.54

TABLE 3.4: Influence of the value of π on the algorithmic performance for instances with 100 nodes of both classes G and SC ($\beta, \epsilon \in \{0.5, 1, 2, 3\}$, $|K| = \{10, 20, 30\}$, RRTLND problem).

In all the results analyzed so far, we have considered $\pi = 10\%$ (in each scenario 10% of the nodes are primary nodes). However, and in order to provide an accurate evaluation of our algorithm we have performed computations by also considering $\pi = 20\%$ and $\pi = 30\%$. For both class G and class SC we selected the group of instances with 100 nodes and tested the developed algorithm for $\beta, \epsilon \in \{0.5, 1, 2, 3\}$ and $|K| = \{10, 20, 30\}$, considering $\pi = 20\%$ and $\pi = 30\%$. For each value of π, 256 problems are solved. In Table 3.4 we report the statistics regarding the running times of those instances that are solved to optimality and the statistics of the gaps of those that reached the time limit before optimality. We observe that increasing the fraction of nodes that are primary in each scenario results in a fewer number of instances that are solved to optimality. However, the gap statistics (over the instances not solved to optimality) are similar for different values of π, in particular the median and mean values remain in all cases below 1%. Hence we may conclude that the overall quality of the solutions produced by our algorithm is not significantly affected for different values of π.

To give clear insights about the utility of the specific separation strategies designed for our algorithmic framework (Mixed Separation and Combinatorial Cuts) Table 3.5 provides a comparison scheme that helps to evaluate the improvement of the algorithmic performance when including these two procedures. We have selected the groups of instances of class G with 50, 75 and 100 nodes and considered $\beta, \epsilon \in \{0.5, 1, 2, 3\}$, $|K| = \{10, 20, 30\}$, $\pi = 10\%$; therefore, 192 problems were solved for each value of n. Rows denoted by "Basic" correspond to the results obtained without Mixed Separation and Combinatorial Cuts, rows "+Mixed Sep." represent those results obtained when Mixed Separation is included in the separation as described in §3.3.3, and in rows "+Comb. Cuts" we report the results obtained when also the Combinatorial Cuts are included. The most important indicator is the number of instances that can be solved to optimality and the average time needed to solve them. As can be seen the performance of the algorithm notably improves when mixed separation and combinatorial cuts are turned on. For the group of instances with 250 nodes, only the gaps are compared since no instance could be solved to optimality. Before we conclude this section, we should note that our instances are motivated by real-world applications and are thus somewhat sparse. It should be clear that as graph density increases the number of variables in our model will increase and the performance of the branch-and-cut

		Running times statistics ($t \leq 1800s$)					Gap (%) statistics ($t > 1800s$)				
n	Separation Strategy	#	Min	Median	Mean	Max	#	Min	Median	Mean	Max
50	Basic	56	2.51	129.00	290.70	1487.00	136	0.01	0.21	0.25	0.99
	+ Mixed Sep.	186	1.44	153.40	267.50	1550.00	6	0.08	0.24	0.25	0.50
	+ Comb. Cuts	191	0.30	62.79	152.80	1589.00	1	0.46	0.46	0.46	0.46
75	Basic	56	4.23	342.50	463.00	1700.00	136	0.01	0.15	0.27	2.84
	+ Mixed Sep.	142	4.09	275.60	430.10	1712.00	50	0.05	0.45	0.60	3.01
	+ Comb. Cuts	160	0.98	128.20	279.50	1594.00	32	0.07	0.45	0.63	3.02
100	Basic	40	27.13	457.40	650.60	1724.00	152	0.01	0.38	0.59	6.09
	+ Mixed Sep.	64	27.80	527.90	637.50	1773.00	128	0.03	0.59	0.89	5.14
	+ Comb. Cuts	87	6.01	368.80	555.80	1690.00	105	0.07	0.48	0.73	5.18
250	Basic	0	-	-	-	-	192	0.03	16.36	17.70	44.50
	+ Mixed Sep.	0	-	-	-	-	192	0.02	3.69	7.99	30.65
	+ Comb. Cuts	0	-	-	-	-	192	0.02	0.99	5.88	29.60

TABLE 3.5: Impact of the branch-and-cut strategies on the algorithmic performance for instances of class `G` ($\beta, \epsilon \in \{0.5, 1, 2, 3\}$, $|K| = \{10, 20, 30\}$, $\pi = 10\%$, RRTLND problem).

algorithm will deteriorate.

3.5.4 Results for the RRTLStT Problem

For the RRTLStT problem, we performed the same computational experiments as the RRTLND problem. The corresponding adaptations of the branch-and-cut algorithm were previously described in §3.4.

Robustness and Recoverability As expected, for the RRTLStT problem the *Effort for Robustness* is *paid* as well. As for the RRTLND problem, increasing the number of scenarios results in a deterioration of the algorithmic performance which can be seen from the columns Gap(%), Time(s) and #Opt of Table 3.6. In general, the average value of these indicators are slightly better than for the RRTLND problem.

A deeper analysis can be done on the basis of the results presented in Table 3.7, where statistics of the running times and of the gaps are presented for both classes of instances. We observe that the number of instances that are solved to optimality decreases and the gap of those that are not solved to optimality increases when increasing $|K|$. These measures are quite similar to those for the RRTLND problem in the case of `G` instances; but it seems that on average for the `SC` instances the effort for robustness is "lower" than for the RRTLND problem.

As can be seen from the columns $\left|E(\mathbf{y^0})\right|$ and $\left|E(\mathbf{y^k})\right| / |K|$ of Table 3.6, just like the RRTLND problem there is a clear balance between the robustness of the first-stage solutions and their recoverability. In this case, again the cost structure has more influence on the configuration of solutions than the level of uncertainty.

Algorithmic Performance For the class `G` the average values of PH(%) in Table 3.6 are considerably worse than those for the RRTLND problem presented in Table 3.1 (the values are almost doubled). Nevertheless, for the case of class `SC` the first primal

Class	n	m	$\|K\|$	Gap(%)	Time (s)	$\|E(\mathbf{y^0})\|$	$\|E(\mathbf{y^k})\|$ / $\|K\|$	PH(%)	#BBN's	#(3.16)	#(3.4)	#(3.5)	#Opt
G	50	163	10	0.00	24.01	12	3	13.25	677	207	60	330	64
			20	0.00	51.73	13	3	11.32	202	218	98	867	64
			30	0.00	80.54	13	3	10.36	288	227	110	1369	64
	75	257	10	0.00	78.52	18	4	13.62	260	318	125	1496	64
			20	0.14	768.01	18	4	13.49	577	352	220	3973	49
			30	0.38	1250.94	19	4	13.81	274	346	195	5470	33
	100	356	10	0.01	341.89	22	6	16.65	637	732	338	1421	64
			20	0.34	1154.46	22	6	16.31	653	726	365	3315	34
			30	1.00	1435.43	22	6	17.05	323	687	287	4379	21
	250	1114	10	1.45	†	55	14	19.24	204	505	0	4604	0
			20	7.58	†	64	14	22.90	8	252	0	5157	0
			30	13.24	†	71	14	28.16	1	157	0	5262	0
SC	50	175	10	0.00	21.77	7	3	3.93	251	167	27	96	64
			20	0.00	31.07	7	4	4.01	58	165	32	209	64
			30	0.00	42.73	8	3	4.15	154	163	47	356	64
	75	287	10	0.00	62.38	10	5	5.34	124	371	57	302	64
			20	0.00	120.31	10	6	4.63	352	364	77	631	64
			30	0.01	155.90	10	5	4.7	282	373	87	1009	63
	100	410	10	0.01	159.72	12	7	2.87	5192	501	84	318	63
			20	0.01	276.45	12	7	3.09	3372	511	130	728	60
			30	0.01	302.83	12	7	3.66	2013	496	136	1142	62
	250	932	10	0.04	1050.02	29	18	4.15	1581	1782	315	1194	45
			20	0.12	1400.67	28	18	4.16	1025	1786	315	2347	25
			30	0.29	1581.26	31	19	4.12	448	1733	245	3151	19
	500	2345	10	0.10	1778.98	61	37	5.98	972	425	0	4179	2
			20	0.19	†	61	39	5.79	56	188	0	6262	0
			30	0.72	†	62	39	5.85	3	125	0	8074	0
	750	3460	10	0.18	†	95	56	5.97	172	454	0	5344	0
			20	1.15	†	101	57	6.35	1	164	0	8436	0
			30	3.75	†	111	60	8.02	0	94	0	10196	0
	1000	4658	10	0.59	†	134	68	6.73	95	570	0	7245	0
			20	2.37	†	141	76	7.73	1	193	0	11440	0
			30	6.52	†	160	81	10.83	0	103	0	12947	0

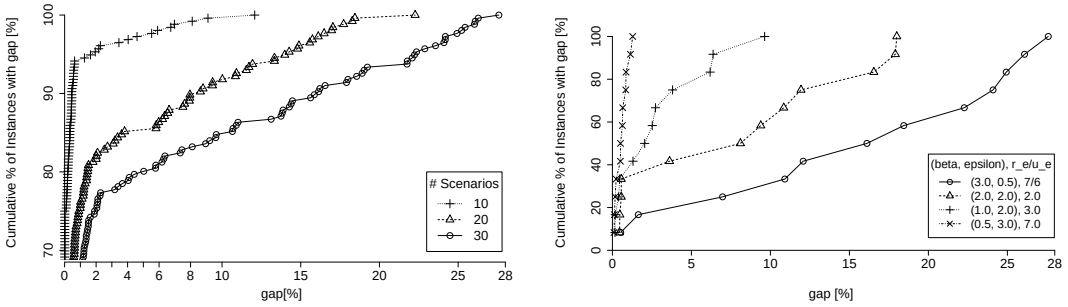
TABLE 3.6: Solution characteristics and performance measures for different values of $|K|$ for classes G and SC (RRTLStT problem, $\pi = 10\%$, $\beta, \epsilon \in \{0.5, 1, 2, 3\}$). †= Time limit (1800 s).

solutions are, on average, as good as for the RRTLND problem. The fact that $\mathbf{x}$, instead of defining a spanning arborescence on G_A, actually defines a Steiner arborescence on R, helps to explain this. The initial secondary Steiner arborescence on which we calculate the corresponding feasible solution is obtained by means of a heuristic procedure as explained in §3.4; while in the case of the RRTLND problem we find the primal solution on the optimal spanning arborescence with costs equal to b_e, $\forall e \in E$.

The average number of explored branch-and-bound nodes (column #BBN's) has, more or less, the same order of magnitude and the same dependance on n and $|K|$, as in the case of the RRTLND problem. From columns #(3.16), #(3.4) and #(3.5), where the average numbers of inserted $\mathbf{x_R}$-, $\mathbf{y^0}$- and *scenario*-cuts are reported, we notice that the separation process behaves differently from the one of the RRTLND problem. Since the separation of $\mathbf{x_R}$-cuts is performed by solving a max-flow from r to all nodes in $R\backslash\{r\}$ (instead of a max-flow from r to a single node in $V\backslash\{r\}$), more $\mathbf{x_R}$-cuts are added compared to the number of $\mathbf{x}$-cuts that are added for the RRTLND problem. We observe that fewer $\mathbf{y^0}$-cuts are inserted during the separation than for the RRTLND problem. This can be explained by the size of the primary subtree built in the first stage which is much smaller for the RRTLStT problem.

		Running times statistics ($t \leq 1800s$)					Gap (%) statistics ($t > 1800s$)				
Class	$\|K\|$	#	Min	Median	Mean	Max	#	Min	Median	Mean	Max
G	10	192	2.86	61.07	148.10	1349.00	64	0.10	0.43	1.45	12.08
	20	147	4.91	126.00	308.40	1728.00	109	0.08	1.39	4.73	22.27
	30	118	6.27	136.30	371.60	1750.00	138	0.12	1.51	6.77	27.59
SC	10	238	1.44	46.16	204.40	1708.00	210	0.01	0.08	0.27	3.48
	20	213	3.53	54.72	185.40	1552.00	235	0.02	0.31	1.04	6.58
	30	208	4.85	78.50	224.70	1730.00	240	0.02	1.56	3.01	16.42

TABLE 3.7: Running times and gap statistics of all instances of classes G and SC ($\beta, \epsilon \in \{0.5, 1, 2, 3\}$, $\pi = 10\%$, RRTLStT problem)



(a) All instances of group G for different values of $|K|$ for the RRTLStT problem ($\beta, \epsilon \in \{0.5, 1, 2, 3\}$, $\pi = 10\%$)

(b) Influence of β and ϵ in the algorithm performance for the 250 nodes group of class G ($\pi = 10\%$, $|K| = \{10, 20, 30\}$, RRTLStT problem)

FIGURE 3.5: Cumulative percentage of instances with a given gap (%) obtained within the time limit for the RRTLStT problem

In Figure 3.5(a) we find further insights about the quality of the solutions for the RRTLStT problem for class G and its dependence to $|K|$. The results are analogous to those for the RRTLND problem. The influence of the cost structure, which depends on β and ϵ, on the algorithmic performance is outlined in Figure 3.5(b), where results for the group of instances with 250 nodes are shown.

3.6. Conclusions

RRO is a concept that falls within the framework of 2SRO. In a certain sense, RRO can be viewed as two-stage robust optimization with limited recourse (where the practical recovery action constitutes the limited recourse available to the decision maker). It models the practical contexts where a robust solution is desired, but where it is possible to "recover" the solution appropriately (i.e., make it feasible using the limited set of recourse actions available) once the uncertainty is resolved. While there has been a lot of work on robust optimization, the work on 2SRO and RRO, especially in the discrete optimization context is somewhat limited. Our work contributes to the 2SRO and RRO literature in the context of the TLND problem.

The recoverable robust counterpart of the TLND problem studied in this chapter addresses uncertainty in the set of primary nodes, which we modeled by means of a set of discrete scenarios. We showed that when the input instance corresponds to a tree, the problem remains NP-Hard and we propose a MIP formulation with a linear number of variables for this case. For general networks we developed a MIP formulation based on cut-set inequalities, and we designed specialized techniques to solve the problem exactly within a branch-and-cut framework. A Steiner variant of the problem was also considered and the exact approach was suitably adapted.

Our branch-and-cut approach was tested extensively on two classes of instances for both the RRTLND and RRTLStT problems. As noted in our experiments the cost structure of the problem has a significant effect on both the solution structure and the running times. We evaluated the benefit of RRO against a traditional (one-stage) Robust Optimization approach and a Wait-and-See approach using a concept termed the *Gain of Recovery* [used previously by Büsing et al., 2011]. Our computational results clearly demonstrate the significant benefits of the RRO approach.

Chapter 4

The Recoverable Robust Facility Location Problem

4.1. Introduction

Nowadays, we are more and more aware of the growing presence of dynamism and uncertainty in decision making. Fortunately, as the decisions become more complex, the availability of modeling, algorithmic and computational tools increases as well. Facility location and allocation decisions are among the most relevant decisions in several private and public logistic contexts and they usually involve strategic and operative policies with mid and long term impacts. Precisely because of the practical relevance of these decisions, it is important that they incorporate the uncertainty that naturally appears during the planning, modeling and operative process. Such uncertainty can be represented by different realizations of the input data: *customers* that actually require a commodity or a service, *locations* where the facilities can be located, the *network* that can be used for connecting customers with facilities, and the corresponding *costs*. The true values of this data usually become available later in the decision process. In such cases a standard deterministic optimization model that considers a single possible outcome of the input data can lead towards solutions that are unlikely to be optimal, or for that matter even feasible, for the final data realization.

Supply chain management is a strategical area in which both *uncertainty* and *facility location* are core elements. For instance, as it is pointed out in [Snyder and Daskin, 2005], supply chains are particularly vulnerable to disruptions (intentional or accidental), imposing the need of taking into account the possible availability of depots and roads and different structures of the demand. Likewise, short-term phenomena such as fluctuations in commodity prices (such as oil) or long-term public policies (such as new toll road concessions) might lead to operational cost increases that should be considered when deciding the transportation network to be used.

In another context, natural events such as tsunamis, hurricanes or blizzards can produce disastrous effects with unpredictable intensity on populated areas and on the transportation infrastructure. Countries such as Bangladesh and the Philippines are two typical examples; both of them are regularly hit by hydrological disasters such as floods and typhoons. According to the Department of Disaster Management of Bangladesh [DDM, 2014], every year around 18% of the country is flooded, which produces over 5000 causalities and the destruction of more than 7 millions of homes. However, flooded areas my exceed the 75% of the country in case of severe events (as in 1988, 1998 and 2004). In the case of the Philippines, between 6 to 9 typhoons make landfall every year producing thousands of human losses and incalculable urban destruction; in November of 2013, typhoon Haiyan produced 6241 causalities and material damage of over 809 millions USD [see PAGASA, 2014]. In these examples, it is crucial to be able to count with a robust system of humanitarian relief facilities that even in the worst possible scenario can provide assistance with the quickest possible response reducing the number of human loses after the occurrence of the event.

The Uncapacitated Facility Location Problem (UFL), also referred as the Simple Plant Location Problem, is one of the fundamental models in the wide spectrum of *Facility Location* problems [see, e.g., recent overviews presented in Eiselt and Marianov, 2011, Daskin, 2013]. In the classical deterministic version of the UFL one is given the set of customers, the set of locations, the facility set-up costs and the allocation costs. The goal is to define where to open facilities and how to allocate the customers to them so that the sum of set-up plus allocation costs is minimized.

In practice, it is usually the case that from the moment that the information is gathered until the moment in which the solution has to be implemented, some of the data might change with respect to the initial setting. As mentioned above, even if some (rough) idea about customers and locations is known, changes in demographic, socio-economic, or meteorological factors can lead to changes in the structure of the demand during the planning horizon, and/or the availability of a given location to host a facility (even if a facility has been already installed). This means that the solution obtained using a classical method might become infeasible and a new solution might have to be redefined from scratch. In these cases it would be better to recognize the presence of different scenarios for the data and find a solution comprised by first- and second-stage decisions.

Two well-known approaches to deal with uncertainty in optimization are Two-stage Stochastic Optimization (2SSO) and Robust Optimization (RO). In 2SSO [see Birge and Louveaux, 2011] the solutions are built in two stages. In the first stage, a *partial* collection of decisions is defined which are later on completed (in the second stage), when the true data is revealed. Hence, the objective is to minimize the cost of the first-stage decisions plus the *expected* cost of the recourse (second-stage) decisions. The quality of the solutions provided by this model strongly depends on the accuracy of

the random representation of the parameter values (such as probability distributions) that allow to estimate the second-stage expected cost. Nonetheless, sometimes such accuracy is not available so the use of RO models dealing with *deterministic uncertainty* arises as a suitable alternative [see Kouvelis and Yu, 1997, Bertsimas and Sim, 2004, Ben-Tal et al., 2010]. On the one hand these models do not require assumptions about the distribution of the uncertain input parameters; but on the other hand, they are usually meant for calculating single-stage decisions that are immune (in a certain sense) to all possible realizations of the parameter values.

A novel alternative that combines RO and 2SSO is Two-stage Robust Optimization (2SRO); as in RO, no stochasticity of the parameters is assumed, and as in 2SSO, decisions are taken in two stages. In this case, the cost of the second-stage decision is computed by looking at the worst-case realization of the data. Therefore, the goal of 2SRO is to find a *robust* first-stage solution that minimizes both the first-stage cost plus the worst-case second-stage cost among all possible data outcomes. 2SRO is a rather generic classification of models; for references on different 2SRO settings we refer the reader to [Ben-Tal et al., 2004, Zhao and Zeng, 2012].

One of the possibilities in the 2SRO framework is *Recoverable Robustness* [see Liebchen et al., 2009]. Recalling our practical motivation, assume that the facility location and allocation policy is defined in two stages such that we are required to find a first-stage solution that should be *robust* against the possible realizations (*scenarios*) of the input data in a second stage. This means that the first-stage solution is expected to perform *reasonably well*, in terms of feasibility and/or optimality, for *any* possible realization of the uncertain parameters. An essential element of this approach is the possibility of *recovering* the solution built in the first stage (i.e., to modify the previously defined location-allocation policy in order to render it feasible and/or cheaper) once the uncertainty is resolved in a second stage. The *recovery plan* is comprised by *recovery actions* which are known in advance and whose costs might also depend on the possible scenario. This recovery plan is *limited*, in the sense that the effort needed to recover a solution may be limited algorithmically (in terms of how a solution may be modified) and economically (in terms of the total cost of recovery actions). Therefore, instead of looking for a static solution that is robust against all possible scenarios without allowing any kind of recovery [which is the case for many RO approaches, see Ben-Tal et al., 2010], we want a solution robust enough so that it can be *recovered* promptly and at low cost once the uncertainty is resolved. This balance between robustness and recoverability is what defines a *recoverable robust optimization* problem.

With respect to the UFL, we want to find a solution whose first-stage component (opening of some facilities and allocating some customers) is implemented before the complete realization of the data. This solution can then be *recovered* in the second stage (to turn it into a feasible one) once the actual sets of customers and locations

become available. In this case the *recovery actions* correspond to the opening of new facilities, the establishment of new allocations and the *re*-allocation of customers.

The *Recoverable Robust UFL* (RRUFL) looks for a solution that minimizes the sum of the first-stage costs plus the second-stage *robust* recovery cost defined as the the worst case recovery cost over all possible scenarios. A formal definition of the RRUFL is given in §4.2.1.

4.1.1 Our Contribution and Outline of the Paper

The contributions of this work can be summarized as follows: (i) Due to the nature of the considered uncertainty, we use a recent concept of recoverable robust optimization to formulate a Mixed Integer Programming (MIP) model that allows to derive a facility location and allocation policy composed by first- and second-stage decisions; (ii) for this novel problem we design a sophisticated algorithmic framework based on Benders decomposition which is complemented by several non-trivial enhancements; (iii) using instances from two different large classes (representing transportation and disaster management settings) we analyze in detail the characteristics of the proposed model and the obtained solutions as well as the effectiveness, behavior, and limitations of the designed approach.

In §4.2 the concept of Recoverable Robustness is presented and the RRUFL is formally defined. The proposed algorithmic framework is described in §4.3. The description of the benchmark instances and a detailed analysis of the computational results are presented in §4.4. Finally, conclusions and final remarks are given in §4.5.

4.1.2 The Uncapacitated Facility Location Problem

It is hard to establish a single seminal work presenting the UFL, nonetheless [Kuehn and Hamburger, 1963] is usually regarded as the earliest work where the UFL is presented as commonly known today. We refer the reader to [Cornuejols et al., 1990, Verter, 2011] (including the references therein) for comprehensive surveys on the UFL and some of its variants.

A MIP formulation for the UFL can be given as follows. Let R be the set of customers, J the set of potential location of facilities, and A a set of links (i, j) connecting customers i in R with locations j in J ($A \subseteq R \times J$). The cost of opening a facility at location $j \in J$ is given by f_j, and the cost of assigning customer $i \in R$ to facility $j \in J$ using an existing link (i, j) is given by c_{ij}. Let $\mathbf{y} \in \{0,1\}^{|J|}$ be a vector of binary variables such that $y_j = 1$ if a facility is opened at location $j \in J$ and $y_j = 0$ otherwise, and let $\mathbf{x} \in \{0,1\}^{|A|}$ be a vector of binary variables such that $x_{ij} = 1$ if customer $i \in R$

is allocated to a facility in $j \in J$ using link $(i,j) \in A$. Using this notation, the UFL can be formulated as follows:

$$
\begin{aligned}
OPT = \min & \sum_{j \in J} f_j y_j + \sum_{(i,j) \in A} c_{ij} x_{ij} & \\
\text{s.t.} \quad & \sum_{(i,j) \in A} x_{ij} = 1, & \forall i \in R \\
& x_{ij} \leq y_j, & \forall (i,j) \in A,\ \forall j \in J \\
& \mathbf{y} \in \{0,1\}^{|J|} \text{ and } \mathbf{x} \in [0,1]^{|A|}. &
\end{aligned}
$$

Despite its simple definition, the UFL is known to be NP-Hard [Cornuejols et al., 1990]; however, the current advances in MIP solvers, computing machinery and the development of sophisticated preprocessing techniques allow to find optimal or nearly optimal solutions for large instances of the UFL within reasonable time. We refer to [Letchford and Miller, 2012] for recent works on reduction procedures for the UFL.

The incorporation of different types of uncertainty when modeling and solving the UFL is not new; in §4.2.2 we will provide a brief review of Facility Location under uncertainty and compare our setting with previously proposed problems.

4.2. The Recoverable Robust UFL

In this section we present a literature review on Recoverable Robustness and formally define the RRUFL.

Recoverable Robust Optimization Recoverable Robust Optimization (RRO) was first introduced in [Liebchen et al., 2007, 2009] as a tool for decision making under uncertainty in applications arising in the railway scheduling. In [Cacchiani et al., 2008] and [D'Angelo et al., 2011] one can find further applications of RRO in the context of railway scheduling.

In the last couple of years, RRO has been applied to other problems. The recoverable robust knapsack problem considering different models of uncertainty is studied in [Büsing et al., 2011]. Formulations and algorithms for different variants of the recoverable robust shortest path problem are given in [Büsing, 2012]. Models, properties and exact algorithms for recoverable robust two-level network design problems are presented in [Álvarez-Miranda et al., 2013c]. A more general framework of the RRO is studied in [Cicerone et al., 2012] where multiple recovery stages are allowed. The authors apply this new model to timetabling and delay management applications.

Different types of uncertainty, e.g., interval, polyhedral and discrete sets, can be included in the decision process trough RRO. In this paper, we use discrete sets to model the uncertainty.

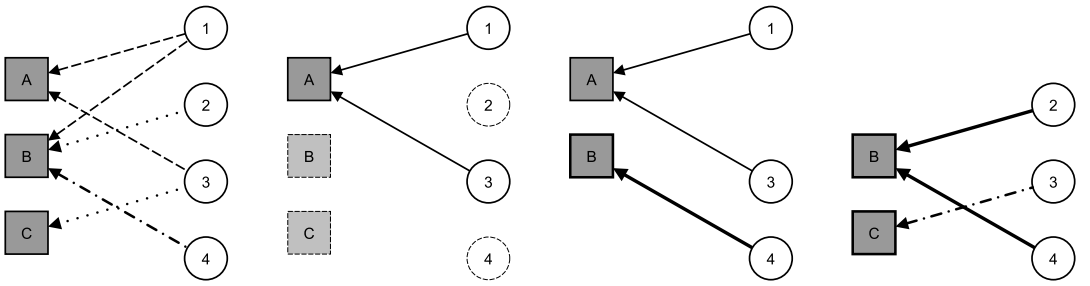
4.2.1 A Formulation of the RRUFL

As mentioned above, facility location along with the corresponding allocation decisions are typically exposed to uncertainty in different input data. As described in [Shen et al., 2011], it is possible to classify uncertainty in three categories: *provider-side* uncertainty, *receiver-side* uncertainty, and *in-between* uncertainty. The first corresponds to the uncertainty in facility capacity, facility reliability, facility availability, etc.; the second is related to the uncertain structure of the set of customers, customer demands, customer locations, etc.; and the third refers to the lack of complete knowledge about the transportation network topology, transportation times or costs between facilities and customers. The proposed recoverable robust UFL model is a general approach and it can address situations in which uncertainty may be present in any of these three categories.

Suppose we are given an instance of the UFL in which uncertainty is present in the set of customers R, the set of locations J, the set of allocation links A and the corresponding set-up and allocation costs. Such application might arise, for instance, in the event of natural disasters. In these cases it can be very hard to estimate in advance (i) which areas will require humanitarian relief, (ii) where the emergency facilities (e.g., Red Cross facilities) can be located and (iii) how the damaged areas can be reached by the emergency brigades coming from the installed facilities. Therefore, instead of dealing with deterministic sets R, J and A we are given a finite set K of discrete scenarios such that each scenario $k \in K$ is characterized by its own sets R^k, J^k and A^k and also by the corresponding set-up and allocation costs.

Formally, let K be a set of scenarios representing possible realizations of the problem data, more precisely, for a given $k \in K$: let R^k be the set of customers that require the service if scenario k is realized; let J^k be the set of locations where facilities can be opened if scenario k is realized; and let A^k be the set of links that can be used if scenario k is realized. We define $R^0 = \bigcup_{k \in K} R^k$ as the set of *potential* customers, $J^0 = \bigcup_{k \in K} J^k$ as the set of *potential* locations and $A^0 = \bigcup_{k \in K} A^k$ as the set of *potential* connections. We assume that the classical UFL has at least one feasible solution for R^0, J^0 and A^0, and that each customer $i \in R^k$ can be reached by some link from A^k.

The decision maker faces a two-stage decision: she/he needs to define a first-stage plan (to open *some* facilities and to allocate *some* customers to these open facilities) without knowing in advance the actual data that will be revealed. Once the actual information is available in a second stage (i.e., a single $k \in K$ and its corresponding R^k, J^k and A^k) additional decisions can be taken in order to *recover* the first-stage plan and turn it into a feasible solution for the revealed data. A second-stage decision is said to be feasible if for all $k \in K$ each customer $i \in R^k$ is allocated to one installed facility in $j \in J^k$ and the allocation link is operational, i.e., $(i, j) \in A^k$. These second-stage decisions consist of (i) the opening of new facilities, (ii) the allocation of customers to



(a) RRUFL instance (b) A 1st-stage solution (c) Solution for $k = 1$ (d) Solution for $k = 2$

FIGURE 4.1: Example of an instance and first- and second-stage solutions for the RRUFL.

facilities that are either opened in the second-stage or were opened in the first-stage, and (iii) the *re*-allocation of customers that were allocated in the first-stage to facilities that are actually not available in the realized scenario.

In Figure 4.1(a) an instance of the RRUFL with set of facilities $J^0 = \{A, B, C\}$, set of customers $R^0 = \{1, 2, 3, 4\}$ and with two scenarios is shown. Scenario $k = 1$ is given by $R^1 = \{1, 3, 4\}$, $J^1 = \{A, B\}$, $A^1 = \{(1, A), (1, b), (3, A), (4, B)\}$, and scenario $k = 2$ is given by $R^2 = \{2, 3, 4\}$, $J^2 = \{B, C\}$, $A^2 = \{(2, B), (4, B), (3, C)\}$. In the first stage, allocation and facility set-up costs are 1 and 2, respectively. In the second stage, allocation and set-up costs are 1.5 and 3, respectively, the cost of *re*-allocating a customer is 2 and the penalty for a facility opened at a non-available site is 3.5. A first-stage solution is shown in Figure 4.1(b); a facility at site A is opened, customers 1 and 3 are allocated to it and the total cost is: 2 (one opening) + 1 + 1 (two allocations) = 4. For this given first-stage decision, we present in Figure 4.1(c) the optimal second-stage solution in case scenario $k = 1$ is realized: a facility at site B has to be installed while the facility at A remains open, customers 1 and 3 keep their allocations while customer 4 is allocated to the facility in B; so the second-stage cost is: 3 (one opening) + 1.5 (one allocation) = 4.5. The optimal second-stage solution in case scenario $k = 2$ is realized is shown in Figure 4.1(d): facilities at B and C have to be installed while the facility at A becomes unavailable, customers 2 and 4 are allocated to the facility at B, while customer 3 has to be *re*-allocated to the facility in C; the cost is: $3 + 3$ (two opening) + 1.5 + 1.5 (two allocations) + 2 (one *re*-allocation) + 3.5 (one penalty) = 14.5. Therefore, in the *worst* case, the overall cost of establishing this first-stage solution and recover it in the second stage is given as $\max\{4 + 4.5, 4 + 14.5\} = 18.5$. Our goal will be to find the optimal first-stage decision, so that in the worst-case total cost of the first- and second-stage is minimized. For this example, the optimal first-stage solution is defined by the installation of a facility in B and the allocation of 4 to it; this solutions induces a first-stage cost of 3 and worst case second stage cost of 6, yielding a total cost of 9.

MIP Formulation In the first stage, decisions are modeled as follows: let $\mathbf{y}^0 \in \{0, 1\}^{|J^0|}$ be a vector of binary variables such that $y_j^0 = 1$ if a facility is opened at

location $j \in J^0$ in the first stage (at cost f_j^0) and $y_j^0 = 0$ otherwise; let $\mathbf{x}^0 \in \{0,1\}^{|A^0|}$ be a vector of binary variables such that $x_{ij}^0 = 1$ if the link $(i,j) \in A^0$ is used to allocate customer $i \in R^0$ to the facility at $j \in J^0$ (at cost c_{ij}^0) and $x_{ij}^0 = 0$ otherwise. For a given scenario $k \in K$, second-stage decisions are defined as follows: let $\mathbf{y}^k \in \{0,1\}^{|J^k|}$ be a vector of binary variables such that $y_j^k = 1$ if a facility is opened at location $j \in J^k$ in the second stage (at cost f_j^k) and $y_j^k = 0$ otherwise; let $\mathbf{x}^k \in \{0,1\}^{|A^k|}$ be a vector of binary variables such that $x_{ij}^k = 1$ if the link $(i,j) \in A^k$ is used to allocate customer $i \in R^k$ to the facility at $j \in J^k$ (at cost c_{ij}^k) and $x_{ij}^k = 0$ otherwise; and let $\mathbf{z}^k \in \{0,1\}^{|A^k|}$ be a vector of binary variables such that $z_{il}^k = 1$ if the link $(i,l) \in A^k$ is used to *re*-allocate customer $i \in R^k$ to the facility at $l \in J^k$ (at cost r_{il}^k) and $z_{il}^k = 0$ otherwise. If a facility is installed in the first stage at a given location $j \in J^0$ ($y_j^0 = 1$) and this location is available if scenario k is realized in a second stage ($j \in J^k$), then this facility remains open and no extra cost is incurred; if the location is not available in the second stage ($j \in J^0 \setminus J^k$), then a penalty p_j^k must be paid.

With this definition of variables, a first-stage solution is a pair $(\mathbf{x}^0, \mathbf{y}^0) \in \{0,1\}^{|A^0|+|J^0|}$ satisfying

$$x_{ij}^0 \leq y_j^0, \quad \forall (i,j) \in A^0 \tag{FS.1}$$

$$\sum_{(i,j)\in A^0} x_{ij}^0 \leq 1, \quad \forall i \in R^0. \tag{FS.2}$$

Given a first-stage solution $(\mathbf{x}^0, \mathbf{y}^0)$ and a scenario $k \in K$, the *recovery cost* is the minimum total cost $\rho(\mathbf{x}^0, \mathbf{y}^0, k)$ of the second-stage recovery actions $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k)$ needed to render the solution feasible. Hence, $\rho(\mathbf{x}^0, \mathbf{y}^0, k)$ is found by solving the following *recovery problem*:

$$\rho\left(\mathbf{y^0}, \mathbf{x^0}, k\right) = \min \sum_{j\in J^k} f_j^k \left(y_j^k - y_j^0\right) + \sum_{(i,j)\in A^k} c_{ij}^k x_{ij}^k + \sum_{(i,l)\in A^k} r_{il}^k z_{il}^k + \sum_{j\in J^0\setminus J^k} p_j^k y_j^0 \tag{R.1}$$

$$\text{s.t.} \quad \sum_{(i,j)\in A^0} x_{ij}^0 + \sum_{(i,j)\in A^k} x_{ij}^k = 1, \quad \forall i \in R^k \tag{R.2}$$

$$\sum_{(i,j)\in A^0\setminus A^k} x_{ij}^0 \leq \sum_{(i,l)\in A^k} z_{il}^k, \quad \forall i \in R^k \tag{R.3}$$

$$x_{ij}^k + z_{ij}^k \leq y_j^k, \quad \forall (i,j) \in A^k, \ \forall i \in R^k \tag{R.4}$$

$$y_j^0 \leq y_j^k, \quad \forall j \in J^k \tag{R.5}$$

$$\mathbf{y^k} \in \{0,1\}^{|J^k|}, \ \mathbf{x^k} \in \{0,1\}^{|A^k|}, \ \mathbf{z^k} \in \{0,1\}^{|A^k|}. \tag{R.6}$$

Objective function (R.1) is comprised by the set-up cost of facilities in the second-stage ($\sum_{j\in J^k} f_j^k(y_j^k - y_j^0)$), the allocation cost in the second-stage ($\sum_{(i,j)\in A^k} c_{ij}^k x_{ij}^k$), the cost of re-allocating customers ($\sum_{(i,l)\in A^k} r_{il}^k z_{il}^k$), and the total penalty paid by those facilities opened in the first stage that can not operate if scenario $k \in K$ is realized ($\sum_{j\in J^0\setminus J^k} p_j^k y_j^0$). Constraints (R.2) state that a customer is either allocated in the first stage ($\sum_{(i,j)\in A^0} x_{ij}^0$) or in the second-stage ($\sum_{(i,j)\in A^k} x_{ij}^k$). Constraints (R.3) model the fact that if a customer $i \in R^k$ has been allocated in the first-stage to a facility $j \in J^0$ by

means of a link $(i,j) \in A^0 \backslash A^k$ then it has to be re-allocated to another facility $l \in J^k$ through a link (i,l) available in the second-stage ($\sum_{(i,l)\in A^k} z^k_{il}$). Constraints (R.4) impose that if a customer is allocated or re-allocated to a facility $j \in J^k$, then that facility has to be available and reachable in the second-stage. The fact that a facility that has been opened in the first stage should remain opened in the second stage is modeled by (R.5). The nature of the variables is imposed in (R.6) (note that one can also relax the integrality constraints for $\mathbf{x}^k$ and $\mathbf{z}^k$, $\forall k \in K$).

For a given first-stage solution $(\mathbf{x}^0, \mathbf{y}^0)$ the *robust recovery cost* $R(\mathbf{x}^0, \mathbf{y}^0)$ corresponds to the maximum *recovery cost* among all $k \in K$, i.e.,

$$R\left(\mathbf{x}^0, \mathbf{y}^0\right) = \max_{k\in K} \rho\left(\mathbf{x^0}, \mathbf{y^0}, k\right). \tag{RR}$$

Combining (FS.1)-(FS.2), (R.1)-(R.6) and (RR), we define the Recoverable Robust UFL problem (RRUFL) as

$$OPT_{\mathcal{R}R} = \min \sum_{j\in J^0} f^0_j y^0_j + \sum_{(i,j)\in A^0} c^0_{ij} x^0_{ij} + R\left(\mathbf{x}^0, \mathbf{y}^0\right) \tag{4.1}$$

$$\text{s.t. (FS.1)-(FS.2), (R.2)-(R.6) and } (\mathbf{x}^0, \mathbf{y}^0) \in \{0,1\}^{|A^0|+|J^0|}. \tag{4.2}$$

In the proposed formulation of the RRUFL we impose that each customer $i \in R^k$ has to be assigned (or re-assigned) to exactly one available facility $j \in J^k$ for any given $k \in K$. It is possible to relax this and, instead, impose a penalty, say t^k_i, if customer $i \in R^k$ is not served by any facility if scenario k is realized. This can be done by introducing a dummy facility π^k with a set-up cost equal to 0 and connecting it to every customer $i \in R^k$ with an allocation (and re-allocation) cost $c^k_{i\pi} = r^k_{i\pi} = t^k_i$.

In many applications it is natural to think that whichever decision we take in the future it will be more expensive than if it would have been taken at present. For instance, opening a facility at a given location is likely to be more expensive later on in the planning horizon than now ($f^k_j \geq f^0_j$). Likewise, an agreement between a depot (facility) and a customer is expected to have better conditions (for one of the two parties at least) if it is established earlier than if it is defined when the market conditions have evolved ($c^k_{ij} \geq c^0_{ij}$). Furthermore, it is also natural to think that if an already agreed pact between a depot and a customer is forced to be changed (e.g., because no allocation link is available between them), this will entail an additional re-allocation cost possibly higher than the original one ($r^k_{il} \geq c^0_{ij}$, for all $l \in J^k$).

An optimal first-stage solution $(\mathbf{x}^0, \mathbf{y}^0)$ is *robust* because, regardless which scenario occurs, it guarantees that the second-stage actions will be efficient (due to the minimization of the worst case) and easy to implement (because only a simple UFL has to be solved). Hence, the more scenarios we take into consideration to find $(\mathbf{x}^0, \mathbf{y}^0)$, the more robust the solution is; because we are foreseeing more possible states of the

future uncertainty. Unlike common approaches of RO that protect solutions against perturbations in parameters as costs or demands, our approach also hedges against uncertainty in the very topology of the network. Likewise, a first-stage solution is *recoverable*, or possesses *recoverability*, because it can become feasible in a second stage by means of second-stage actions.

The Robust UFL without Recovery To assess the effectiveness and benefits of the RRULF, we also introduce another natural, but more conservative, model. Assume a decision-making context equivalent to the one taken into account before. Consider a model in which first-stage decisions are comprised *only* by $\mathbf{y}^0$ and second-stage decisions *only* by $\mathbf{x}^k$, $\forall k \in K$. This is, an 2SRO model in which facilities can be opened only in the first stage and allocations can be decided only in the second stage. We will refer to this new problem simply as Robust Uncapacitated Facility Location without Recovery (RUFL). This alternative model lacks the concept of *recoverability* since the solution cannot be intrinsically changed: no new facility can be opened and there is no need to re-allocate any customer in the second stage. Therefore, the solutions of such model although possibly *more* robust (since they are more conservative) are expected to be more expensive, either because unnecessarily many facilities have to be opened in the first stage or because the second-stage allocation costs are considerably higher than those of the first stage. If we consider again the instance in Figure 4.1(a), one can easily see that for this new model the optimal (and only feasible) first-stage solution would be given by the installation of facilities in A, B and C (with a cost of 6). In both $k = 1$ and $k = 2$ the optimal second-stage cost would be 8. This leads to a total cost equal to $6 + \max\{8, 8\} = 14$, which is more than the cost of the optimal solution of the RRUFL which is 9.

4.2.2 The RRUFL and Previously Proposed Problems

Already in the 70's efforts were devoted to provide both theoretical and algorithmic contributions on Stochastic UFL. In [Snyder, 2006] one can find an excellent review on Facility Location under uncertainty, describing contributions not only from the stochastic but also from the RO perspective. More recent references to Facility Location under uncertainty include [Snyder and Daskin, 2005, Averbakh, 2005, Snyder and Daskin, 2006, Cui et al., 2010, Shen et al., 2011, Albareda-Sambola et al., 2011, Adjiashvili, 2012, Gao, 2012, Alumur et al., 2012, Albareda-Sambola et al., 2013, Gïlpinar et al., 2013] and [Li et al., 2013].

Our definition of the RRUFL, as well as the algorithmic framework described later, spans different possible cases of uncertainty in Facility Location. Some of them have been already addressed in the literature by the use of stochastic and robust two-stage models.

For instance if $J^k = J^0$ and $A^k = A^0$, $\forall k \in K$, then we are only addressing uncertainty in the set of customers and, eventually, in the second-stage costs. A 2SSO approach for this problem has been considered in [Ravi and Sinha, 2006], where approximation algorithms have been proposed. In [Snyder and Daskin, 2005, Cui et al., 2010, Shen et al., 2011] and [Li et al., 2013], uncertainty has been addressed only in the set of locations ($R^k = R^0$ and $A^k = A^0$, $\forall k \in K$). As stressed by the authors, this model is suitable for applications where facilities might become *unavailable* in a second stage due to disruptions caused by natural disasters, terrorists attacks or labor strikes [see Cui et al., 2010]. These papers share two important features. First, uncertainty is tackled by means of 2SSO since probabilities of facility failure are known in advance for each scenario. Second, a user is assigned to a so-called *primary* facility that will serve it under normal circumstances, as well as to a set of *ordered backup* facilities such that the first of them that is available will serve the customer when the primary is not available [see Snyder and Daskin, 2005]. This second feature cannot be included in our framework without introducing additional binary variables; nonetheless decision-maker preferences about the *re*-allocation of a customer in case the originally assigned facility fails can be incorporated by a proper definition of the re-allocation second-stage costs.

A third case is the one where only connections are subject to uncertainty ($R^k = R^0$ and $J^k = J^0$, $\forall k \in K$). A 2SRO model of this case is studied in [Hassin et al., 2009] where the relevance of such a model of uncertainty is emphasized in the context of response planning after disasters.

4.3. Algorithmic Framework

Note that formulation (4.1)-(4.2) has a polinomial number of variables and constraints with respect to $|R^0|$, $|A^0|$ and $|K|$. Therefore it can be solved directly (as a compact model) through any state-of-the-art MIP solver (e.g., CPLEX). However, as we will show later, when large realistic instances have to be solved, the direct use of solvers turns out to be impractical.

Model (4.1)-(4.2) is a natural candidate to be solved by means of a Benders-like decomposition approach: the first-stage variables $(\mathbf{x}^0, \mathbf{y}^0)$ are incorporated in the master problem (MP) and the second-stage variables $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{z}^k)$ are projected out and replaced by a single variable ω representing the robust recovery cost, for a given $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$, that is computed by solving $|K|$ slave problems (SPs). Thus, the objective function (4.1) becomes $OPT_{\mathcal{R}R} = \min \sum_{j \in J^0} f_j^0 y_j^0 + \sum_{(i,j) \in A^0} c_{ij}^0 x_{ij}^0 + \omega$, where $\omega \geq \rho\left(\mathbf{x}^0, \mathbf{y}^0, k\right), \forall k \in K$. Hence, for each given value of $\left(\mathbf{x}^0, \mathbf{y}^0, k\right)$, ω can be computed by independently solving $|K|$ problems (R.1)-(R.6).

One of the main drawbacks of traditional implementations of Benders decomposition for two-stage integer problems is the need for solving several MIP problems (MP and

SPs) at each iteration in order to obtain a single Benders-cut. Nonetheless, nowadays most of MIP optimization suites provide branch-and-cut frameworks supported by the use of callbacks. Therefore, a Benders decomposition algorithm can be transformed into a pure *branch-and-cut* approach by the use of callbacks. Benders cuts are added to the model as valid lower-bounds on ω each time a potential solution of the MP is found by means of solving a Linear Programming (LP) problem in a given node of the enumeration tree. This technique exploits the benefits of the decomposition allowing to implement additional methods for heuristically finding more cuts and/or for strengthening the obtained ones. That way, both, the speed and the convergence of the algorithm can be improved [see Ljubić et al., 2013, Pérez-Galarce et al., 2014].

Basic Separation of *L*-shaped and *Integer L*-shaped Cuts In our approach, a valid lower bound on ω is iteratively imposed by means of *L*-shaped and *integer L*-shaped cuts [see Van Slyke and Wets, 1967, Laporte and Louveaux, 1993]. For a given first-stage solution, the second-stage problem can be decomposed into $|K|$ independent problems: dual variables of the LP-relaxations of these SPs yield *L*-shaped cuts that are added to the MP while integer solutions of the SPs yield *integer L*-shaped cuts.

At a given node of the enumeration tree, let $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ be a first-stage solution satisfying (FS.1)-(FS.2) and let $\tilde{\omega}$ be the current value of variable ω. For a given $k \in K$, the dual of (R.1)-(R.6) after removing the integrality constrains can be formulated as

$$\max \sum_{i \in R^k} \left[\alpha_i \left(1 - \sum_{(i,j) \in A^0} \tilde{x}^0_{ij} \right) + \gamma_i \left(\sum_{(i,j) \in A^0 \setminus A^k} \tilde{x}^0_{ij} \right) \right] + \sum_{j \in J^k} \left(\epsilon_j - f^k_j \right) \tilde{y}^0_j + \sum_{j \in J^0 \setminus J^k} p^k_j \tilde{y}^0_j \tag{D.1}$$

$$\text{s.t. } \alpha_i - \delta_{ij} \leq c^k_{ij}, \quad \forall (i,j) \in A^k, \ \forall i \in R^k \tag{D.2}$$

$$\gamma_i - \delta_{il} \leq r^k_{il}, \quad \forall (i,l) \in A^k, \ \forall i \in R^k \tag{D.3}$$

$$\epsilon_j + \sum_{(i,j) \in A^k} \delta_{ij} \leq f^k_j, \quad \forall j \in J^k \tag{D.4}$$

$$(\boldsymbol{\alpha},\ \boldsymbol{\gamma},\ \boldsymbol{\delta},\ \boldsymbol{\epsilon}) \geq \mathbf{0}, \tag{D.5}$$

where $(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\epsilon})$ correspond to the dual variables of constraints (R.2), (R.3), (R.4) and (R.5), respectively. Let $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\delta}}, \tilde{\boldsymbol{\epsilon}})$ be an optimal solution to (D.1)-(D.5) with optimal value $\tilde{\rho}^k$. Following the LP-duality theory, an *L*-shaped (*optimality*) cut is given by

$$\omega \geq \sum_{i \in R^k} \left[\tilde{\alpha}_i \left(1 - \sum_{(i,j) \in A^0} x^0_{ij} \right) + \tilde{\gamma}_i \left(\sum_{(i,j) \in A^0 \setminus A^k} x^0_{ij} \right) \right] + \sum_{j \in J^k} \left(\tilde{\epsilon}_j - f^k_j \right) y^0_j + \sum_{j \in J^0 \setminus J^k} p^k_j y^0_j, \tag{LS}$$

which is added to the model if $\tilde{\omega} < \tilde{\rho}^k$. Note that an *L*-shaped cut (LS) can be found regardless of $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ being integer.

Now suppose that $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ is integer. If there is no $k \in K$ with $\tilde{\omega} < \tilde{\rho}^k$, then one can attempt to find *integer L*-shaped cuts [see Laporte and Louveaux, 1993]. For a

given $k \in K$, let $\breve{\rho}^k$ be the optimal value of (R.1)-(R.6) (preserving the integrality constraints), if $\tilde{\omega} < \breve{\rho}^k$, then the following valid inequality can be added to the MP,

$$\omega \geq \breve{\rho}^k \left(\sum_{(i,j)\in\mathcal{A}^k} (x^0_{ij} - 1) - \sum_{(i,j)\in A^k\setminus\mathcal{A}^k} x^0_{ij} + \sum_{j\in\mathcal{J}^k} (y^0_j - 1) - \sum_{j\in J^k\setminus\mathcal{J}^k} y^0_j + 1 \right), \quad (i\text{-LS})$$

where $\mathcal{A}^k = \{(i,j) \in A^k \mid \tilde{x}^0_{ij} = 1\}$ and $\mathcal{J}^k = \{j \in J^k \mid \tilde{y}^0_j = 1\}$ are the index sets of the links $(i,j) \in A^k$ and locations $j \in J^k$ *chosen* in the first stage, respectively.

4.3.1 Strengthening and Calculating Additional *L*-shaped Cuts

In the following we will describe the different enhancements that we have incorporated into our algorithmic framework.

Scenario Sorting Formally speaking, when separating (LS) cuts we only need to add the cut associated with the worst-case scenario $k^* = \arg\max_{k\in K}(\tilde{\rho}^k)$ for a given $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$. However this entails an important disadvantage: exactly $|K|$ LP and/or ILP problems have to be solved to optimality, and only a single cut is generated out of this eventually large computational effort.

In order to overcome the above described drawback we have designed a strategy that first *sorts* scenarios dynamically according to the information of previous iterations and then attempts to add not a single but many potentially *good* cuts. We first note that as long as $\tilde{\omega} < \tilde{\rho}^k$, one can add an (LS) cut. Secondly, it is intuitive to think that for a given instance there is a subset of scenarios that systematically induce violated cuts, while another subset of scenarios rarely do so. Therefore, on the basis of the *cut violation* values, $\tilde{\rho}^k - \tilde{\omega}$, one can dynamically update a list $\overline{K} = [\overline{k}_1, \overline{k}_2, \ldots, \overline{k}_K]$, placing in the first positions those scenarios that consistently induce large cut violation and at the end those that rarely satisfy $\tilde{\omega} < \tilde{\rho}^k$.

In our strategy we apply *learning mechanisms* to identify $\overline{K}$ and prioritize the search of violated *L*-shaped cuts using the first elements of the list until a pre-fixed number $MAX_{\text{cut}} \leq |K|$ of violated cuts has been found or a pre-fixed number $MAX_{\text{fail}} \leq |K|$ of failed attempts has been reached.

In Algorithm 4 we present the general scheme of the separation of *L*-shaped cuts using the scenario sorting strategy. For each scenario $k \in K$, the value $freq[k]$ accumulates the number of separation calls in which we have solved the corresponding SP. Likewise, the value $viol[k]$ is a cumulative cut violation value of scenario k, over all previous separation calls. In Step 1 the list $\overline{K}$ is created and its elements are sorted in decreasing order with respect to $viol[k]/freq[k]$, which represents the *average* violation that each scenario has induced in the previous iterations. In loop 3-12 the *L*-shaped cuts are added: in line 4 the first scenario in the list $\overline{K}$ is taken and removed; the $k-$th SP

Algorithm 4 Basic L-shaped cut Separation with *Scenario sorting*

Input: Fractional solution $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0, \tilde{\omega})$; vectors **freq** and **viol**; MAX_{cut} and MAX_{fail}.

1: $\overline{K} = \texttt{sortScenarios}(K, \textbf{viol}, \textbf{freq})$;
2: Set $\text{c}_{\text{cut}} = 0$ and $\text{c}_{\text{fail}} = 0$;
3: **repeat**
4: $k = \texttt{getFirst}(\overline{K})$;
5: Solve the LP-relaxation of the $k-$th SP (R.1)-(R.6) and let $\tilde{\rho}^k$ be the corresponding optimal value;
6: $freq[k] = freq[k] + 1$ and $viol[k] = viol[k] + (\tilde{\rho}^k - \tilde{\omega})$;
7: **if** $\tilde{\omega} < \tilde{\rho}^k$ **then**
8: Insert an L-shaped cut given by (LS) into the LP;
9: c_{cut}++;
10: **else**
11: c_{fail}++;
12: **until** $\text{c}_{\text{cut}} = MAX_{\text{cut}}$ or $\text{c}_{\text{fail}} = MAX_{\text{fail}}$
13: Resolve the LP;

is solved in line 5; both vectors needed to sort scenarios are updated in line 6; if the solution of the SP induces a violated cut (line 7) then the corresponding inequality is added in line 8 and the counter of added cuts is increased (line 9); if no violated cut is generated, the corresponding counter is increased in line 11.

In our default implementation (and after parameter tuning), we have set $MAX_{\text{cut}} = 0.25 \times |K|$ and $MAX_{\text{fail}} = 0.25 \times |K|$.

Dual Lifting Clearly, the strength of the generated L-shaped cuts will strongly influence the performance of the algorithm; the stronger they are, the less MP iterations (hence, the less explored nodes in the enumeration tree) are needed. In this paper we use a recently proposed technique to strengthen L-shaped cuts [see Ljubić et al., 2013]. In contrast to other approaches for generating stronger cuts [see, e.g., Magnanti and Wong, 1981], this method does not require to solve any additional LP problem and the strengthening process can be performed in linear time (with respect to the number of variables).

Let $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ be a pair satisfying (FS.1)-(FS.2), $\tilde{\omega}$ the current value of variable ω, and $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}}, \tilde{\boldsymbol{\delta}}, \tilde{\boldsymbol{\epsilon}})$ an optimal solution to (D.1)-(D.5) that satisfies $\tilde{\omega} < \tilde{\rho}^k$. The scheme to strengthen the corresponding L-shaped cut is the following: (i) If a for customer $i \in R^k$ we have $\sum_{(i,j)\in A^0} \tilde{x}^0_{ij} = 1$, then the corresponding dual variable α_i does not appear in (D.1). (ii) If a for customer $i \in R^k$ we have $\sum_{(i,j)\in A^0 \setminus A^k} \tilde{x}^0_{ij} = 0$, then the corresponding dual variable γ_i does not appear in (D.1). (iii) If for a facility $j \in J^k$ we have $\tilde{y}^0_j = 0$, then the corresponding dual variable ϵ_j does not appear in (D.1). (iv) Moreover, variables $\boldsymbol{\delta}$ do not appear in the objective (D.1) neither. On the basis of (i)-(iv) we observe that we deal with a highly degenerate LP and one can expect that the optimal solutions to (D.2)-(D.4) usually produce positive slacks (typically, an LP solver will fix the associated dual variables to zero). The idea is now to produce another LP optimal solution of the dual SP such that these slacks are reduced to zero.

Therefore, the values of the dual coefficients in (LS) will be lifted as follows:

$$\check{\alpha}_i = \begin{cases} \tilde{\alpha}_i & \text{if } \sum_{(i,j)\in A^0} \tilde{x}^0_{ij} < 1 \\ \min_{(i,j)\in A^k}\{c^k_{ij} + \tilde{\delta}_{ij}\} & \text{otherwise} \end{cases}$$

$$\check{\gamma}_j = \begin{cases} \tilde{\gamma}_j & \text{if } \sum_{(i,j)\in A^0\setminus A^k} \tilde{x}^0_{ij} > 0 \\ \min_{(i,j)\in A^k}\{r^k_{ij} + \tilde{\delta}_{ij}\} & \text{otherwise} \end{cases}$$

$$\check{\epsilon}_j = \begin{cases} \tilde{\epsilon}_j & \text{if } \tilde{y}^0_j > 0 \\ f^k_j - \sum_{(i,j)\in A^k} \tilde{\delta}_{ij} & \text{otherwise} \end{cases}.$$

This is why we refer to this procedure as *dual lifting*. If $\check{\alpha}_i > \tilde{\alpha}_i$, $\check{\gamma}_j > \tilde{\gamma}_j$ or $\check{\epsilon}_j > \tilde{\epsilon}_j$ for at least one $i \in R^k$ or $j \in J^k$, respectively, then the *lifted* L-shaped cut is given by

$$\omega \geq \sum_{i\in R^k} \left[\check{\alpha}_i \left(1 - \sum_{(i,j)\in A^0} x^0_{ij} \right) + \check{\gamma}_i \left(\sum_{(i,j)\in A^0\setminus A^k} x^0_{ij} \right) \right] + \sum_{j\in J^k} \left(\check{\epsilon}_j - f^k_j\right) y^0_j + \sum_{j\in J^0\setminus J^k} p^k_j y^0_j. \tag{l-LS}$$

Lemma 4.1 (Ljubić et al. [2013])**.** *The* lifted L*-shaped cuts* (l-LS) *are valid and strictly stronger than the standard* L*-shaped cuts* (LS)*.*

From the algorithmic point of view, to apply this approach one simply has to insert a cut of type (l-LS) instead of one of type (LS) in line 8 of Algorithm 4.

Zero-half-*L*-shaped Cuts Zero-half cuts are a subclass of rank-1 Chvátal-Gomory cuts with multipliers restricted to $\{0, \frac{1}{2}\}$ [Caprara and Fischetti, 1996]. They play an important role in polyhedral theory, and nowadays they are also incorporated in major MIP solvers. Instead of using a generic zero-half cut generation [see, e.g., Andreello et al., 2007], we impose zero-half cuts in combination with the learning mechanisms introduced in the previous section. To this end, for a given $k \in K$, observe that by reordering terms, an arbitrary (LS) or (l-LS) can be written as

$$\omega \geq \Lambda(\overline{\boldsymbol{\xi}}^k) + \sum_{(i,j)\in A^0} \overline{\xi}^k_{ij} x^0_{ij} + \sum_{j\in J^0} \overline{\epsilon}^k_j y^0_j, \tag{4.3}$$

where $\Lambda(\overline{\boldsymbol{\xi}}^k)$ is a constant value and $\overline{\boldsymbol{\xi}}^k$ and $\overline{\boldsymbol{\epsilon}}^k$ are the corresponding condensed dual multipliers. Now, let us consider two scenarios k_1 and k_2 inducing cuts (l-LS) in a given node of the search tree and such that all coefficients of (4.3) are integer for k_1 and k_2 (with a least one odd value). By first multiplying each coefficient of the two induced cuts by $1/2$ and then summing the two resulting inequalities, we get:

$$\omega \geq \frac{1}{2}\left(\Lambda(\overline{\boldsymbol{\xi}}^{k_1}) + \Lambda(\overline{\boldsymbol{\xi}}^{k_2})\right) + \sum_{(i,j)\in A^0} \frac{1}{2}\left(\overline{\xi}^{k_1}_{ij} + \overline{\xi}^{k_2}_{ij}\right) x^0_{ij} + \sum_{j\in J^0} \frac{1}{2}\left(\overline{\epsilon}^{k_1}_j + \overline{\epsilon}^{k_2}_j\right) y^0_j. \tag{4.4}$$

By rounding up the constant term and each of the coefficients of the above inequality, we get the following zero-half cut:

$$\omega \geq \left\lceil \frac{1}{2}\left(\Lambda(\overline{\boldsymbol{\xi}}^{k_1}) + \Lambda(\overline{\boldsymbol{\xi}}^{k_2})\right)\right\rceil + \sum_{(i,j)\in A^0}\left\lceil \frac{1}{2}\left(\overline{\xi}_{ij}^{k_1} + \overline{\xi}_{ij}^{k_2}\right)\right\rceil x_{ij}^0 + \sum_{j\in J^0}\left\lceil \frac{1}{2}\left(\overline{\epsilon}_{j}^{k_1} + \overline{\epsilon}_{j}^{k_2}\right)\right\rceil y_j^0. \tag{zh-LS}$$

Now, suppose that the cut induced by k_1 is stronger than the one induced by k_2; in this case the resulting zero-half cut (zh-LS) is stronger than the L-shaped cut corresponding to k_2. We use this observation to incorporate zero-half cuts (zh-LS) into the scheme described in Algorithm 4 for separating L-shaped cuts as follows: Let k_1 be the first scenario in $\overline{K}$ that induces an L-shaped cut (l-LS); afterwards, for all other scenarios explored in $\overline{K}$ inducing violated cuts we obtain the corresponding (l-LS) and we combine it with the one obtained by k_1, which yields a stronger violated (zh-LS). This strategy is justified by the fact that the ordering of the elements in $\overline{K}$ is based on how strong the previously produced cuts have been with respect to the cut violation.

A Matheuristic for Generation of Additional *L*-shaped Cuts We have described how we use the current fractional solution $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ in order to obtain a collection of valid inequalities of type (LS), (l-LS), (zh-LS) and (i-LS). The idea now is to use $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ in order to heuristically obtain an alternative feasible pair $(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0)$ and use it to find additional L-shaped cuts at the root node.

The pair $(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0)$ is found by a *matheuristic* that resembles the basic ideas of Local Branching [see Fischetti and Lodi, 2003, Rei et al., 2009]. Let $S_{\mathbf{x}^0} = \{(i,j) \in A^0 \mid \tilde{x}_{ij}^0 > \pi\}$ and $S_{\mathbf{y}^0} = \{j \in J^0 \mid \tilde{y}_j^0 > \pi\}$, be the sets of first-stage allocation and location decisions whose corresponding optimal LP-values are greater than π, where π is a predefined threshold value. If $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ is integer, sets $S_{\mathbf{x}^0}$ and $S_{\mathbf{y}^0}$ exactly represent a feasible first-stage solution. Hamming distances of an arbitrary pair $(\mathbf{x}^0, \mathbf{y}^0)$ to $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ can be defined as

$$\Delta\left(\mathbf{x}^0, \tilde{\mathbf{x}}^0\right) = \sum_{(i,j)\in S_{\mathbf{x}^0}} (1 - x_{ij}^0) + \sum_{(i,j)\in A^0 \setminus S_{\mathbf{x}^0}} x_{ij}^0$$

and

$$\Delta\left(\mathbf{y}^0, \tilde{\mathbf{y}}^0\right) = \sum_{j\in S_{\mathbf{y}^0}} (1 - y_j^0) + \sum_{j\in J^0 \setminus S_{\mathbf{y}^0}} y_j^0.$$

For a given $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$, the alternative solution $(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0)$ is found as follows. Let $\boldsymbol{\Phi}$ be the set of points $\left(\mathbf{x}^0, \mathbf{y}^0, \omega\right)$ defined by the cuts of type (LS), (l-LS), (zh-LS) or (i-LS) that have been added to the model before. The solution $(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0)$ is found by solving the

following LP problem:

$$\left(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0\right) = \arg\min \sum_{j \in J^0} f_j^0 y_j^0 + \sum_{(i,j) \in A^0} c_{ij}^0 x_{ij}^0 + \omega \tag{MH.1}$$

$$\text{s.t.} \quad \Delta\left(\mathbf{x}^0, \tilde{\mathbf{x}}^0\right) \leq \kappa_{\mathbf{x}} \tag{MH.2}$$

$$\Delta\left(\mathbf{y}^0, \tilde{\mathbf{y}}^0\right) \leq \kappa_{\mathbf{y}} \tag{MH.3}$$

$$\Delta\left(\mathbf{y}^0, \tilde{\mathbf{y}}^0\right) \geq 1 \tag{MH.4}$$

$$\left(\mathbf{x}^0, \mathbf{y}^0, \omega\right) \in \mathbf{\Phi} \tag{MH.5}$$

$$\text{(FS.1), (FS.2) and } \left(\mathbf{x}^0, \mathbf{y}^0\right) \in [0,1]^{\left|A^0\right|+\left|J^0\right|}, \tag{MH.6}$$

where the constants $\kappa_{\mathbf{x}}$ and $\kappa_{\mathbf{y}}$ of (MH.2) and (MH.3), respectively, define the *neighborhood* within which we want to find $\left(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0\right)$. Constraint (MH.4) ensures that the new solution will differ from the original one in at least 1 unit of distance with respect to $\mathbf{y}^0$. The later condition is imposed considering that a small change regarding the set of opened facilities is more likely to yield a different (and potentially useful) solution than a change on the allocation decisions.

Once that (MH.1)-(MH.6) is solved, the solution $\left(\hat{\mathbf{x}}^0, \hat{\mathbf{y}}^0\right)$ is used to obtain cuts of type (l-LS) (or (zh-LS) if the feature is enabled) applying the same procedures explained above. Furthermore, we have implemented an iterative process in which problem (MH.1)-(MH.6) is solved M_h times, such that the neighborhood size is slightly increased in each following iteration. More precisely, at a given iteration t, $\kappa_{\mathbf{x}}$ and $\kappa_{\mathbf{y}}$ are given by:

$$\kappa_{\mathbf{x}} = \left\lceil (1+t) \times \vartheta \times \left|S_{\mathbf{x}^0}\right| \right\rceil \quad \text{and} \quad \kappa_{\mathbf{y}} = \left\lceil (1+t) \times \vartheta \times \left|S_{\mathbf{y}^0}\right| \right\rceil,$$

where $\vartheta \in [0,1]$ is a user defined parameter. In our default implementation, parameters π, ϑ and M_h are set to 0.1, 0.75 and 2 respectively.

It is well-known that the incorporation of constraints such as (MH.2) and (MH.3) usually decreases the practical difficulty of a model [see Fischetti and Lodi, 2003], therefore, finding these additional cuts is computationally inexpensive.

4.3.2 Primal Heuristic

Another component of our algorithm is a primal heuristic that uses the information of the current fractional solution $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ and attempts to construct a feasible solution $(\check{\mathbf{x}}^0, \check{\mathbf{y}}^0, \check{\omega})$ that improves the current upper bound. The scheme of the primal heuristic is presented in Algorithm 5.

Algorithm 5 Primal Heuristic

Input: Fractional solution $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0, \tilde{\omega})$; threshold Θ.
1: $\overline{y} = \texttt{averageLP-Val}(\tilde{\mathbf{y}}^0, \Theta)$;
2: $\overline{x} = \texttt{averageLP-Val}(\tilde{\mathbf{x}}^0, \Theta)$;
3: Initialize $\check{J}^0 = \emptyset$, $\check{R}^0 = \emptyset$ and $\check{\omega} = 0$;
4: $\check{J}^0 = \{j \in J^0 \mid \tilde{y}^0_j > \texttt{rand}[\Theta, \overline{y}]\}$;
5: $\check{R}^0 = \{i \in R^0 \mid \sum_{(i,j)\in A^0} \tilde{x}^0_{ij} > \texttt{rand}[\Theta, \overline{x}]\}$;
6: **if** $|\check{J}^0| > 0$ **then**
7: Set $\check{y}_j = 1$ if $j \in \check{J}^0$ and $\check{y}_j = 0$ otherwise;
8: Set $\check{x}_{ij^*} = 1$ if $i \in \check{R}^0$ and $j^* = \arg\min_{\{(i,j)\in A^0 | j \in \check{J}^0\}} c_{ij}$ and $\check{x}_{ij} = 0$ otherwise.
9: $\check{\omega} = \max_{k\in K} \rho\left(\check{\mathbf{x}}^0, \check{\mathbf{y}}^0, k\right)$
10: Try to set $(\check{\mathbf{x}}^0, \check{\mathbf{y}}^0, \check{\omega})$ as incumbent solution;

Function $\texttt{averageLP-Val}(\tilde{\mathbf{y}}^0, \Theta)$ (see line 1), is given by

$$\frac{\sum_{j\in J^0: \tilde{y}_j > \Theta} \tilde{y}^0_j}{|J^0 : \tilde{y}^0_j > \Theta|};$$

which means that $\overline{y}$ is computed using only those elements whose LP-values are larger than Θ, where Θ is a predefined threshold value. The value $\overline{x}$ is computed similarly (see line 2).

A key element of the proposed heuristic is given in lines 4 and 5: set $\check{J}^0$ (resp. $\check{R}^0$) is built by adding an element j (resp. i) if $\tilde{y}^0_j$ (resp. $\sum_{(i,j)\in A^0} \tilde{x}^0_{ij}$) is greater than a value, uniformly randomly generated in the interval $[\Theta, \overline{y}]$ (resp. $[\Theta, \overline{x}]$). Thanks to the use of average LP-values $\overline{x}$ and $\overline{y}$, important information about the solution topology is transferred from the current LP solution to the heuristic solution. On the other hand, the use of random thresholds (lines 4 and 5) provides diversification to the heuristic and helps in escaping local optima. The feasible first-stage solution $(\check{\mathbf{x}}^0, \check{\mathbf{y}}^0)$ is computed in lines 7 and 8 by means of a very simple greedy heuristic. The heuristic value of $\check{\omega}$ is found in line 9. Although $|K|$ ILP problems (R.1)-(R.6) have to be solved they are not solved to optimality but until a gap of less than 1% is reached (which typically takes at most a few seconds). The default value of Θ was set to 0.01.

4.3.3 Auxiliary Variables and Branching Priorities

Looking more carefully at the objective function of a k-th subproblem, one easily observes that for each customer $i \in R$, its assignment variables are grouped together into binary decisions: (i) the customer is served in the first stage ($\sum_{(i,j)\in A^0} x^0_{ij}$), and (ii) the customer is served in the first stage by a *wrong* facility ($\sum_{(i,j)\in A^0 \setminus A^k} x^0_{ij}$). This motivates us to introduce additional binary decision variables and impose a new non-standard branching on them. More precisely, we introduce auxiliary binary variables

$\mathbf{q}$, $\mathbf{s} \in \{0,1\}^{|R^k|}$, for all $k \in K$, as follows:

$$q_i^k = \sum_{(i,j)\in A^0} x_{ij}^0, \quad \forall i \in R^k,\ \forall k \in K \tag{4.5}$$

$$s_i^k = \sum_{(i,j)\in A^0 \setminus A^k} x_{ij}^0, \quad \forall i \in R^k,\ \forall k \in K. \tag{4.6}$$

These auxiliary variables play two important roles in our algorithmic framework. First, they are useful in the efficient construction of the LP (and ILP) SPs. The right-hand-side of (R.2) and (R.3) can be fixed for each $i \in R^k$ without the need of any extra loop to sum up the values of the first-stage solution $\tilde{x}^0$. Second, and more important, these auxiliary variables are used to *guide* the branching in a more effective way by imposing higher branching priorities on them. Clearly, fixing to 0 or to 1 one of these variables immediately fixes the value of other variables. For instance if $q_i^k = 1$ and $s_i^k = 0$ for a given $i \in R^k$ (customer $i \in R^k$ has been allocated in the first-stage to a facility through a link that is available in scenario k in the second stage), then $x_{ij}^k = z_{ij}^k = 0$ $\forall (i,j) \in A^k$. Otherwise, if $q_i^k = 0$ (customer $i \in R^k$ has not been allocated in the first-stage to any facility), then $s_i^k = 0$, $\sum_{i\in R^k} x_{ij}^k = 1$ and $z_{ij}^k = 0$ $\forall (i,j) \in A^k$. Other combinations can be analyzed straightforwardly.

Adding these variables and constraints (4.5)-(4.6) does not modify the polyhedral characterization of (4.1)-(4.2), so the computational effort does not intrinsically change by including them.

4.4. Computational Results

In this section we first introduce two sets of benchmark instances that resemble application of facility location in transportation networks and in the disaster management, respectively. We use these instances (i) to analyze the properties of the obtained solutions and their dependence on the cost structure, (ii) for showing the advantages of the recoverable robustness, and (iii) for assessing the performance of the proposed branch-and-cut algorithm. Finally, we also compare the performance of the proposed algorithm with the performance of CPLEX when solving formulation (4.1)-(4.2) directly (i.e., as a compact model).

All the experiments were performed on an Intel Core$^{\text{TM}}$ i7 (4702QM) 2.2GHz machine (8 cores) with 16 GB RAM. The branch-and-cut was implemented using CPLEX$^{\text{TM}}$ 12.5 and Concert Technology framework. When testing our branch-and-cut all CPLEX parameters were set to their default values, except the following ones: (i) All cuts were turned off, (ii) heuristics were turned off, (iii) preprocessing was turned off, (iv) the time limit was set to 600 seconds. Besides, higher branching priorities were given to $\mathbf{y}^0$ and to the auxiliary variables $\mathbf{q}$ and $\mathbf{s}$ as described in §4.3.3.

We have turned some CPLEX features off (only when running our algorithm) in order to make a fair assessment of the performance of the techniques described in §4.3.

4.4.1 Benchmark Instances

We consider two classes of instances, that we refer to as `Trans` and `Dis`. Instances of the first class are intended to resemble real transportation networks in which the transportation costs depend on both the distance to be covered and the amount of commodities to be transported, and where the set-up cost of facilities strongly depends on the demographic characteristics of the corresponding (urban) area. `Dis` instances approximate situations such as humanitarian relief in natural disasters in which some transportation links are *interdicted*, i.e., they are damaged so that the transportation time can be severely increased. We assume that if a given city $i \in R^k$ requires to be served but each path from any $j \in J^k$ to i contains at least one interdicted link, then the city is still assisted although at a very high response time. Besides, set-up costs f_j^0 are such that one might favor to install facilities in cities where the average distance to all the potential customers is relatively small.

`Trans` Instances In this class of instances we consider three groups: `US`, `Germany` and `ND-I`. In groups `US` and `Germany` we consider the geographical coordinates and updated data of population of the 500 most populated cities in each country [see United Nations Statistics Division, 2013]. In group `ND-I` we consider random instances with up to 500 nodes randomly located in a unit square and population being an integer number taken uniformly at random from the interval $[1 \times 10^4, 2.5 \times 10^6]$. We denote by d_{ij} the Euclidean distance between cities i and j, and by pop_i the population size of city i.

Given the coordinates and the population size associated with each node, an instance of the RRUFL is then generated as follows:

(i) take the first n cities in terms of population;

(ii) define R^0 by randomly selecting 50% of the cities;

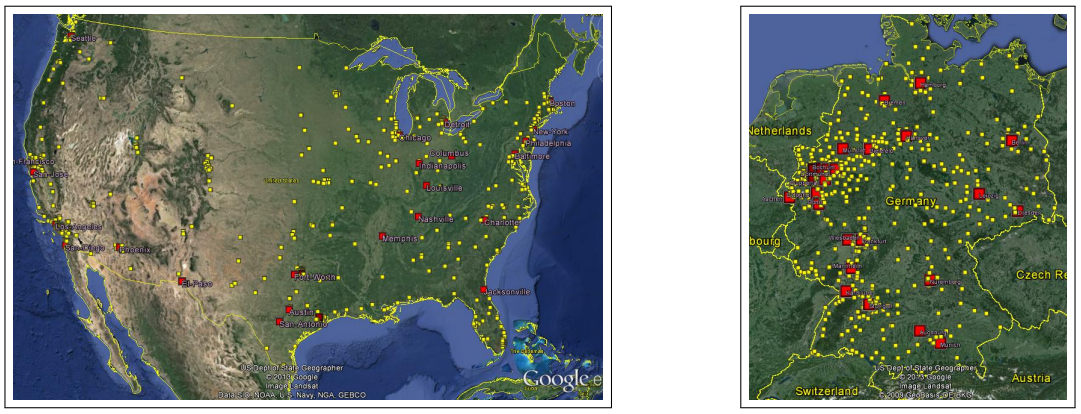
(iii) for $k \in K$ define R^k by randomly taking $|R^0| \times \texttt{rand}[0.4, 0.6]$ cities from R^0;

(iv) for $k \in K$ define J^k by randomly taking $(n - |R^k|) \times \texttt{rand}[0.2, 0.3]$ cities from $1, \ldots, n$ ($J^0 = \cup_{k \in K} J^k$);

(v) for $k \in K$ define $A^k = R^k \times J^k$ ($A^0 = R^0 \times J^0$);

(vi) first- and second-stage transportation/allocation costs are defined as $c_{ij}^0 = d_{ij} \times \frac{1}{2}(pop_i + pop_j) \times \varphi$, $c_{ij}^k = (1 + \sigma_1) \times c_{ij}^0$ and $r_{ij}^k = (1 + \sigma_2) \times c_{ij}^0$ for $k \in K$;

(vii) first- and second-stage set-up costs and penalties are defined as $f_j^0 = \rho \times pop_j$, $f_j^k = (1 + \sigma_3) \times f_j^0$ and $p_j^k = (1 + \sigma_4) \times f_j^0$ for $k \in K$.



(a) `Trans-US` (b) `Trans-Germany`

FIGURE 4.2: Representation of `Trans` Instances.

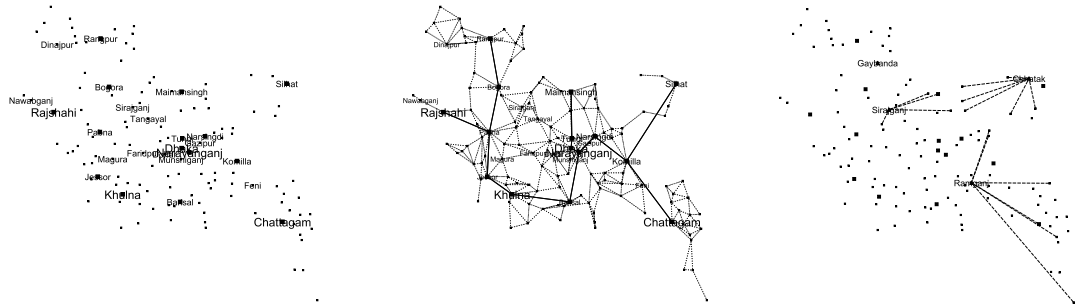
All coefficients are finally rounded to their nearest integer values.

Parameter φ is given in \$ per unit of distance per unit of demand, so the allocation costs are purely expressed in \$; parameter ρ is given in \$ per inhabitant (so the larger a city is, the more expensive the set-up of a facility is); parameters σ_1, σ_2, σ_3 and σ_4 are $[0, 1]$ factors representing the increase of the allocation and set-up costs in the second stage.

Figures 4.2(a) and 4.2(b) show the graphical representation of the 500 cities used in groups `US` and `Germany` respectively (the name of the first 25 cities are provided). For $n = 500$, each scenario resembles a UFL instance with ≈ 125 customers and ≈ 100 locations (the sets J^k and R^k may intersect).

In our experiments we use: $n \in \{100, 250, 500\}$, $\varphi \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, $\rho \in \{0.001, 0.01, 0.1, 1\}$ $\sigma_1, \sigma_2 \in \{0.05, 0.5\}$, and $\sigma_3, \sigma_4 \in \{0.10, 1\}$. In our computations we consider up to 75 scenarios which are created in advance. By doing this, when dealing with instances with 25 scenarios, we simply use the first 25 scenarios out of those 75. The same applies for 50 scenarios. The scenarios are identical for the different values of all other parameters. By proceeding in this way, it is easier to measure the impact of considering a larger number of scenarios. For a given group (`US`, `Germany`, or `ND-I`) there are $3 \times 4 \times 4 \times 2 \times 2 \times 2 \times 2 \times 3 = 2304$ instances to be solved.

`Dis` Instances In this class of instances we consider three groups: `Bangladesh`, `Philippines` and `ND-II`. In group `Bangladesh` (resp. `Philippines`) we consider the geographical coordinates and updated data of population of the 128 (resp. 100) most populated cities in each case [see United Nations Statistics Division, 2013]; in group `ND-II` we consider random instances with 100 nodes randomly located in a unit square and the size of the population is taken uniformly at random from $[1 \times 10^4, 2.5 \times 10^6]$. In the case of groups `Bangladesh` and `Philippines` we use pairwise Euclidean distances between selected cities and embed them in a network $N = (V, A)$, with V being the



(a) Nodes of the input network

(b) $N = (V, A)$ after adding links

(c) Example of a 1st-stage solution

FIGURE 4.3: Construction process of `Dis` Instances and an example of a first-stage solution. `Bangladesh` Instances.

set of n cities and A the allocation links ($n = 128$ for group `Bangladesh` and $n = 100$ for group `Philippines`). For the case of the group `ND-II`, the network $N = (V, A)$ is obtained such that a link is established between two cities i and j if the Euclidean distance is smaller than or equal to $\alpha/\sqrt{n}$ (α is an input parameter fixed to 1.6 in our computations). Figure 4.3(a) shows the location of the 128 cities for the `Bangladesh` group of instances, Figure 4.3(b) illustrates the embedded network $N = (V, A)$ of the same group, and Figure 4.3(c) shows an example of a first-stage solution.

With the information of each group, `Bangladesh`, `Philippines` or `ND-II`, an instance of the RRUFL is generated as follows:

(i) define R^0 by randomly selecting $t\%$ of the cities, with $t \in \{25, 50, 75\}$;

(ii) for $k \in K$ define R^k by randomly taking $|R^0| \times \texttt{rand}[0.4, 0.6]$ cities from R^0;

(iii) for $k \in K$ define J^k by randomly taking $(n - |R^k|) \times \texttt{rand}[0.08, 0.12]$ cities from $1, \ldots, n$ ($J^0 = \cup_{k \in K} J^k$);

(iv) first-stage allocation costs c^0_{ij} are equal to the shortest path cost between i and j in $N = (V, A)$ using Euclidean distances d_{uv}.

(v) for the second-stage allocation costs we consider random link interdiction, that is: let I^k be a set of $f \times |A| \times \texttt{rand}[0.8, 1.2]$ links randomly chosen from A. Then $d^k_{uv} = d_{uv}$, for all $\{u, v\} \in A \setminus I^k$, and $d^k_{uv} = 100 \times d_{uv}$, for all $\{u, v\} \in I^k$, so c^k_{ij} is equal to the cost of the shortest path between i and j with edge costs given by $\mathbf{d}^k$. Reallocation cost r^k_{ij} is $1.5 \times d^k_{ij}$, for $k \in K$;

(vi) first-stage set-up costs are given by $f^0_j = \sum_{i \in R^0} c^0_{ij}/|R^0|$, and second-stage set-up and penalty costs are given by $f^k_j = (1 + \sigma_3) \times f^0_j$ and $p^k_j = (1 + \sigma_4) \times f^0_j$, for $k \in K$.

The remaining parameters are $f \in \{0, 0.10, 0.25, 0.50\}$, $\sigma_3 = \{0.00, 1.00\}$ and $\sigma_4 = \{0.10, 1.0, 4.0\}$. All possible parameter settings, in combination with $k \in \{25, 50, 75\}$

(a) $(0.1, 1.0)$, $|\mathbf{y}^0| = 3, |\mathbf{x}^0| = 8$

(b) $(1.0, 0.1)$, $|\mathbf{y}^0| = 14, |\mathbf{x}^0| = 54$

FIGURE 4.4: Solutions considering different combinations of (σ_3, σ_4) (Instances `US`, $n = 500$, $\varphi = 0.001$, $\rho = 0.1$ $\sigma_1 = 0.5$, $\sigma_2 = 0.05$ and $|K| = 25$)

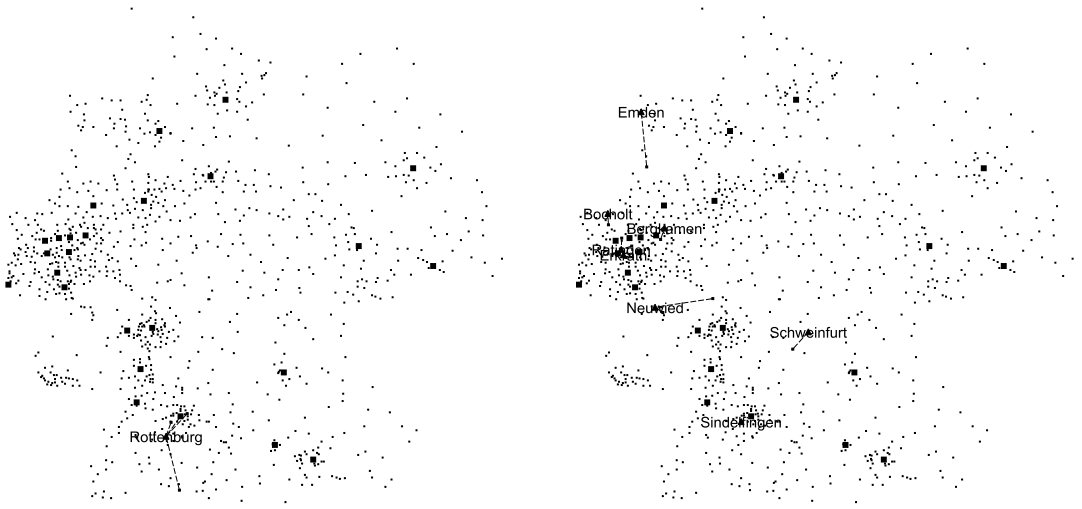
imply that there are $3 \times 4 \times 2 \times 3 \times 3 = 216$ instances to be solved for each fixed value of n within each group.

4.4.2 `Trans` Instances: Robustness and Recoverability

Influence of the Cost Structure The characteristics of a *robust* first-stage solution and the corresponding recovery actions depend not only on the scenario structure but also on the cost structure. If, for example, for a given instance the second-stage costs are very high with respect to the first-stage costs then the solutions of the RRUFL will tend to have more facilities and assignments defined in the first stage. Likewise, if the second-stage set-up costs are much higher than the penalty costs ($f_j^k >> p_j^k$), we would expect that more facilities will be opened in the first-stage (and eventually more assignments) than if $f_j^k \leq p_j^k$, where the cost of setting-up a facility in the second stage is cheaper than the penalty for a facility placed at a non-available location.

In Figure 4.4 we show the later case by comparing two solutions of an instance of group `US`. For the first one (Figure 4.4(a)), the penalties are $\approx 81\%$ more expensive than the second-stage set-up costs, while for the second one (Figure 4.4(b)), the penalties are 45% cheaper. We can see how changing the relation between f_j^k and p_j^k leads to very different solutions: while in the first case 3 facilities are opened in the first stage and 8 customers are allocated to them, in the second case 14 facilities are opened in the first stage and 54 customers are allocated.

The relation between parameters φ (\$ per unit of distance per inhabitant) and ρ (\$ per inhabitant), also influences the solution structure. Assume that we are given an instance with $\varphi < \rho$ (set-up costs are higher than the allocation costs) and another



(a) $(0.0001, 0.01)$, $|\mathbf{y}^0| = 1, |\mathbf{x}^0| = 5$ (b) $(0.01, 0.001)$, $|\mathbf{y}^0| = 8, |\mathbf{x}^0| = 12$

FIGURE 4.5: Solutions considering different combinations of (φ, ρ) (Instances `Ger`, $n = 250$, $\sigma_1 = 0.5$, $\sigma_2 = 0.5$ $\sigma_3 = 0.1$, $\sigma_4 = 1.0$ and $|K| = 25$)

instance with $\varphi > \rho$ (allocation costs are higher than the set-up costs). We would expect that the solution of the second instance will be comprised by a larger first-stage component compared to the solution of the first instance. Figure 4.5 depicts this by comparing the solution obtained for $\varphi = 0.0001$ and $\rho = 0.01$ (Figure 4.5(a)) with the one obtained for $\varphi = 0.01$ and $\rho = 0.001$ (Figure 4.5(b)). In the first case, only one facility is open in the first stage and 5 allocations are defined, while in the second case 8 facilities are installed and 12 allocations are established. This effect is quite intuitive considering that the second stage costs are proportional to the first stage costs for these instances: it is better to open facilities in the same place where the demand is located, i.e., in a subset of $R^0 \cap J^0$, in order to avoid high allocation expenses ($\varphi > \rho$).

The Gain of Recovery A more accurate measure of the benefits of the *recovery* can be calculated by comparing the solutions obtained for the RRUFL with those obtained for the RUFL presented in §4.2.1. Recall that the RUFL model is such that facilities can only be opened in the first stage, whereas allocations can only be established in the second stage. Hence, no recovery actions (in terms of setting-up new facilities or re-allocating customers) are allowed. To illustrate the benefits of the recovery, we now define a measure that we will refer to as the *Gain of Recovery* (*GoR*). *GoR* is defined as the relative gain in terms of cost when using the solution produced by our recoverable robust approach instead of the one produced by the approach without recovery (the RUFL, in our case).

In Table 4.1 we report on statistics regarding the *GoR*. Columns $GoR(OPT_{\mathcal{RR}})$ correspond to statistics of the *GoR* defined as $GoR(OPT_{\mathcal{RR}}) = \frac{OPT_R - OPT_{\mathcal{RR}}}{OPT_R} \times 100\%$, where OPT_R is the objective function value produced by the RUFL. Columns $GoR(OPT_\omega)$ correspond to statistics of the *GoR* defined as $GoR(OPT_\omega) = \frac{\omega_R - \omega}{\omega_R} \times 100\%$, where ω_R

		$GoR(OPT_{\mathcal{RR}})$			$GoR(\omega)$		
Group	$\|K\|$	Median	Ave.	Max	Median	Ave.	Max
US	25	34.73	36.28	89.07	31.55	33.96	82.87
	50	38.64	39.30	91.94	35.52	37.31	87.46
	75	41.34	40.81	93.50	34.61	36.29	89.95
Ger	25	29.17	32.73	90.01	24.91	26.39	84.37
	50	29.44	34.39	92.65	26.66	28.31	88.61
	75	32.52	37.26	93.90	28.40	32.05	90.62
ND-I	25	24.69	25.31	79.84	29.41	30.24	70.47
	50	23.47	25.75	83.49	22.57	26.18	75.79
	75	24.12	26.79	85.22	22.76	27.33	78.32

TABLE 4.1: Statistics of two measures of the *Gain of Recovery* for different values of n and $|K|$ (Groups US, Germany and ND-I with $n = 100$)

is the worst-case second stage cost for the RUFL. The obtained values emphasize the practical benefits of recoverable robustness in cases in which recovery is possible; both, the costs of the complete policy (first- and second-stage solutions) and the worst-case second stage solutions are on average 25-40% cheaper (and the difference can scale above 90%). These results clearly justify the benefits of the recovery in the second stage, when compared to a less flexible decision making policy.

The Effort for Robustness and the Price of Robustness The more scenarios (possible data realizations) we take into account, the more *robust* the first-stage solution is expected to be. Nonetheless, this *additional* robustness is obtained at the expenses of (i) an increase of the difficulty of the problem, since a larger search space must be considered, and (ii) an increase of the total solution cost, $OPT_{\mathcal{RR}}$, because more facilities have to be opened and more allocations have to be established in the first stage or because a new worst-case scenario induces a higher robust recovery cost (i.e., ω increases). The first of these effects has been coined as the *Effort for Robustness* in [Álvarez-Miranda et al., 2013c] ; the second effect is similar to what is called the *Price of Robustness* in [Bertsimas and Sim, 2003].

To illustrate these effects, in Table 4.2 we report average values of the results obtained for groups US, Germany and ND-I for varying number of nodes and scenarios (columns *Group*, n and $|K|$, respectively). The presented values are related to the solution characteristics and to the algorithmic performance. Each row corresponds to the results of 256 instances. Column *Time [s]* reports the average running times expressed in seconds; column *Gap (%)* shows the average gaps attained within the time limit; the average number of facilities opened in the first stage is reported in column $|\mathbf{y}^0|$ and the average number of first-stage allocations is given in column $|\mathbf{x}^0|$; in columns $\Delta OPT\%$ and $\Delta\omega\%$ we report the average relative increase in the value of $OPT_{\mathcal{RR}}$, resp. ω, when considering 50 and 75 scenarios with respect to the value obtained for 25 scenarios. In column *#Opt* the number of instances that were solved to optimality (out of 256) is shown.

Group	n	$\|K\|$	Time [s]	Gap (%)	$\|\mathbf{y}^0\|$	$\|\mathbf{x}^0\|$	$\Delta OPT\%$	$\Delta\omega\%$	#(l-LS)	#(l-LS)$_{\text{MH}}$	#(i-LS)	#BBN	#Opt
US	100	25	44.94	0.00	5	7	0.00	0.00	54	4	0	342	251
		50	72.09	0.01	5	6	0.12	0.64	79	4	1	284	252
		75	86.56	0.01	5	6	1.11	5.23	96	3	0	219	250
	250	25	243.26	0.18	11	14	0.00	0.00	105	9	0	183	175
		50	229.45	0.06	11	11	2.89	5.61	89	3	0	125	197
		75	285.92	0.07	11	11	3.50	6.68	99	2	0	84	172
	500	25	458.06	1.10	18	25	0.00	0.00	61	11	0	19	82
		50	586.68	1.32	15	19	2.66	3.90	67	5	0	7	11
		75	600.00	1.99	16	20	3.63	5.06	79	2	0	1	0
Ger	100	25	21.52	0.00	6	5	0.00	0.00	43	4	0	143	256
		50	45.97	0.00	6	5	0.01	0.01	67	4	0	169	252
		75	70.71	0.00	6	6	1.51	1.00	98	4	1	171	249
	250	25	347.82	0.36	13	14	0.00	0.00	274	9	1	243	134
		50	407.43	0.37	12	13	2.94	4.15	172	6	1	165	107
		75	449.38	0.55	12	13	3.03	4.25	158	6	0	98	92
	500	25	431.51	0.52	17	20	0.00	0.00	59	9	0	31	96
		50	542.32	0.58	17	18	1.34	2.78	65	3	0	13	43
		75	600.00	2.50	23	28	8.13	6.27	79	1	0	1	0
ND-I	100	25	37.22	0.00	7	10	0.00	0.00	94	5	0	282	256
		50	31.60	0.00	6	10	6.44	11.85	66	3	0	99	256
		75	48.10	0.00	6	10	6.45	11.82	91	3	0	100	256
	250	25	296.20	0.07	16	18	0.00	0.00	103	5	0	287	153
		50	384.78	0.12	15	18	7.32	8.22	103	5	1	167	115
		75	330.24	0.13	14	16	8.86	10.71	106	4	1	106	151
	500	25	543.29	1.98	25	38	0.00	0.00	77	15	0	30	33
		50	584.45	2.01	24	33	0.67	5.07	63	7	0	6	12
		75	600.00	2.38	23	36	12.25	18.06	79	2	0	1	0

TABLE 4.2: Statistics of solution characteristics and algorithmic performance for different values of n and $|K|$ (Groups US, Germany and ND-I)

The *Effort for Robustness* is clearly illustrated by the worsening of the algorithmic performance when increasing the number of scenarios: (i) the running times increase (cf. column *Time [s]*); (ii) the attained gaps increase (cf. column *Gap (%)*); and, hence, the number of solutions solved to optimality (cf. column *#Opt*) decreases.

The *Price of Robustness* is demonstrated in columns $\Delta OPT\%$ and $\Delta\omega\%$, where one can see that, without exception, the average values of the solution cost and the robust recovery cost increase when increasing $|K|$ from 25 to 50 and from 25 to 75. We observe that in all cases (except for two entries of the group Germany) the value of $\Delta OPT\%$ is smaller than the value of $\Delta\omega\%$. This means that the obtained first-stage solutions are such that they allow to reduce the impact of a higher robust recovery cost by *balancing* robustness and recoverability. This difference between $\Delta OPT\%$ and $\Delta\omega\%$ can be regarded as the *marginal cost benefit* due to the possibility of defining a first-stage solution that can be recovered in a second-stage. The two entries in which the average value of $\Delta OPT\%$ is greater than the average value of $\Delta\omega\%$, can be explained by the fact that not all instances are solved to optimality (especially for $n = 500$ and

$|K| = 75$), so the non-optimal first-stage solutions are such that the corresponding recovery costs are sub-optimally high.

In columns $|\mathbf{y}^0|$ and $|\mathbf{x}^0|$ one can see that the size of the first-stage solution is more or less constant for a given n, regardless of the value of $|K|$. Because of the chosen criterion for generating scenarios, solutions are rather *balanced*: none of them is too different from the others. Hence, our model is able to capture the nature of the uncertainty already with 25 scenarios, so increasing the number of scenarios does not produce a measurable effect on the structure of the first-stage solution but only on the second-stage recovery actions (which induces a higher value of ω).

4.4.3 `Trans` Instances: Algorithmic Performance

Assessment of Algorithmic Enhancements In §4.3 we have described several enhancements for our algorithm: cut strengthening based on dual-lifting, scenario sorting, zero-half cuts, matheuristic generation of cuts and branching priorities on auxiliary variables. In Figure 4.6 we show box-plots of the gaps attained when solving instances of group `US` with $n = 250$ when incrementally including the proposed techniques. Each box represents the distribution of the obtained gaps over a set of 678 instances. The first box-plot corresponds to the basic setting of the algorithm, that is, with the cuts of type (LS) and (i-LS); the second box-plot shows the gaps obtained when using the strengthening technique based on dual variables (i.e., when adding (l-LS) instead of (LS)); in the third box-plot we display the gaps obtained when adding the strategy of scenario sorting; the fourth box-plot shows the gaps attained when adding cuts generated by our matheuristic approach; the gaps attained when strengthening found cuts using zero-half cuts are given in the fifth box-plot; finally, in the sixth box-plot we show the gaps obtained when imposing higher branching priorities on the auxiliary variables (this last configuration is our *default* one). The bold points are the maximum gaps, asterisks are the average gaps and on top of each box we show the total number of instances (out of 678) that were solved to optimality.

The results clearly demonstrate that all the proposed techniques *contribute* to the effectiveness of the algorithm and *complement* each other: the average gap decreases, more instances are solved to optimality and the performance is more stable. In terms of the marginal contribution to the algorithmic performance, the strengthening technique based on dual-lifting and imposing higher branching priorities on the auxiliary variables seem to be the techniques that produce largest improvements of the algorithmic performance. Using the basic strategy, only 131 instances can be solved to optimality. On the contrary, using a combination of our enhancement methods, 544 instances are solved to optimality within the same time limit.

More detailed indicators of the effectiveness of the considered cuts and their algorithmic performance are provided in Table 4.2. In columns #(l-LS), #(l-LS)$_{\text{MH}}$ and #(i-LS) we

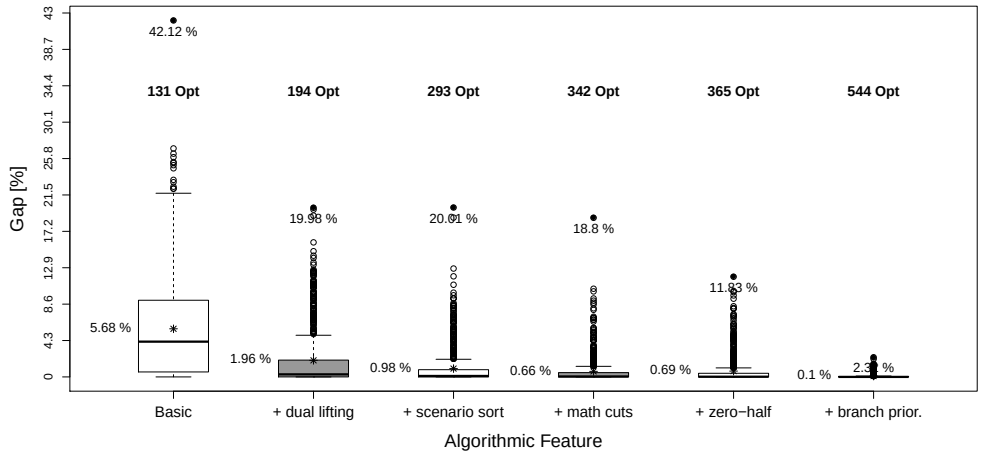


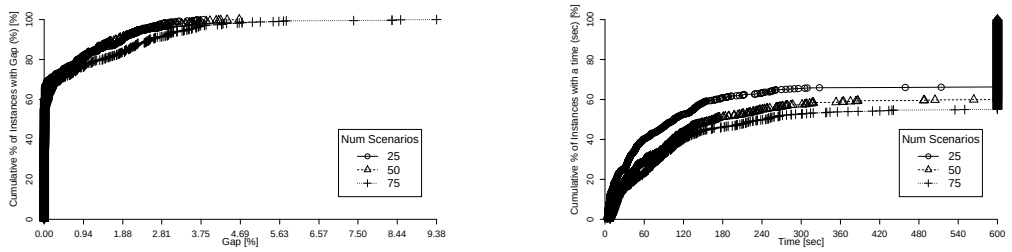
FIGURE 4.6: Influence of the special enhancement strategies on the algorithmic performance (Group US, $n = 250$)

report the average number of L-shaped Cuts, L-shaped Cuts found via the matheuristic approach, and integer L-shaped Cuts, respectively, that are added during the optimization process. Column *#BBN* reports the average number of enumeration nodes explored within the running time.

It is remarkable that (cf. column #(i-LS)), integer L-shaped cuts are added in very rare cases. In a more detailed analysis we observed that whenever the current solution $(\tilde{\mathbf{x}}^0, \tilde{\mathbf{y}}^0)$ was integer, usually (l-LS) were able to close the gap, so no attempt was made to find integer L-shaped cuts.

The number of explored enumeration nodes (column *#BBN*) clearly shows that increasing the size of the instance and the number of scenarios produces a slowdown in the exploration of the search-space. This happens because more time is spent at each node solving the separation problem and performing the algorithmic enhancements described before.

The effectiveness of the proposed solution approach on the 2034 instances derived from group US is shown in Figure 4.7. The performance profile of the attained gaps for different values of $|K|$ in Figure 4.7(a) shows that (regardless of the value of $|K|$): (i) about 65% of the instances are solved to optimality or a very small gap is reached, (ii) for almost 80% of the instances a gap of less than 1.5% is reached, and (iii) for almost all, expect 5 instances, the attained gap is less than 4.7%. As for the running times, Figure 4.7(b) shows that: (i) between 20% and 40% of the instances can be solved in less than 60 seconds, (ii) about 50% can be solved in less than 300 seconds, and (iii) for almost 45% of the instances the time limit is reached. Detailed performance profiles of the attained gaps for different values of n are provided in the Appendix (Figure 4.10). The observed behavior is not very different in the case of the group



(a) Performance Profile of attained gaps

(b) Performance Profile of running times

FIGURE 4.7: Performance Profile of attained gaps and running times for different number of scenarios (Group `US`, 2034 instances)

`Germany` (Figure 4.11 in the Appendix), nor in the case of the group `ND-I` (Figure 4.12 in the Appendix).

Recall that for our branch-and-cut approach we have disabled some CPLEX features (pre-processing, heuristics and general-purpose cutting planes) in order to get a better assessment of the proposed techniques. For the sake of completeness, we have performed some experiments where all CPLEX parameters are set to their default values. In Table 4.6 in the Appendix we report statistics on the algorithmic performance when solving instances with $n = 100$ of groups `US`, `Germany` and `ND-I` with the default CPLEX settings. Comparing this table with Table 4.2, one observes that enabling these CPLEX features does not produce any improvement on the algorithmic performance; moreover, it actually deteriorates it: fewer instances are solved to optimality within the same time limit and the attained gaps are slightly worse.

As mentioned above, formulation (4.1)-(4.2) can also be solved directly through any state-of-the-art MIP solver such as CPLEX. Nonetheless, this straightforward strategy cannot be applied successfully, even to our smallest instances ($n = 100$). In Table 4.3 (cf. Table 4.7) we report statistics on the performance of CPLEX with default configuration when solving instances of class `Trans` with $n = 100$ (within the same time limit of 600 seconds). We observe that much less instances are solved to optimality, and the gaps of the unsolved instances can be as high as 99%(!) and average gaps can range from 17.0% to more than 60.0%. In the table we also report the number of explored enumeration nodes (#BBN) and the number of cuts added by the solver (#CPX Cuts). What seems surprising is the small number of general-purpose cuts added during the optimization with respect to the number of explored nodes; this means that cutting planes as those included in CPLEX are insufficient to tackle the structure of the RRUFL (at least for the considered instances) and the lower-bound improvement mainly relies on branching.

		Opt. Times		Attained Gaps			B&C Indicators	
Group	k	Ave.	#Opt	Ave.	max	#Nopt	#BBN	#CPX Cuts
	25	17.43	256	–	–	0	213	14
US	50	36.08	237	39.56	84.34	19	182	26
	75	54.16	221	62.72	99.1	35	190	22
	25	14.20	256	–	–	0	72	9
Ger	50	41.87	244	17.8	42.85	12	133	19
	75	66.25	206	31.6	98.01	50	269	25
	25	7.72	253	–	–	0	138	13
ND-I	50	38.10	256	–	–	0	131	17
	75	54.40	228	49.48	99.63	28	175	31

TABLE 4.3: Algorithmic performance of CPLEX when solving the compact model. `Trans` Instances with $n = 100$ (256 instances per row).

4.4.4 `Dis` Instances: Solutions and Algorithmic Performance

Solutions `Dis` class is intended to represent situations of natural disasters in which different number of cities are likely to need assistance ($t = \{0.25, 0.50, 0.75\}$), few cities are in conditions to host a facility, a portion of the allocation links can be heavily damaged ($f = \{0.00, 0.10, 0.25, 0.50\}$) and the attractiveness of a location depends more on its position than on its economical characteristics.

As in the case of `Trans` instances, the structure of the first-stage solutions strongly depends on the instance definition. Figure 4.8 displays solutions of instances of group `Philippines` considering different combinations of (t, f). We can observe that for a fixed value of t (Figures 4.8(a)-4.8(c) for $t = 0.50$ and Figures 4.8(d)-4.8(f) for $t = 0.75$), a larger first-stage component is defined when increasing f, i.e., more facilities are opened and more allocations are defined. This behavior is expected due to the dramatic effect produced by the presence of road failures; it is better to define *robust* first-stage allocations to prevent from very high transportation times in the second stage.

Note that, from a practical point of view, if a given city i is assigned in the first stage to a facility j, the *actual* allocation cost (the one incurred when assistance comes from j to i after the disaster) will still be scenario dependent (chosen roads might be damaged in any case). However, this first-stage decision can help to decision makers (i) to define preventive plans to endure some roads, (ii) to have in mind how to access the affected areas regardless of the presence of failures, and (iii) to make sure that if a given allocation should be re-defined, this re-allocation will be economically efficient (due to the worst-case emphasis of the model).

Figures 4.8(a)-4.8(f) have been produced by transforming our solutions into `kml` files that can be displayed with the Google Earth free software [see Google, 2014].

In Table 4.4 (equivalent to Table 4.2) additional information on solutions' structure is provided. As in the case of `Trans` instances, we can see that the number of scenarios does not change the average values of $|\mathbf{y}^0|$ and $|\mathbf{x}^0|$, which again shows that our model

(a) $(0.50, 0.10)$, $|\mathbf{y}^0| = 3, |\mathbf{x}^0| = 11$ (b) $(0.50, 0.25)$, $|\mathbf{y}^0| = 4, |\mathbf{x}^0| = 5$ (c) $(0.50, 0.50)$, $|\mathbf{y}^0| = 7, |\mathbf{x}^0| = 13$

(d) $(0.75, 0.10)$, $|\mathbf{y}^0| = 3, |\mathbf{x}^0| = 2$ (e) $(0.75, 0.25)$, $|\mathbf{y}^0| = 5, |\mathbf{x}^0| = 29$ (f) $(0.75, 0.50)$, $|\mathbf{y}^0| = 7, |\mathbf{x}^0| = 58$

FIGURE 4.8: Solutions considering different combinations of (t, f) (Group `Philippines`, $\sigma_3, \sigma_4 = 1$, $|K| = 50$)

tackles uncertainty in a way that cost structure influences more the characteristics of first-stage solutions than the uncertainty. The values reported in columns $\Delta OPT\%$ and $\Delta\omega\%$ reinforce the previous observation. There is an important increment of the total cost of the solutions ($\Delta OPT\%$) when increasing $|K|$ but most of this increment is due to the second-stage component ($\Delta\omega\%$). The marginal difference between $\Delta\omega\%$ and $\Delta OPT\%$ is due to the robustness cost of the corresponding first-stage solutions. The values of $\Delta OPT\%$ and $\Delta\omega\%$ are one order of magnitude larger than those obtained for `Trans`; this can be explained by the great increase in the second stage costs.

Further insights on the influence of the cost structure on the first-stage solutions are shown in the Appendix: Figures 4.13 and 4.14 (`Bangladesh` group), Figures 4.16 and 4.17 (`Philippines` group), and Figures 4.19 and 4.20 (`ND-II` group). From these figures we can see that the average values of $|\mathbf{y}^0|$ and $|\mathbf{x}^0|$ depend more on factors t and f (as previously shown in the examples) than on (σ_3, σ_4) (the second-stage set-up and penalty factors).

Algorithmic Performance As in the case of `Trans` instances, one can identify the *Effort for Robustness* when solving `Dis` instances. From columns *Time* [s], *Gap* (%) and *#Opt* in Table 4.4 we observe that the greater the value of $|K|$: (i) the greater

Type	n	$\|K\|$	Time [s]	Gap (%)	$\|\mathbf{y}^0\|$	$\|\mathbf{x}^0\|$	$\Delta OPT\%$	$\Delta\omega\%$	#(l-LS)	#(l-LS)$_{\text{MH}}$	#(i-LS)	#BBN	#Opt
Bang	128	25	208.84	0.73	3	10	0.00	0.00	90	3	0	1548	54
		50	308.63	1.81	3	11	23.50	23.60	138	3	0	1128	42
		75	293.61	1.75	3	10	29.69	30.82	145	3	0	665	43
Phi	100	25	169.79	0.31	3	12	0.00	0.00	120	3	0	2126	61
		50	265.63	1.53	3	12	29.30	32.15	119	2	0	1872	52
		75	341.57	2.90	3	14	34.83	36.86	153	2	0	1480	39
ND	100	25	219.90	0.56	3	8	0.00	0.00	104	2	0	2661	55
		50	249.05	1.74	3	7	10.39	13.05	133	2	0	1394	47
		75	294.59	2.75	3	7	22.64	25.12	142	2	0	1105	39

TABLE 4.4: Statistics of solution characteristics and algorithmic performance for different values of $|K|$ (Groups `Bangladesh`, `Philippines` and `ND-II`)

	`Bangladesh-128`					`Philippines-100`					`ND-II-100`				
	Opt. Times		Attained Gaps			Opt. Times		Attained Gaps			Opt. Times		Attained Gaps		
$\|K\|$	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt
25	78.45	54	2.92	7.43	18	92.21	61	2.04	3.93	11	102.41	55	2.38	5.09	17
50	100.51	42	4.33	9.75	30	137.03	52	5.48	15.11	20	62.38	47	5.00	9.20	25
75	86.98	43	4.33	9.94	29	122.90	39	6.32	15.60	33	36.16	39	6.00	10.87	33

TABLE 4.5: Running times needed for optimality and attained gaps when reaching the time limit for different values of $|K|$ (Groups `Bangladesh`, `Philippines` and `ND-II`)

the average running time, (ii) the greater the average attained gap, and (iii) the fewer instances are solved to optimality. From columns #(l-LS) and *#BBN*, we observe that, compared with `Trans` instances of almost the same size, much more (l-LS) cuts are added but also much more nodes are explored. This means that, on average, fewer cuts are added per enumeration node. This can be explained by the increase of numerical instability due to the presence of coefficients with different orders of magnitude. These differences lead to weaker or non-violated cuts. Therefore, our scenario sorting strategy interrupts the cut-generation cycle and forces more branching. A similar argument applies for explaining the small amount heuristically generated cuts (column #(l-LS)$_{\text{MH}}$) and of *integer* L-shaped cuts (column #(i-LS)).

Table 4.5 reports more details regarding the algorithmic performance. The results indicate that `Dis` instances are more difficult to solve than `Trans` instances. Even if the running times for reaching optimality are still quite reasonable, the attained gaps are high (especially when considering the maximum values). The additional difficulty of these instances is explained by their more complex structure entailed by the presence of link failures (that can be very different from one scenario to another).

The relatively high average gaps, according to Table 4.5, are a consequence of the presence of a few outliers with high gaps. The performance profiles of the gaps attained for different $|K|$ are shown in the Appendix in Figures 4.15, 4.18, and 4.21 corresponding to groups `Bangladesh`, `Philippines`, and `ND-II` respectively. One can conclude that in all cases the following pattern is observed: (i) for at least 60% of the instances optimality or a very small gap is reached (regardless of the value of $|K|$); (ii) for 75-85%

of the instances a gap below 5% is attained (regardless of the value of $|K|$); (iii) for at most 5% of the instances gaps above 10% are obtained (only for $|K| = \{50, 75\}$).

The previously described instability of the attained gaps and their dependence on the instance structure is clearly depicted in the complementary charts provided in the Appendix. One observes that factors t and f (Figures 4.13, 4.16 and 4.19) have more influence on the stability of the algorithmic performance, than factors σ_3 and σ_4 (Figures 4.14, 4.17 and 4.20).

4.5. Conclusions

The UFL is a classical combinatorial optimization problem of an enormous practical and theoretical relevance. Its simplicity and versatility makes it suitable to model different problems of real-world decision making. Nonetheless, when truly implementable solutions are sought, the consideration of uncertainty is unavoidable. For the UFL under different sources of uncertainty, we applied a new recoverable robust optimization approach (RRO) that falls within the framework of 2SRO. In this new concept, a robust solution is sought such that it can be recovered (i.e., rendered feasible using a limited set of recovery actions) once the uncertainty is revealed in a second stage. For the resulting problem, RRUFL, we designed an algorithmic framework based on Benders decomposition and we included several tailored enhancements to improve its performance.

The proposed algorithm was extensively tested on more than 7500 realistic instances divided into two groups. The results show the efficacy of the algorithm to find good quality solutions within a short running time. Moreover, the results demonstrate the strong influence of the instance cost structure on both the algorithmic performance and solution characteristics. Our computational study also illustrates how robustness and recoverability are expressed in the structure of optimal solutions, and it demonstrates the benefits of RRO when compared to a RO model without recovery.

Finally, the obtained results indicate that solving the RRUFL is a not an easy task for general purpose MIP solvers. To cope with the size of realistic instances, it is inevitable to use more sophisticated decomposition techniques, like the one presented in this study.

4.6. Appendix

4.6.1 Additional Results

In our deafult runs of the proposed branch-and-cut approach we have disabled some CPLEX features (pre-processing, heuristics and general-purpose cutting planes) in order to get a better assessment of the proposed techniques. For the sake of completeness, we have performed some experiments where all CPLEX parameters are set to their default values. In Table 4.6 we report statistics on the algorithmic performance when solving instances with $n = 100$ of groups `US`, `Germany` and `ND-I` with the default CPLEX settings.

	US					Germany					ND-I				
	Opt. Times		Attained Gaps			Opt. Times		Attained Gaps			Opt. Times		Attained Gaps		
$\|K\|$	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt
25	22.85	244	0.06	0.10	12	17.16	253	0.03	0.07	3	38.64	238	0.26	1.04	18
50	41.95	250	0.07	0.32	6	32.66	253	0.03	0.04	3	40.80	251	0.06	0.10	5
75	54.15	246	0.08	0.60	10	59.74	245	0.05	0.13	11	52.36	238	0.09	0.29	18

TABLE 4.6: Running times needed for optimality and attained gaps when reaching the time limit for different values of n and $|K|$ when enabling CPLEX Heuristics, Cuts and Preprocessing ($n = 100$, Instances `US`, `Germany` and `ND-I`)

Regarding the impact of $|K|$ and n on the algorithmic performance of our branch-and-cut, further information is provided in Table 4.7. In columns *Opt. Times* we show the average running times (*Ave.*) needed to reach optimality as well as the number of instances solved to optimality (*#Opt*); in columns *Attained Gaps*, we report the average gaps of those instances that were not solved to optimality (*Ave.*), the maximum attained gap (*max*) and the number of instances that were not solved to optimality (*#Nopt*). These values are calculated considering 256 instances per row. This table further illustrates how incorporating more robustness influences the difficulty of the problem: running times and attained gaps increase while the number of instances solved to optimality decreases. Nonetheless, one observes that even for the largest instances (500 nodes) our algorithmic framework is able to provide reasonable gaps (around 2.2% on average over all instances) even for 75 scenarios.

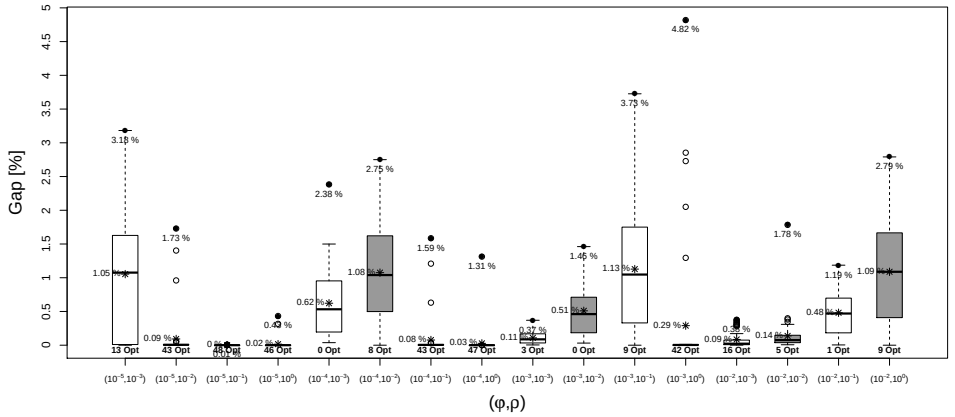
Besides the influence of n and $|K|$ on the solution structure and algorithmic performance, the coefficients (φ, ρ) and $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ also play an important role in both aspects. In Figure 4.9(a) we show the box-plots of the attained gaps for all the combinations of (φ, ρ) when solving `Germany` group with $n = 250$. Each box-plot contains information about 48 instances. The maximum and attained gaps are marked with a bold circle and an asterisk, respectively, and the number of instances solved to optimality is displayed under each box-plot. Recall that φ is a factor expressed in \$ per unit of distance per unit of demand, and ρ is expressed in \$ per inhabitant. We can observe the following: (i) The problem becomes easier (more instances can be solved

		US					Germany					ND-I				
		Opt. Times		Attained Gaps			Opt. Times		Attained Gaps			Opt. Times		Attained Gaps		
n	$\|K\|$	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt	Ave.	#Opt	Ave.	max	#Nopt
100	25	33.89	251	0.04	0.06	5	21.52	256	–	–	0	37.22	256	–	–	0
	50	63.71	252	0.59	0.88	4	37.18	252	0.01	0.01	4	31.60	256	–	–	0
	75	74.23	250	0.30	0.95	6	55.83	249	0.02	0.03	7	48.10	256	–	–	0
250	25	78.14	175	0.57	2.32	81	118.23	134	0.75	2.54	122	91.68	153	0.16	1.59	103
	50	118.48	197	0.25	2.12	59	139.28	107	0.63	2.17	149	120.91	115	0.22	1.94	141
	75	132.53	172	0.22	1.40	84	180.89	92	0.86	4.82	164	142.65	151	0.32	6.35	105
500	25	156.86	82	1.62	3.81	174	150.70	96	0.83	2.30	160	160.04	33	2.27	10.65	223
	50	290.02	11	1.38	4.66	245	256.63	43	0.70	2.59	213	268.23	12	2.11	6.19	244
	75	–	0	1.99	9.38	256	–	0	2.50	13.95	256	–	0	2.38	7.93	256

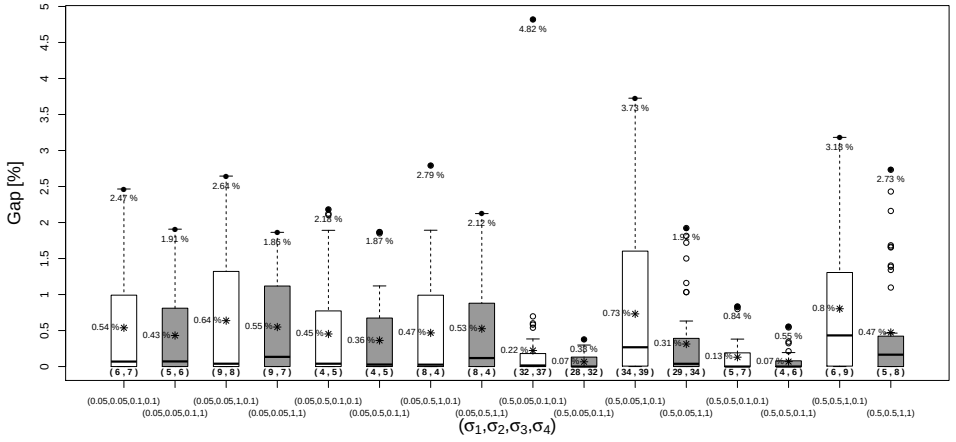
TABLE 4.7: Running times needed for optimality and attained gaps when reaching the time limit for different values of n and $|K|$ (Instances `US`, `Germany` and `ND-I`)

to optimality) when φ is considerably smaller than ρ ($10^3 - 10^5$ times smaller), that is, for those instance where the set-up costs are considerably higher than the operating costs (transportation). (ii) When $\rho < \varphi$ we have that the transportation costs are larger than the set-up costs; in these cases the attained gaps are relatively small. (iii) The problems become harder when $\frac{\varphi}{\rho} > 10^{-2}$. These three behaviors can be explained by the fact that in the easier first two cases there is not as much symmetry in the cost structure between opening and transportation costs as in in the third case (where the opening and transportation costs are of the same magnitude).

In Figure 4.9(a) we show the box-plots of the attained gaps for the 16 combinations of $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$. Average and maximum gaps are marked with bold circles and asterisk as before, and under each box-plots we provide the average value of the number of facilities open in the first stage and the number of first-stage decisions (opened facilities and defined allocations). From this graphic, one can highlight the following observations: (i) The largest first-stage components (as well as high gaps) are obtained when the factor of the re-allocation cost σ_2 is 0.05 and, especially, when $\sigma_1 = 0.5$ (the increasing factor of the second-stage allocation costs). (ii) The algorithmic performance is considerably more stable (but not better on the average) when σ_1 is 0.05 than when it is 0.5. (iii) The algorithm behaves better when the penalty factor σ_4 is 0.1 than when it is 1.0 (the difference is more clear when $\sigma_1 = 0.5$). These outcomes can be explained as follows. When the second-stage allocation costs are *expensive* (50% higher the first-stage value), but the re-allocation costs are *cheap* (only 5% higher), then an optimal or nearly optimal first-stage solution will tend to consist of several allocations which, therefore, implies that several facilities have to opened in the first-stage. On the other hand, if both costs are expensive ($\sigma_1 = \sigma_2 = 0.5$), then having a large first-stage component does not pay off. Having expensive second-stage allocation costs ($\sigma_1 = 0.5$) implies that the $\mathbf{x}^k$ variables will likely be equal to 0 (regardless of k); this immediately reduces the average computational effort of the separation problem. At the same time, this implies that a good first-stage policy is required for having a globally good solution. However, such a first-stage solution might be hard to find quickly, which



(a) Box-plots of attained Gap (%) vs. (φ, ρ)

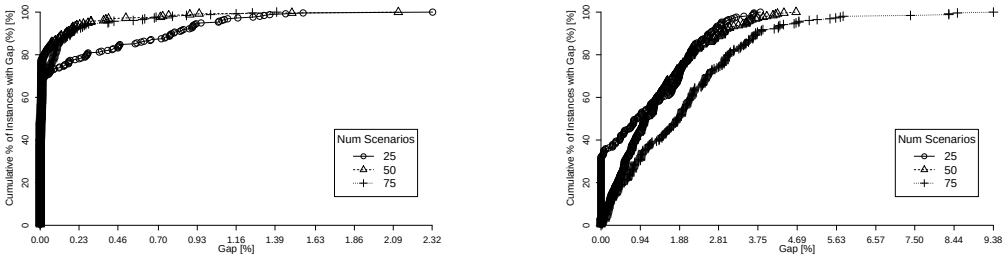


(b) Box-plots of attained Gap (%) vs. $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$

FIGURE 4.9: Influence of cost parameters (φ, ρ) and $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ on the algorithmic performance and the solution structure (Group `Germany`, $n = 250$)

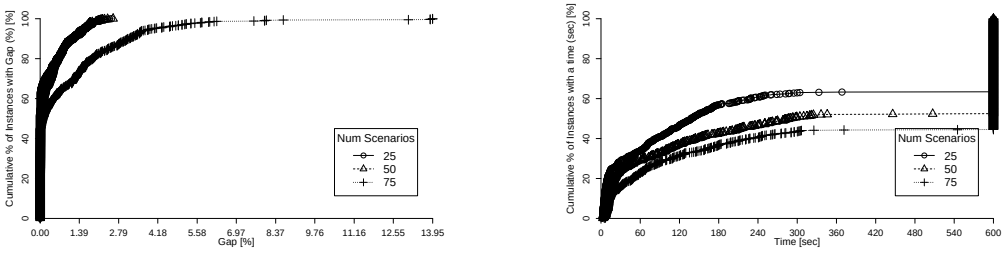
explains the large dispersion of gaps observed when $\sigma_1 = 0.5$. Likewise, if the penalty paid for having a first-stage facility in a non-available location is *expensive* ($\sigma_4 = 1.0$), then the first-stage solutions will tend to consist of as few facilities as possible (so the total second-stage penalty for the misplaced facilities is as small as possible); again, the need of a *good* first-stage policy (at least *better* than when $\sigma_4 = 0.1$) explains why the problem becomes harder, especially when a greater value of σ_1 *pushes* towards solutions with more facilities opened in the first stage.

4.6.2 Additional Performance Profiles of `Trans` Instances



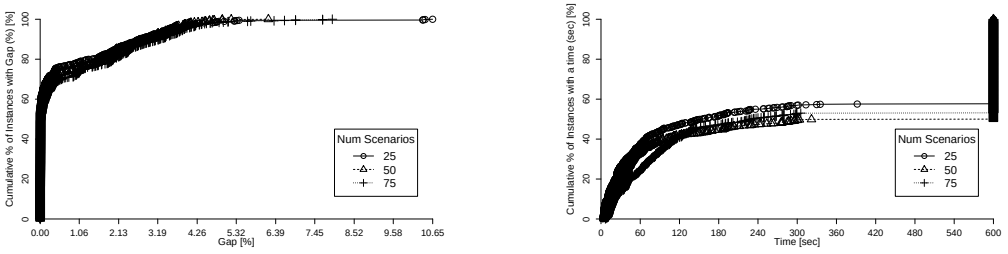
(a) `US` with $n = 250$ (b) `US` with $n = 500$

FIGURE 4.10: Performance Profile of attained gaps for different $|K|$ (Group `US` with $n \in \{250, 500\}$)



(a) Performance Profile of attained gaps (b) Performance Profile of running times

FIGURE 4.11: Performance Profile of attained gaps and running times for different $|K|$ (Group `Germany`)



(a) Performance Profile of attained gaps (b) Performance Profile of running times

FIGURE 4.12: Performance Profile of attained gaps and running times for different $|K|$ (Group `ND-I` group)

4.6.3 Detailed Results for `Bangladesh` Instances

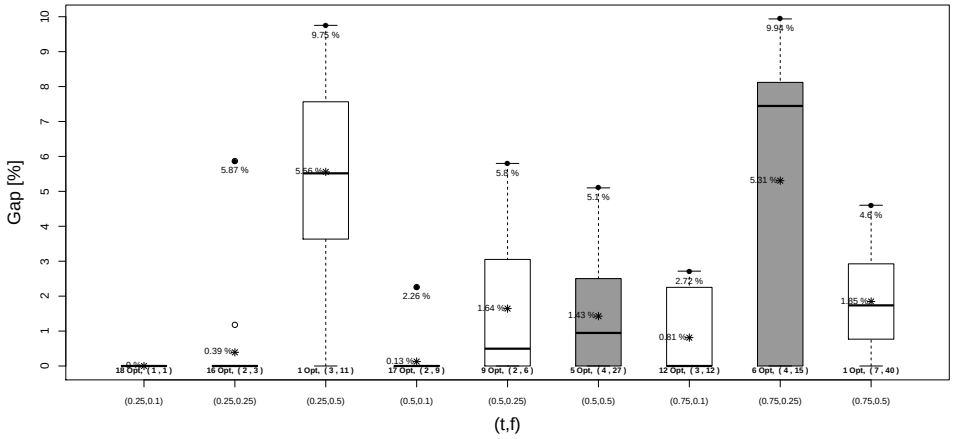


FIGURE 4.13: Box Plot of attained gaps for different combinations of (t, f) (Group `Bangladesh`, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

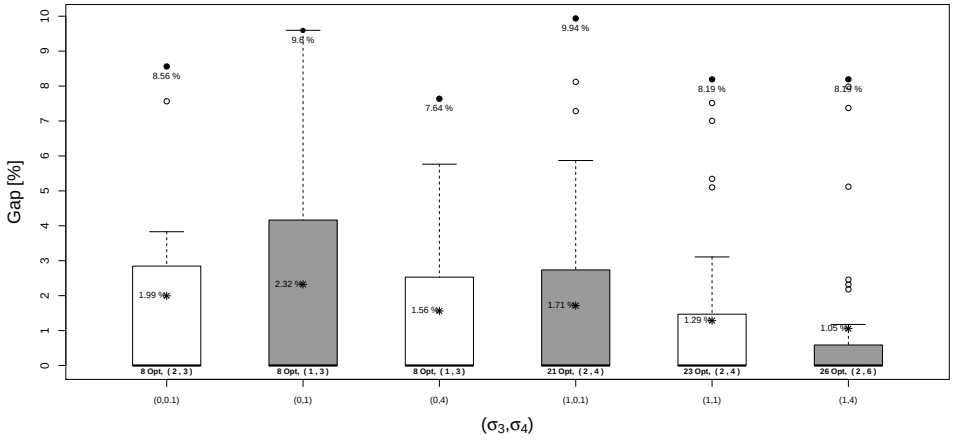


FIGURE 4.14: Box Plot of attained gaps for different combinations of (σ_3, σ_4) (Group `Bangladesh`, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

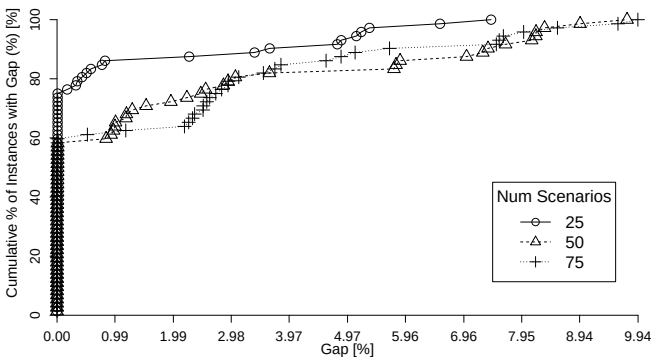


FIGURE 4.15: Performance Profile of attained gaps for different number of scenarios (Group `Bangladesh`)

4.6.4 Detailed Results for `Philippines` Instances

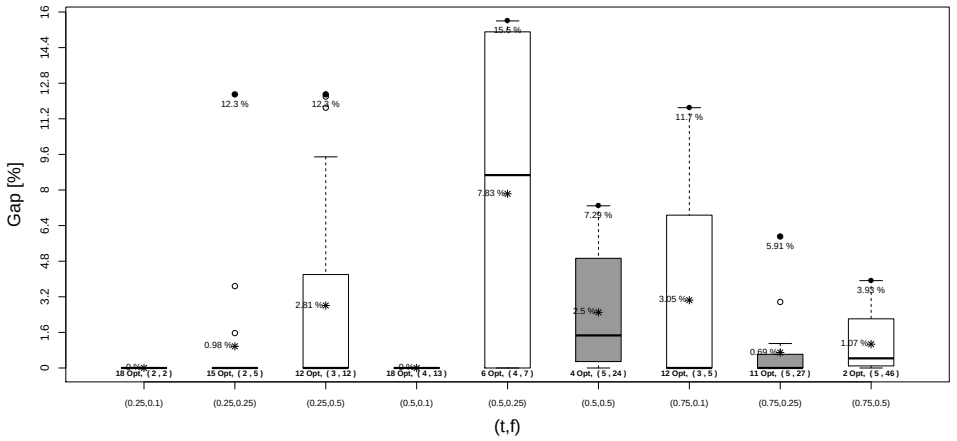


FIGURE 4.16: Box Plot of attained gaps for different combinations of (t, f) (Group `Philippines`, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

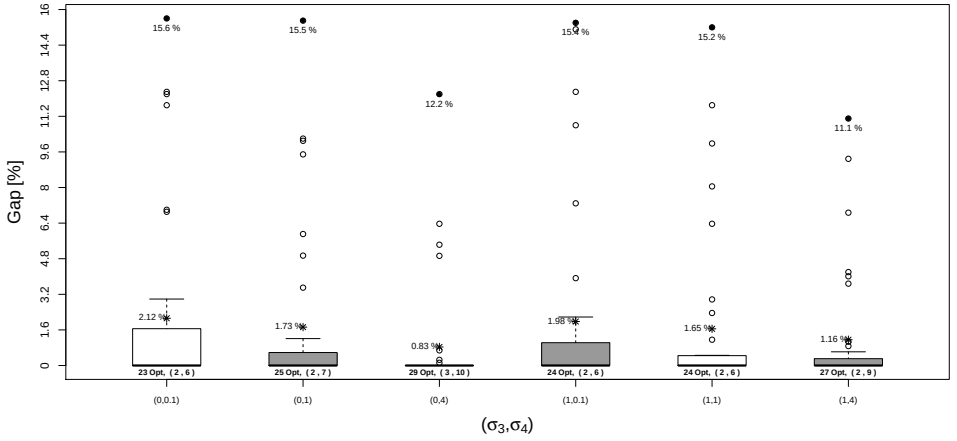


FIGURE 4.17: Box Plot of attained gaps for different combinations of (σ_3, σ_4) (Group `Philippines`, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

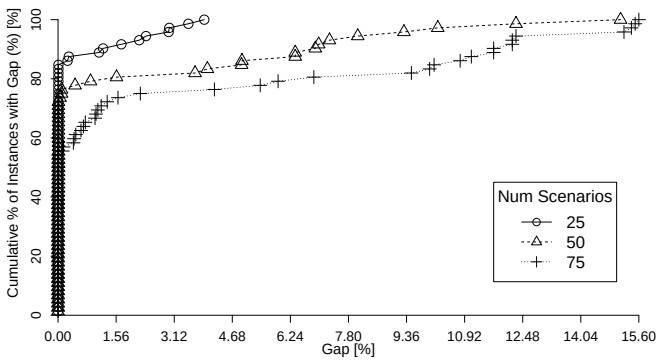


FIGURE 4.18: Performance Profile of attained gaps for different number of scenarios (Group `Philippines`)

4.6.5 Detailed Results for ND-II Instances

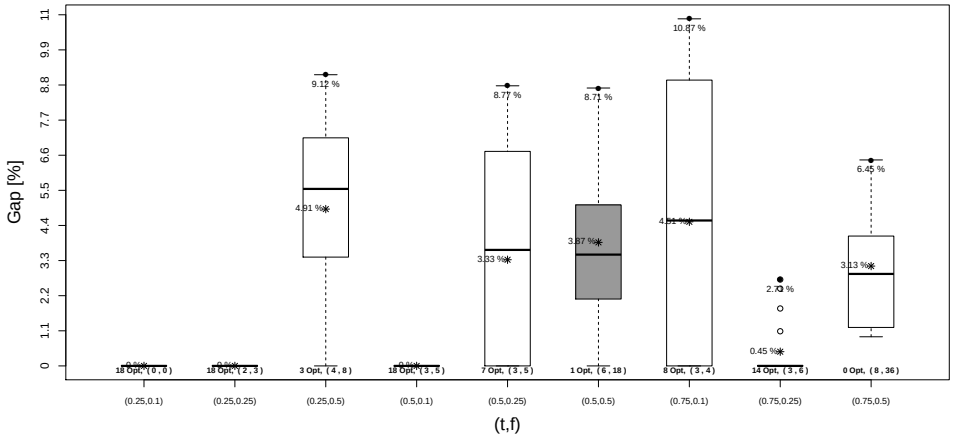


FIGURE 4.19: Box Plot of attained gaps for different combinations of (t, f) (Group ND-II, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

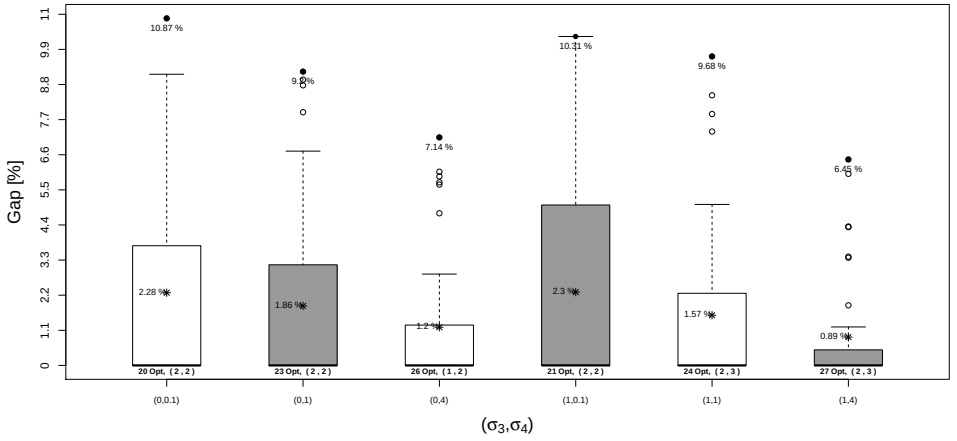


FIGURE 4.20: Box Plot of attained gaps for different combinations of (σ_3, σ_4) (Group ND-II, under each box-plot the number of optimally solved instances and the average values of $(|\mathbf{y}^0|, |\mathbf{x}^0|)$ are reported)

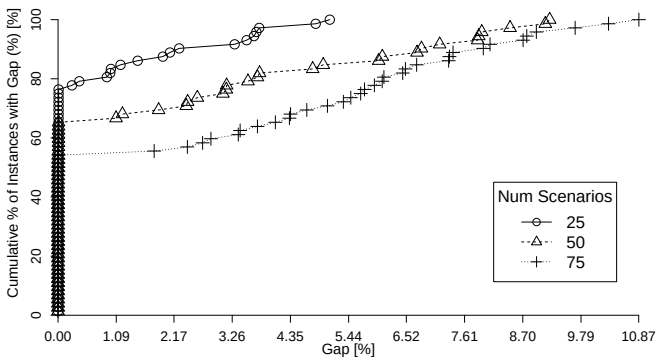


FIGURE 4.21: Performance Profile of attained gaps for different number of scenarios (Group `ND-II`)

Chapter 5

Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions

5.1. Introduction

Network design problems arise in many different areas, such as transportation and telecommunication. Recently, the class of *robust network design* problems has received increasing attention. The term robust can represent the capability of the network to cope with disruptions or to deal with different traffic scenarios in different times of the day, as is the case of our work.

In this work, we study the *single-commodity* Robust Network Design problem (RND) defined as follows. We are given an undirected graph $G = (V, E)$, a cost vector (c_e) $(e \in E)$ and an integer *balance* matrix $B = (b_i^q)$ $(i \in V,\ q = 1, \ldots, K)$. The q-th row b^q of B is called the q-th *scenario*.

For a given scenario, we call a node with nonzero balance a *terminal*. More specifically, a node i with positive balance is called a *source* and we call the balance of i its *supply*. A node with negative balance is called a *sink* and its balance is called *demand*.

Let us denote by (i, j) and (j, i) the arcs (directed from i to j and from j to i, respectively) corresponding to edge $e = \{i, j\} \in E$. In addition, let us call $f_{i,j}^q \in \mathbb{Z}_+$ the integral amount of flow that is sent along arc (i, j) from i to j in scenario q and by f^q the corresponding flow vector.

RND calls for determining integer capacities $(u_e) \in \mathbb{Z}_+^{|E|}$ $(e \in E)$ with minimal costs $c^T u$ such that, for each q $(q = 1, \ldots, K)$, there is a directed network flow f^q in G that is feasible with respect to the capacities and the balances of the q-th scenario. In particular, the flow f^q $(q = 1, \ldots, K)$ must fulfill the following constraints:

1. $f^q_{i,j} + f^q_{j,i} \leq u_e$ for all edges $e = \{i, j\} \in E$, which imposes that the sum of the flows going along every edge (in both directions) must respect the installed edge capacity, for every scenario,
2. $\sum_{\{i,j\} \in E} (f^q_{i,j} - f^q_{j,i}) = b^q_i$ for all nodes $i \in V$, which implies that the flow must satisfy the required integer balances.

An overall natural model for RND reads as follows

$$\min \sum_{\{i,j\} \in E} c_{ij} u_{ij} \tag{5.1}$$

$$\sum_{j:\{j,i\} \in E} f^q_{ji} - \sum_{j:\{i,j\} \in E} f^q_{ij} = b^q_i \qquad \forall i \in V,\ q = 1, \ldots, K \tag{5.2}$$

$$f^q_{ij} + f^q_{ji} \leq u_{ij} \qquad \forall \{i, j\} \in E,\ q = 1, \ldots, K \tag{5.3}$$

$$f^q_{ij} \geq 0 \qquad \forall \{i, j\} \in E,\ q = 1, \ldots, K \tag{5.4}$$

$$u_{ij} \in \mathbb{Z}_+, \qquad \forall \{i, j\} \in E \tag{5.5}$$

where the objective function (5.1) is to minimize the total cost of the installed capacities. Constraints (5.2) ensure flow-conservation in each scenario and impose to satisfy the required balances. Constraints 5.3 model that the capacity of an edge is at least as large as the flow it carries. Integral flows are enforced through integrality of the capacity variables, as all balances are integral [Ford and Fulkerson, 1957].

As described in [Buchheim et al., 2011], an example of a practical application of the considered problem is the following: some clients wish to download some program stored on several servers. For a client, it is not important which server he or she is downloading from, as long as the demand is satisfied. In other words, we consider servers that store identical data: examples are video on demand or large datacentre in which one mirrors his data over several locations. This is opposed to multi-commodity network design, in which point-to-point connections are considered, i.e. each client requests a specific server. In addition, we consider the robust version of the problem: at different times of the day, the demands may change (e.g. different clients show up), and the goal is to design a network that is able to route all flow in all different scenarios. In particular, we consider a finite list of demands, i.e. we sample different times of the day.

Contribution of the paper Preliminary computational investigations have been performed on classical graphs from the literature with random balances [Buchheim

et al., 2011] and on special hypercubes with $\{-1, 0, 1\}$ balances [Álvarez-Miranda et al., 2012]. The results in both papers have shown that the former instances are surprisingly easy for a general-purpose Mixed-Integer Programming (MIP) solver on the natural flow-formulation (5.1)-(5.5), while the latter instances are structurally difficult. The first contribution of the paper is in studying the complexity of some RND special cases[1] associated with the above instances and enlightening the reasons of the observed computational behavior. Second, based on the complexity results, we propose a new family of randomly generated RND instances that are computationally challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$. Third, motivated by those instances (available upon request from the authors), we propose new and general heuristic approaches that provide high-quality approximated solutions for large graphs (tests are reported for $|V|$ up to 500) in short computing times[2].

Organization of the paper Section 5.2 reviews the (vast) related literature by pointing out differences and similarities. In Section 5.3 we present the complexity results we achieved on special classes of instances, while Section 5.4 describes the proposed heuristic algorithm and its performance is reported in Section 5.5. Finally, in Section 5.6 we draw conclusions and describe ideas for future research.

5.2. Related Literature

The work on classical (i.e., non-robust) network design goes back as far as the early 1960s where it was studied by Chien [Chien, 1960] and Gomory and Hu [Gomory and Hu, 1961, 1962]. Since then, network design has evolved to a vast field of research which we cannot fully discuss in the scope of this article. We rather refer to [Chekuri, 2007] for a complete overview and restrict ourselves to a few exemplary related works that are of direct importance for us here.

The common theme of network design problems is installing optimum-cost capacities in a given network topology such that a set of traffic requests can be routed through the network. In practice, however, the traffic requests are not exactly known in advance. This can be due to measuring errors or simply because they cannot be predicted [see Ben-Tal and Nemirovski, 2000]. Here, the robustness comes in: Following an idea by Soyster [Soyster, 1973], Ben-Tal and Nemirowski [Ben-Tal and Nemirovski, 1999] coined the term of an *uncertainty set* that is added to the model and contains all possible (or likely) scenarios against which the robustness should protect. Since then, robust network design has been very actively studied. The notions of *network topology*,

[1] The RND problem is strongly NP-hard [see Sanità, 2009].

[2] A preliminary version of the heuristic approaches described here was introduced in [Álvarez-Miranda et al., 2012] where the first phase of the investigation on RND, which was the topic of the "Vigoni 2011-2012" project between the University of Köln and the University of Bologna, was summarized.

cost, *capacity*, *traffic request* and *routing* can vary – as well as the exact way in which the problem is *robustified*.

In this work, we study a *worst-case* robust model in the sense of [Ben-Tal and Nemirovski, 1999]. This means that our solutions must be feasible for all the scenarios from the uncertainty set. The uncertainty set is *finite* and *explicitly given* as part of the input (an idea that goes back to [Minoux, 1981]). We use an *undirected* graph as the network topology and allow *dynamic routing* (each scenario may be routed on different paths). Furthermore, we assume *linear costs* for the capacities and integer multiples of a unit capacity may be installed on each edge. Each node specifies its traffic request by a scalar number that gives its supply or demand and each such traffic request may be routed on an arbitrary number of paths (the routing is *splittable*) as long as each edge carries an integer amount of flow in total. Therefore, the underlying flow model is a standard *single-commodity*, splittable network flow in our case.

To the best of our knowledge, only two prior publications on this specific problem exist. The problem was first studied in [Buchheim et al., 2011]. They gave an exact branch-and-cut algorithm that solves a flow-model MIP through sophisticated general-purpose cutting planes. Lately, in [Álvarez-Miranda et al., 2012] is introduced a capacity-based MIP-model, and discussed a preliminary set of results of the biennial "Vigoni 2011-2012" between the universities of Köln and Bologna.

Atamtürk [Atamtürk, 2000] considers a variant of the non-robust single-commodity network design problem where integer multiples of a facility with fixed capacity can be installed on each arc. Ortega and Wolsey [Ortega and Wolsey, 2003] report on the performance of general MIP solvers on various network design problems and develop an exact algorithm for the single-commodity fixed-charge network design problem (all arcs may be bought at a fixed-charge and then be used at full capacity).

A close variant of single-commodity RND is the *multi-commodity* robust network design problem. Here, the traffic requests specify the amount d_{ij} of flow that should be exchanged among all pairs of nodes i and j. In particular, this defines fixed source/sink pairs – which is not the case in our problem. Also, each commodity has a single source (or sink). While this condition can also be established in the single-commodity case, it requires the use of fixed-capacity edges and therefore, our single-commodity variant is not a true special case of the multi-commodity problem. Sanità showed in her doctoral thesis [Sanità, 2009] that the multi-commodity variant is NP-hard even if there are only three scenarios, all scenarios use a unique source node and all demands are from $\{-1, 0, 1\}$. This immediately implies that the single-commodity variant is NP-hard as well. The thesis contains many further complexity results; among others Sanità gives a $O(\log |V|)$-approximation for the multi-commodity robust network design problem with unsplittable routing and shows that removing the integrality constraint from the

capacities makes the problem polynomial time solvable. This is also true for the single-commodity RND. The multi-commodity RND was also first considered as a classical (non-robust) problem [Bienstock et al., 1998].

A vast variety of problems exists in the multi-commodity case. The case where the uncertainty set is finite was studied by Minoux [Minoux, 1981], though fractional capacities are assumed in [Minoux, 1981], and in Labbé et al. [Labbé et al., 1999]. In [Duffield et al., 1999], the authors introduced the Hose uncertainty model in which the uncertainty set is defined by inflow and outflow bounds on all nodes. Ben-Ameur and Kerivin [Ben-Ameur and Kerivin, 2005] observed that this type of uncertainty set is a polytope and developed an exact approach that additionally assumes *static routing* (i.e., in all scenarios, the flow must be routed along the same subset of paths). This configuration is also known as the *Virtual Private Network* problem. An exact approach for this problem was given in [Altin et al., 2007] under the additional constraint that each commodity may only use a single path (unsplittable routing).

In the case of dynamic routing, an exact approach by Mattia [Mattia, 2013] exists. Bertsimas and Sim [Bertsimas and Sim, 2004] introduced Γ-robustness as a general model for robustification. Exact approaches that apply this type of robustness to multicommodity network design are presented in [Koster et al., 2013].

Finally, one of the most basic network design problems, the *Steiner Tree* problem, is the special case of the single-commodity robust network design problem where for each pair i, j of Steiner nodes, there exists a scenario in which exactly i and j are terminals with supply/demand of $1/-1$. If not all the Steiner node scenarios are present, the single-commodity RND instance is instead a special case of the survivable network design problem. Note, however, that, in general, RND does not consider the requirement of disjointness that is in Survivable Network Design. We refer the reader to [Kerivin and Mahjoub, 2005] for an extensive survey on this subject.

5.3. Complexity

In this section, we characterize the complexity of some RND special cases. The RND case in which we have a single scenario ($K = 1$) corresponds to a standard polynomial time minimum cost flow problem. Already for $K = 3$, RND is NP-hard (see [Sanità, 2009]): the reduction comes from the 3-Dimensional Matching Problem for the special case of RND in which there is the same source in each scenario and balances are $\{-1, 0, 1\}$.

Motivated by the computational investigations in [Buchheim et al., 2011, Álvarez-Miranda et al., 2012], in the following, we analyze some special cases:

- RND with balances different from 1 and -1;

- RND on hypercubes with all balances equal to 1, 0, or -1;
- RND on hypercubes with all balances equal to r, 0, or $-r$, with r integer and > 1.

The analysis is intended to show some classes of hard instances and some classes of easier instances. According to the results that we present in the following subsections, we are able to get a better understanding of empirical results in [Buchheim et al., 2011, Álvarez-Miranda et al., 2012], and we propose a family of randomly generated instances that are challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$.

5.3.1 All balances different from 1 and -1

Because instances defined on random graphs with random integer balances on the (randomly chosen) terminals turn out to be surprisingly easy for a general-purpose MIP solver on the natural flow-formulation (5.1)-(5.5), a natural question to ask is if this special case remains NP-hard. The following theorem answers positively through a reduction from Hamiltonian cycle [see Sanità, 2013].

In order to prove that RND, defined on graph $G = (V, E)$ ($|V| \geq 3$), with balances different from 1 and -1, is NP-hard, let us define the following RND instance I_R. We use G without modification and install a cost of 1 on each edge. We choose some arbitrary numbering of the nodes. We install $|V| - 1$ scenarios. In scenario i, only nodes 1 and $i + 1$ are terminals; the node 1 gets a balance of 2 while the node $i + 1$ has a balance of -2.

Theorem 1. A graph $G = (V, E)$ (with $|V| \geq 3$) has a Hamiltonian cycle C if and only if the described RND instance I_R has a solution with cost equal to $|V|$.

Proof. If G has a Hamiltonian cycle C, we build a feasible solution for I_R by installing a capacity of 1 on each edge of C. In each scenario i, both unique terminals 1 and $i+1$ lie on C. The node $i+1$ decomposes C into two paths P_1, P_2 from 1 to $i+1$ (one clockwise, one counterclockwise). We can route one unit of flow on P_1 and one unit of flow on P_2, satisfying the demands of scenario i. Thus, our solution for I_R is feasible and additionally, it has cost of $|C| = |V|$.

On the other hand, suppose we have a solution for I_R of cost $|V|$. By our choice of scenarios (we have a single source at node 1 and all other nodes are terminals in some scenario), each node must be connected to node 1. Therefore, any feasible solution for I_R must have a support S that induces a connected component of G containing all nodes. S must contain at least $|V| - 1$ edges, otherwise it cannot be connected. If S contains exactly $|V| - 1$ edges, a capacity of 2 must be installed on each edge in S in

order to route all demands. However, such a solution has cost of $2 \cdot |V| - 2 > |V|$ and therefore S must contain at least $|V|$ edges. If some node in $G[S]$ has a degree of 1, then we must install a capacity of 2 on its unique incident edge. By the same argument as before, the remaining nodes $|V| - 1$ nodes must be connected by at least $|V| - 1$ edges. Then again, the cost of the solution is at least $|V| - 1 + 2 > |V|$. Therefore, all nodes in $G[S]$ must have a degree of at least 2 and because we can have at most $|V|$ edges in S, each node must have exactly degree 2. Together with our observation that $G[S]$ is connected and contains all nodes, we have a Hamiltonian cycle. □

5.3.2 Hypercubes

The authors defined a structurally difficult class of instances in [Álvarez-Miranda et al., 2012], based on d-dimensional hypercubes. In the following we repeat the construction.

Definition 1. A *d-dimensional hypercube* $\mathcal{H}_d$ is the result of the following recursive construction: $\mathcal{H}_0$ is the graph that consists of a single node. For $d > 0$, $\mathcal{H}_d$ is obtained by duplicating the nodes and edges of $\mathcal{H}_{d-1}$ and connecting each node v to its copy v' with an additional edge $\{v, v'\}$.

Definition 2. We say that two nodes v, w are *diagonally opposite* on $\mathcal{H}_d$ iff the shortest path from v to w in $\mathcal{H}_d$ has maximum length, i. e., length d.

Notice that for every node v in $\mathcal{H}_d$ there is exactly one node v^o that is diagonally opposite to v. It is well-known that $\mathcal{H}_d$ has $N_d := 2^d$ nodes and $M_d := d \cdot 2^{d-1}$ edges.

We can now define a class of instances on d-dimensional hypercubes as follows. For $d \in \mathbb{Z}_+$, consider the following instance I_d of the RND problem on $\mathcal{H}_d$. Observe that $\mathcal{H}_d$ is composed of two hypercubes H^s, H^t of dimension $d - 1$. Add 2^{d-1} scenarios to $\mathcal{H}_d$. In scenario $1 \leq q \leq 2^{d-1}$, assign a supply of 1 to the q-th node v_q (in some fixed numbering) of H^s and a demand of -1 to its diagonally opposite node v_q^o which lies in H^t by our construction. Set all other balances of scenario q to zero and set the costs for each edge to 1. Figure 5.1 shows the construction.

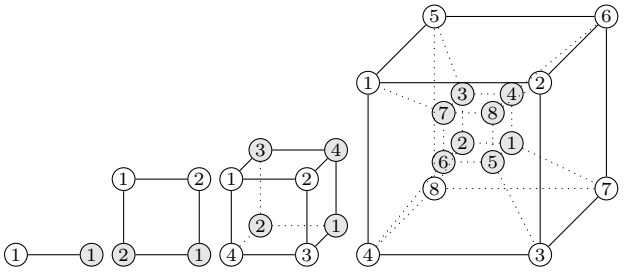


FIGURE 5.1: The hypercubes in 1, 2, 3 and 4 dimensions. Copied nodes are displayed in gray. The node numbering refers to the scenarios.

We denote the instance obtained in this way by $\mathcal{H}_d^1$. Scaling all balances in $\mathcal{H}_d^1$ by $r \in \mathbb{Z}_+$, we obtain the instance $\mathcal{H}_d^r$.

5.3.2.1 All balances equal to 1, 0, or -1

It is shown in [Álvarez-Miranda et al., 2012] that this class of instances is difficult for MIP-based solution approaches as the integrality gap (i.e., the ratio of an optimum integral solution value and an optimum fractional solution value) of $\mathcal{H}_d^1$ converges to 2 as $d \to \infty$. We refer the reader to [Álvarez-Miranda et al., 2012] for details.

5.3.2.2 All balances equal to r, 0, or $-r$, r integer and > 1

We characterize the integrality gap for $r > 1$. The optimum values for integer and fractional solutions are the same, i.e. the integrality gap is 1. We need a series of Lemmata to prove this result, stated and proven at the end of this section.

It is a well-known fact that $\mathcal{H}_d$ is hamiltonian for any $d \geq 2$ and we shall use this fact on several occasions. In particular, we can obtain a feasible integer solution for $\mathcal{H}_d^2$ by installing a capacity of 1 on each edge of a Hamiltonian cycle in $\mathcal{H}_d^2$.

Lemma 1. For any $d \geq 2$, there is a feasible integer solution for $\mathcal{H}_d^2$ with costs 2^d.

To derive the cost of this solution, recall that $\mathcal{H}_d$ has 2^d nodes. Similarly, we can state a feasible integer solution for $\mathcal{H}_d^3$.

Lemma 2. For any $d \geq 3$, there is a feasible integer solution for $\mathcal{H}_d^3$ with costs $3 \cdot 2^{d-1}$.

Proof. Let $d \geq 3$. Then $\mathcal{H}_d$ decomposes into two copies H_1, H_2 of $\mathcal{H}_{d-1}$ and a set of edges F connecting H_1 and H_2. We install a capacity of 1 on each edge in F. Since $d - 1 \geq 2$, we find Hamiltonian cycles C_1, C_2 in H_1 and H_2, respectively, and install a capacity of 1 on each edge of C_1 and of C_2.

This solution is feasible: For any scenario $i \in \{1, \ldots, q\}$, let s_i, t_i be the corresponding terminal pair. We need to route three units of flow from s_i to t_i. To do that, let $s_i' \in H_2$ and $t_i' \in H_1$ be the unique nodes such that $e_1 = \{s_i, s_i'\} \in F$ and $e_2 = \{t_i', t_i\} \in F$. Also, let $e_3 = \{u, v\} \in F$ with $u \in H_1$ and $v \in H_2$ be an arbitrary connecting edge that is different from e_1 and e_2. Mark here that F contains at least four edges because $d \geq 3$. Figure 5.2 shows an example for the situation on $\mathcal{H}_4^3$. Now, by sending one unit of flow over each of e_1, e_2, e_3, we have reduced the instance to two instances on $\mathcal{H}_{d-1}$: The first instance is defined on H_1; here, s_i has a balance of 2 and both u and t_i' have a balance of -1. However, these balances can be routed along the Hamiltonian C_1. In the second instance, which is defined on H_2, the sink t_i has a balance of -2 and both s_i' and v have a balance of 1. Again, these balances can be routed along the Hamiltonian cycle C_2.

Both C_1 and C_2 contain exactly 2^{d-1} edges, each with capacity 1. There are 2^{d-1} edges in F, all of them having capacity 1. This gives a total cost of $3 \cdot 2^{d-1}$. □

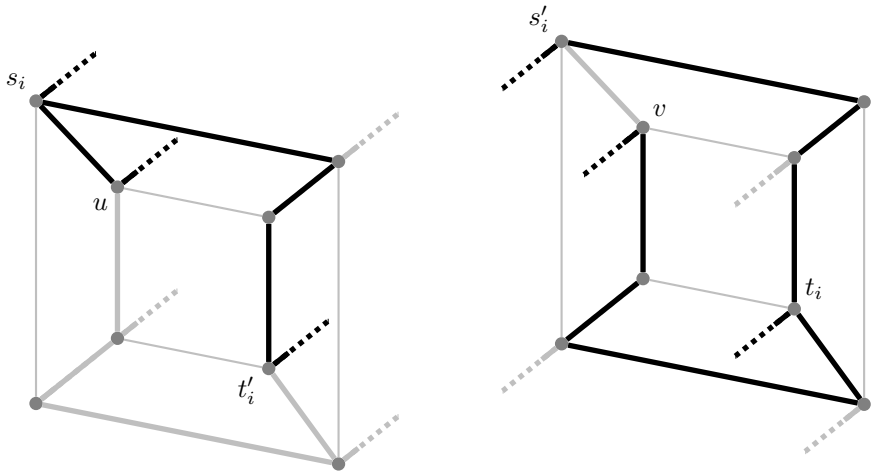


FIGURE 5.2: An example for $\mathcal{H}_4^3$.

We show next that we can construct an integer feasible solution for any $\mathcal{H}_d^r$ using the two previous ones.

Lemma 3. Let $d \geq 2$ and let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0, 1\}$. If there exists an integer feasible solution for $\mathcal{H}_d^2$ with cost at most c_2 and an integer feasible solution for $\mathcal{H}_d^3$ with cost at most c_3, then there exist an integer feasible solution for $\mathcal{H}_d^r$ with cost at most

$$m \cdot c_2 + n \cdot c_3.$$

Proof. We can decompose $\mathcal{H}_d^r$ into m copies of $\mathcal{H}_d^2$ and, if r is odd, a single copy of $\mathcal{H}_d^3$. The copies have costs of c_2 and c_3 each, respectively. For the i-th copy and $i = 1, \ldots, m + n$, we have an integer capacity vector u_i that allows for routing all scenarios. Then, $u = \sum_{i=1}^{m+n} u_i$ is an integer capacity vector that admits a routing of all scenarios of $\mathcal{H}_d^r$ and has exactly cost $mc_2 + nc_3$. □

To calculate the integrality gap for our solutions, we also need the value of an optimum fractional solution. Such a solution can be obtained by installing r/d units of capacity on each edge of $\mathcal{H}_d$ and since $\mathcal{H}_d$ has $d \cdot 2^{d-1}$ edges, this gives the following result.

Lemma 4. An optimum fractional solution for $\mathcal{H}_d^r$ has a value of $r \cdot 2^{d-1}$.

Proof of Lemma 4. If we define the set

$$\mathcal{S} := \left\{ S \subset V^d \mid S \text{ is connected and separates at least one } v_q \text{ from its partner } v_q^o \right\},$$

we can find an optimum fractional solution for $\mathcal{H}_d^r$ with the following linear program [Álvarez-Miranda et al., 2012].

$$\begin{aligned} \min \quad & \sum_{e \in E^d} u_e \\ & \sum_{e \in \delta(S)} u_e \geq r \quad \text{for all } S \in \mathcal{S} \\ & u_e \geq 0 \quad \text{for all } e \in E \end{aligned} \tag{CAP}$$

If $d = 2$, it holds that $|S| = d = 2$ for all $S \in \mathcal{S}$. Consequently, if we set $u_e = r/2$ for all $e \in E^d$, all primal constraints are satisfied with equality and the solution is optimal. If $d \geq 3$, we introduce dual variables ξ_S for all $S \in \mathcal{S}$ and obtain the following dual program:

$$\begin{aligned} \max \quad & \sum r \cdot \xi_S \\ & \sum_{\substack{S \in \mathcal{S}: \\ \{i,j\} \in S}} \xi_S \leq 1 \quad \text{for all } \{i,j\} \in E^d \\ & \xi_S \geq 0 \end{aligned} \tag{CAP*}$$

We consider the following pair of primal and dual solutions:

$$u_e := r/d \qquad \text{for all } e \in E^d \qquad\qquad \xi_S := \begin{cases} 0, & \text{if } |\delta(S)| > d \\ 1/2, & \text{if } |\delta(S)| = d \end{cases} \qquad \text{for all } S \in \mathcal{S}.$$

To prove our claim, we need to show that u and ξ are feasible and satisfy complementary slackness. Feasibility of u follows by the first part of Lemma 5: For all $S \in \mathcal{S}$, we have $|\delta(S)| \geq d$ and thus $\sum_{e \in \delta(S)} u_e = |\delta(S)|(r/d) \geq r$. Observe that by the second part of Lemma 5 equality holds if and only if $|\delta(S)| = d$. Thus, we have $(\sum_{e \in \delta(S)} u_e - r) \cdot \xi_S = 0$ for all $S \in \mathcal{S}$, yielding primal complementary slackness. To see why ξ is feasible for (CAP*) we need to show that

$$\sum_{\substack{S \in \mathcal{S}: \\ \{i,j\} \in S}} \xi_S = \sum_{\substack{S \in \mathcal{S}: \\ |\delta(S)| = d \\ \{i,j\} \in S}} \xi_S \leq 1 \qquad \text{for all } \{i,j\} \in E^d.$$

By applying Lemma 5, we can rewrite this as

$$\sum_{\substack{S \in \mathcal{S}: \\ |\delta(S)| = d \\ \{i,j\} \in S}} \xi_S = \sum_{\substack{S \in \mathcal{S}: \\ |S| = 1 \\ \{i,j\} \in S}} \xi_S = \xi_{\{i\}} + \xi_{\{j\}} = 1 \qquad \text{for all } \{i,j\} \in E^d$$

which also yields that $(\sum_{S \in \mathcal{S}: e \in S} \xi_S - 1) \cdot u_e = 0$ for all $e \in E^d$, i.e., we have dual complementary slackness. Finally, both solutions yield the desired objective value of

$\sum_{e \in E^d} r/d = d \cdot 2^{d-1} \cdot (r/d) = r \cdot 2^{d-1}$. □

The following lemma provides the missing piece for the above proof.

Lemma 5. Let $d \geq 3$. Then in $\mathcal{H}_d$, $|\delta(S)| \geq d$ for all $\emptyset \subsetneq S \subsetneq V^d$. Moreover, equality is attained if and only if $|S| = 1$ or $|S| = |V^d| - 1$.

Proof. The first part of the lemma is well-known: Saad and Schultz [Propositions 3.2 and 3.3 Saad and Schultz, 1988] proved that for any two nodes i, j of a d-dimensional hypercube, there are at least d node disjoint paths between i and j. By Menger's Theorem [Menger, 1927], this implies that $|\delta(S)| \geq d$ for all $\emptyset \subsetneq S \subsetneq V^d$. Also, if S contains a single node i, then $|\delta(S)| = |\delta(i)| = d$. It remains to show that the inequality is strict if $2 \leq |S| \leq |V^d| - 2$. Without loss of generality, we can assume that $|S| \leq \frac{1}{2}|V^d|$ since $\delta(S) = \delta(V \setminus S)$.

Now, choose an arbitrary decomposition of $\mathcal{H}_d$ into two $(d-1)$-dimensional hypercubes $H_1 = (V_1, E_1), H_2 = (V_2, E_2)$ such that neither of $S_1 := S \cap V_1$ and $S_2 := S \cap V_2$ is empty. This is possible because S contains at least two and at most $|V|/2$ nodes. It also implies that neither $S_1 = V_1$ nor $S_2 = V_2$, as otherwise S_2 or S_1 would be empty, respectively.

For $i = 1, 2$, the node set S_i defines a cut $\delta_i(S_i)$ in H_i. Since $S_i \neq \emptyset$ and $S_i \neq V_i$, we know that $|\delta_i(S_i)| \geq d - 1$, since H_i is a $(d-1)$-dimensional hypercube. Also, $\delta_1(S_1), \delta_2(S_2) \subseteq \delta(S)$ and therefore $|\delta(S)| \geq 2 \cdot (d-1) > d$ for $d \geq 3$. □

We can now prove that the optimum values for integer and fractional solutions are the same:

Theorem 2. For $d \geq 3$ and $r \geq 2$, an optimum integer solution for $\mathcal{H}_d^r$ has value $r \cdot 2^{d-1}$. In particular, the integrality gap for $\mathcal{H}_d^r$ is 1.

Proof. Let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0, 1\}$. Putting together Lemma 3 with Lemma 1 and Lemma 2, we obtain that there is an integer solution for $\mathcal{H}_d^r$ with value $c_r := m \cdot 2^d + n \cdot 3 \cdot 2^{d-1}$. If r is even, we have $n = 0$ and $m = r/2$. Therefore, $c_r = r \cdot 2^{d-1}$. On the other hand, if r is odd, we have $n = 1$ and $m = (r-3)/2$. Then, $c_r = (r-3)/2 \cdot 2^d + 3 \cdot 2^{d-1} = r \cdot 2^{d-1} - 3 \cdot 2^{d-1} + 3 \cdot 2^{d-1} = r \cdot 2^{d-1}$. By Lemma 4, this is optimal. □

5.3.3 Challenging Instances

In the previous sections we have shown that, although *computationally easy* [Buchheim et al., 2011, Álvarez-Miranda et al., 2012], RND instances defined on random graphs with random balances are difficult in theory. The explanation of this is suggested by the

fact that structurally hard instances like those defined on hypercubes and $\{-1, 0, 1\}$ balances become *theoretically easy* when balances are in r, 0, or $-r$, with r integer and $r > 1$. have an integrality gap of value one. Building on top of those results, we concentrate on instances on random graphs with balances $\{-1, 0, 1\}$ that turn out to be computationally challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$. An effective heuristic approach for this family is described and computationally evaluated in Section 5.4 and Section 5.5, respectively.

5.4. Heuristic Algorithm

In this section, we present our heuristic algorithm, which, although general, is designed having in mind the class of hard instances introduced in the precious section, i.e., random graphs with balances of $\{-1, 0, 1\}$. It consists of three phases. In the first phase (*constructive phase*, CP), the graph is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase (*neighborhood search phase*, NSP) consists of a neighborhood search on the reduced graph in order to improve the solution found: in particular, the MIP flow-formulation is solved, within a time limit, by the general-purpose MIP solver Cplex. Finally, the third phase (*proximity search phase*, PSP) consists of iteratively applying a local search (by solving a carefully constructed MIP) to further improve the solution, taking into account the original graph, and is based on the recent work [Fischetti and Monaci, 2013].

In the following, we describe the three phases in detail.

5.4.1 Constructive Phase

Initially, the graph we are dealing with is reduced, and then a solution is built. Our goal is to reduce the graph so that we are able to quickly compute a feasible solution, and we can warm start the NSP described in 5.4.2. At the same time, the graph reduction should not be too "aggressive", because the NSP should be able to improve the solution found. In other words, we need to find a trade-off between reducing the computing time and reducing the solution space. Note that, since the (nonzero) balances are 1 or -1 in our problem, it is not common to have large capacity values installed on the edges. Therefore, solutions differ mainly because of the different set of edges on which capacity is installed. Our goal is to select a "large enough" set of edges for our reduced graph.

The following steps are executed in the CP:

1. Consider the scenarios from 1 to K and multiply all balances by a given constant F;

2. construct a feasible solution for the new obtained RND instance (see Section 5.4.1.1);
3. reduce the graph by deleting all the edges that are not used in the solution found (and the nodes such that they do not have any incident edge after edge deletion) and obtain graph $\overline{G} = (\overline{V}, \overline{E})$;
4. set back the balances to 1 and -1, and construct a feasible solution (see Section 5.4.1.1) for the original RND instance on the reduce graph $\overline{G}$.

Step 1 is used to define the search space that we want to use in the NSP. Indeed, by increasing the absolute value of the balances, more edges are likely to be used in the solution computed in step 2 and they constitute the neighborhood of the solution computed in step 4. The next section describes how to compute a feasible solution for an RND instance.

5.4.1.1 Construction of a Feasible Solution

In the case of a single scenario, an algorithm for the Minimum Cost Flow (MCF) problem can be used to solve RND as follows: we define a directed graph having the same set of nodes as G and two arcs for each edge of G (one for each direction) with infinite upper bounds on the capacities. The flows that we obtain by solving the MCF problem on the defined graph determine the edge capacities, i.e., the RND solution.

In the case of K scenarios, ordered from 1 to K, in a generic scenario q we can use for free the capacities that have already been installed on the edges in scenarios $1, \ldots, q-1$. A straightforward heuristic algorithm consists of iteratively solving a MCF problem for each scenario (in the order from 1 to K), updating the capacities that can be used for free after each MCF execution. In particular, we define an auxiliary directed graph $G_{dir} = (V, A)$ having the same set of nodes of G and the set of arcs defined as follows. For each edge $e = \{i, j\} \in E$, we introduce four arcs a_1^e, a_2^e, a_3^e and a_4^e: a_1^e and a_2^e are directed from i to j, while a_3^e and a_4^e are directed from j to i. Two arcs are needed for each direction in order to take into account, in a generic scenario q, the previous scenarios $1, \ldots, q-1$: one arc has an upper bound on its capacity equal to the capacity already installed on the corresponding edge in the previous scenarios $1, \ldots, q-1$ and has zero cost; the other arc has an infinite upper bound on its capacity and has cost equal to the cost of the corresponding edge. More precisely, for each arc $a \in A$, we initialize the upper bounds UB_a on the capacities and the costs c_a as $UB_{a_1^e} := \infty$, $UB_{a_2^e} := 0$, $UB_{a_3^e} := \infty$ and $UB_{a_4^e} := 0$; $c_{a_1^e} := c_e$, $c_{a_2^e} := 0$, $c_{a_3^e} := c_e$, $c_{a_4^e} := 0$. A MCF problem is then solved for each scenario and the upper bounds are updated according to the capacities installed on each arc.

The described algorithm follows a greedy approach. It would be useful if, when solving scenario q, we could know what happens in the next scenarios $q+1, \ldots, K$ so that we

could choose accordingly the best capacity installation. In addition, a MCF solution for a generic scenario q that installs capacity on more edges (at the same cost) should be preferred: indeed, it is more likely that free capacity can be used in scenarios $q+1,\dots,K$. Therefore, a MCF solution with integer flows split over disjoint paths should be preferred with respect to a MCF solution that sends flows along a single path.

Based on these two observations, we derive an improvement of the described heuristic algorithm. We apply a preprocessing in which we divide each scenario $q=1,\dots,K$ in R sub-scenarios $g^q_1,\dots,g^q_R$, where R is an integer positive number. We consider the sub-scenarios in the order g^q_1, $(q=1,\dots,K)$, g^q_2, $(q=1,\dots,K)$, up to g^q_R, $(q=1,\dots,K)$. In this way, the generic sub-scenario g^q_l of scenario q can already take into account the partial solution computed for all the scenarios $1,\dots,K$. The balances are defined as follows: $b_v^{g^q_1}=\lfloor b^q_v/R\rfloor$, $b_v^{g^q_2}=\lfloor b^q_v/(R-1)\rfloor$, up to $b_v^{g^q_R}=b^q_v$, $v\in V$. This means that the complete MCF solution of a generic scenario q will more likely have a split integer flow over disjoint paths, because each sub-scenario might use different subsets of arcs.

The improved heuristic algorithm iteratively solves a MCF problem for each sub-scenario g^q_l $(l=1,\dots,R,\ q=1,\dots,K)$. Let us call u^{RND} the RND solution that we compute with the improved heuristic algorithm. At $h=0$, u^{RND} is initialized to be the zero vector. Let f^{h*} be the MCF solution obtained at iteration h corresponding to sub-scenario g^q_l. The flows in f^{h*} along the arcs with infinite upper bound determine the additional capacities that must be installed on the corresponding edges: for each $e=\{i,j\}\in E$, $u^{RND}_e=u^{RND}_e+f^{h*}_{a^e_1}+f^{h*}_{a^e_3}$. Note that, in each sub-scenario, there will always be an optimal solution using, for each edge, only arcs in one of the two directions: it is a single commodity flow, so we could simply do flow cancellation on cycles. In addition, the values u^{RND} are used to update the upper bounds on the capacities, before considering the following sub-scenario: $UB_{a^e_2}:=u^{RND}_e$ and $UB_{a^e_4}:=u^{RND}_e$. When all the sub-scenarios have been considered, the algorithm returns the solution found u^{RND}.

Note that in step 2 the described algorithm is used to define the reduced graph $\overline{G}$: all the edges such that $u^{RND}=0$ are deleted from G and the nodes that do not have anymore incident edges are removed as well. Step 4 is instead used to obtain a first feasible solution to our problem and is executed on the reduced graph $\overline{G}$. Let us call u^{CP} the solution obtained at the end of the constructive phase.

5.4.2 Neighborhood Search Phase

This phase consists of solving an MIP flow-formulation for RND (5.1)-(5.5) on the reduced graph $\overline{G}$ defined in the previous section.

Then, NSP explores the neighborhood of solution u^{CP} by allowing the use of different edges belonging to $\overline{G}$ and by allowing the installation of different capacities on the edges. The neighborhood is explored by solving the proposed model, initialized with u^{CP}, by Cplex within a given time limit. Let us call u^{NSP} the obtained improved solution and c^{NSP} its cost. Since we consider the reduced graph, this phase is able to quickly obtain an improved solution, as it will be seen in Section 5.5.

5.4.3 Proximity Search Phase

Recently, Fischetti and Monaci [Fischetti and Monaci, 2013] investigated the effects of replacing the objective function of a 0-1 Mixed-Integer Convex Programming problem with a "proximity" one, i.e., with minimizing the distance from a feasible solution of the problem, with the aim of enhancing the heuristic behavior of a black-box solver. In particular, they consider the Hamming distance:

$$\Delta(x, \tilde{x}) := \sum_{j \in J: \tilde{x}=0} x_j + \sum_{j \in J: \tilde{x}=1} (1 - x_j), \tag{5.6}$$

where $x_j \in \{0, 1\}$, $\forall j \in J$, and $\tilde{x}$ is a feasible solution to the considered problem. The idea consists of starting with an initial feasible solution $\tilde{x}$ with cost $f(\tilde{x})$, and iteratively searching for an improved solution by adding a cutoff constraint that imposes the cost of the improved solution to be smaller than $f(\tilde{x})$ by at least a quantity θ. The search is performed by solving with a black-box solver the new model with objective function that minimizes the Hamming distance from $\tilde{x}$, until a termination condition is reached, namely, until the first improved solution has been found. If no improved solution is found, θ is reduced. The process is then iterated by using the improved solution found as new $\tilde{x}$. The algorithm is terminated when a given time limit is reached. The method can be enhanced by providing an incumbent solution to each iteration of proximity search. This is obtained by adding an auxiliary continuous variable z which is used to keep the cutoff constraint feasible:

$$f(x) \leq f(\tilde{x}) - \theta + z \tag{5.7}$$

and has a large cost M in the objective function. In this way, $\tilde{x}$ is a (very costly) feasible solution for the MIP it defines. As soon as z becomes 0, an improved solution is found.

We apply this idea to RND, i.e. we deal with an MIP. We start with initial solution u^{NSP} and we consider the original graph G (instead of the reduced one) in order to have a higher probability of improving u^{NSP}. Since capacities assume integer (and not only binary) values, we need to modify the definition of distance presented in [Fischetti and Monaci, 2013]. Instead of expressing the distance as $|u - u^{NSP}|$, u_{ij} integer $\forall \{i, j\} \in E$, we fix upper bounds on the capacity variables, based on the values of u_{ij}^{NSP}, as follows.

For each edge $\{i,j\} \in \overline{E}$ such that $u_{ij}^{NSP} > 0$, the upper bound is set to u_{ij}^{NSP}. For all the remaining edges the upper bound is the set to be infinite. The distance is then defined as

$$\sum_{\{i,j\}\in E: u_{ij}^{NSP}=0} u_{ij} + \sum_{\{i,j\}\in E: u_{ij}^{NSP}>0} (u_{ij}^{NSP} - u_{ij}). \tag{5.8}$$

By imposing upper bounds on the capacity variable, we limit the search space and, consequently, the computing time, by using the solution found u^{NSP}. At the same time we leave the possibility of installing capacity on edges that were not used in the previous solution. Note that, by imposing upper bounds on the capacities, the proximity search becomes a heuristic method for RND. Given this distance measure definition, we iteratively solve the following proximity search model

$$\min \sum_{\{i,j\}\in E: u_{ij}^{NSP}=0} u_{ij} - \sum_{\{i,j\}\in E: u_{ij}^{NSP}>0} u_{ij} + Mz \tag{5.9}$$

$$\sum_{\{i,j\}\in E} c_{ij}u_{ij} - z\theta \leq c^{NSP} - \theta, \tag{5.10}$$

$$\sum_{j:\{j,i\}\in E} f_{ji}^q - \sum_{j:\{i,j\}\in E} f_{ij}^q = b_i^q \qquad \forall i \in V,\ q = 1,\dots,K \tag{5.11}$$

$$f_{ij}^q + f_{ji}^q \leq u_{ij} \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{5.12}$$

$$u_{ij} \leq u_{ij}^{NSP} \qquad \forall \{i,j\} \in \overline{E} : u_{ij}^{NSP} > 0 \tag{5.13}$$

$$f_{ij}^q \geq 0 \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{5.14}$$

$$u_{ij} \in \mathbb{Z}_+ \qquad \forall \{i,j\} \in E \tag{5.15}$$

$$z \in \{0,1\}, \tag{5.16}$$

where the auxiliary variable z is to guarantee feasibility of u^{NSP}. The objective function (5.9) calls for minimizing the distance from the previous solution u^{NSP} and for obtaining a solution with $z = 0$, i.e., an improved solution that respects the cutoff constraint (5.10). Constraint (5.10) imposes to obtain a reduction in the cost of the improved solution of at least θ. Constraints (5.11) and (5.12) correspond to the RND problem constraints. Constraints (5.13) impose the upper bounds on the capacity variables. Finally, constraints (5.14)-(5.16) impose variables' bounds. Note that z is defined as binary as it turned out in our computational experiments that it is very effective to impose branching priority on z, in order to quickly obtain a solution with $z = 0$.

Model (5.9)-(5.16) is solved by Cplex until the first feasible solution with $z = 0$ is obtained. In our experiments θ was set to 1. Therefore, if $z = 1$ the process is stopped. On the first feasible solution found, Cplex *polishing* (see, Rothberg [Rothberg, 2007]) is applied until the first improved solution is found. Formulation (5.9)-(5.16) is then solved again by replacing u^{NSP} with the improved solution. The proximity search

phase is executed until a given time limit is reached. When the time limit is reached, PS returns the best solution found u^{PSP}.

5.5. Computational Results

In this section, we report the computational results that we achieved on instances generated on random graphs with balances $\{-1, 0, 1\}$. Instances are generated as follows: n nodes are randomly located in a unit Euclidean square. Two nodes are connected with an edge if the Euclidean distance is less than $\alpha/\sqrt{n}$ where α is a parameter set to 2 in our generator. The edge cost for capacity installation is proportional to the Euclidean distance. For each scenario, 25%, 50% or 100% of the nodes are randomly selected to be terminals. We consider 5 or 10 scenarios.

The heuristic was developed in C language, and Cplex version 12.5 with 4 threads was used as a general purpose solver. The tests were executed on a PC 1.73 GHz, 6 GB Ram. The computing times are expressed in seconds. The algorithm CS2 by Goldberg [Goldberg, 1997] was used for solving the Minimum Cost Flow. The following parameter setting is used for the heuristic: a time limit of 300 seconds is given to NSP and a time limit of 600 seconds is given to PSP. The total time limit for the heuristic is fixed to 900 seconds, because the computation time of the CP is negligible. We fix $F = 100$, $R = 10$, $\theta = 1$ and $M = 100u^{NSP}$, based on parameter tuning.

An important feature of our heuristic algorithm is that it is robust to parameter setting, i.e. the efficacy of the algorithm is not really dependent on the specific parameter values, as long as balances are increased and scenarios are split in sub-scenarios (i.e., $F > 1$ and $R > 1$). In particular, the difference between average gaps, computed with respect to the solutions obtained by Cplex 5h, for different combinations of F and R (with $F > 1$ and $R > 1$) are negligible (below 1%). Among all combinations, the one that has the best balance between gap and standard deviation is given by $R = 10$ and $F = 100$. The other parameter used in our heuristic algorithm is θ. We choose $\theta = 1$ as a conservative value, i.e. a value that allows us to obtain good solutions on average on all the instances. In particular, we observed that, on the small instances (with 50 to 100 nodes), larger values for θ do not produce high quality solutions. When the instances get larger, a more aggressive policy (e.g. with $\theta = 100$) can give better results. We decided to keep a conservative value, in order to avoid parameter overtuning.

In Figure 5.3, we show the results obtained, with a time limit of 900 seconds, by the proposed method after each of the three phases described in Section 5.4. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented with respect to the solutions obtained after the constructive phase and we show in black the percentage improvement of the solutions

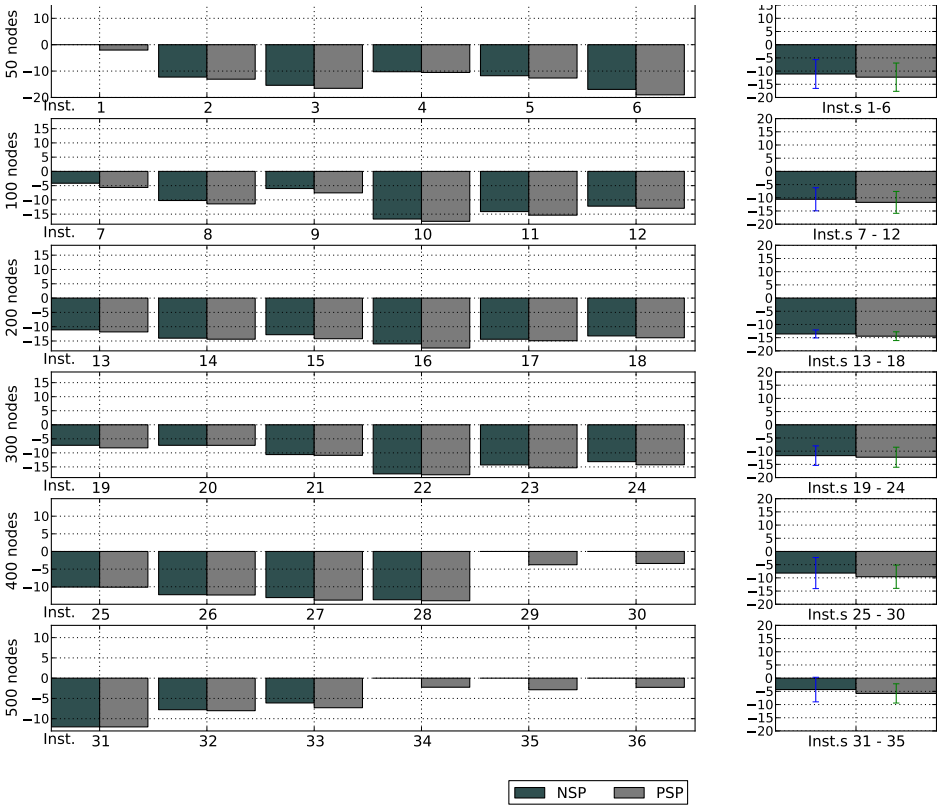


FIGURE 5.3: Comparison of the results obtained after NSP and after PSP with those obtained after CP.

obtained after the neighborhood search phase (NSP) and in gray the percentage improvement of the solutions obtained after the proximity search phase (PSP). On the right of the figure, we also show the average and standard deviation for each class of instances. As it can be seen, both the NSP and the PSP are effective in obtaining improvements for all instances but six, on which only PSP is able to improve the solution found u^{CP} after the constructive phase. The detailed results are reported in Table 5.2 in Section 5.7.

In the following, we present a comparison of the results obtained by the proposed heuristic (indicated as RND Heur.) with those obtained by Cplex applied to the MIP flow model (5.1)-(5.5) on the original graph G. In particular, we show the results obtained when Cplex is run with a time limit of five hours (Cplex 5h) in default setting, and the results obtained with Cplex in the effective heuristic configurations suggested in [Fischetti and Monaci, 2013] (Cplex Pol. 900s), i.e., solution *polishing* is applied, and the time limit is set to 900 seconds. The proposed method, Cplex 5h and Cplex Pol. 900s are initialized with the solution u^{CP} constructed as explained in Section 5.4.1.

In Table 5.1, we report the results obtained by Cplex 5h, which will be used as our benchmark for comparison. In particular, we report the data on the instances, the solution (u^{CP}) obtained at the end of the constructive phase and used for initializing

					Cplex 5h				
Inst.	n	$t\%$	K	u^{CP}	LB	UB	Gap%	BBn	Time
1	50	25	5	22904	22438	22438	0.00	1977	6
2	50	50	5	60921	52947	52952	0.01	602223	2666
3	50	100	5	79487	66334	66340	0.01	968741	4634
4	50	25	10	52835	44129	47272	6.65	464679	18000
5	50	50	10	66781	55277	57861	4.47	374536	18000
6	50	100	10	88323	70085	71526	2.01	544735	18000
7	100	25	5	39255	36741	37031	0.78	960936	18000
8	100	50	5	89264	76498	78702	2.80	430334	18000
9	100	100	5	81126	74331	74822	0.66	675507	18000
10	100	25	10	86929	67148	71192	5.68	63827	18000
11	100	50	10	115437	92265	97246	5.12	44262	18000
12	100	100	10	132233	112062	114624	2.24	76558	18000
13	200	25	5	98497	77288	85676	9.79	67461	18000
14	200	50	5	142509	113158	122062	7.29	53440	18000
15	200	100	5	169962	139302	144508	3.60	75979	18000
16	200	25	10	134999	98358	114995	14.47	11630	18000
17	200	50	10	173335	133819	148087	9.63	9406	18000
18	200	100	10	219903	175660	184992	5.04	14684	18000
19	300	25	5	92259	73805	83302	11.40	28125	18000
20	300	50	5	139954	115164	128296	10.24	22212	18000
21	300	100	5	183689	150048	162860	7.87	22284	18000
22	300	25	10	148349	103953	148349	29.93	2927	18000
23	300	50	10	201301	151200	199456	24.19	2198	18000
24	300	100	10	271340	214577	268072	19.96	2603	18000
25	400	25	5	109241	87877	98297	10.60	14881	18000
26	400	50	5	217300	175286	190328	7.90	12671	18000
27	400	100	5	291469	234266	252987	7.40	18336	18000
28	400	25	10	158033	117143	158033	25.87	1191	18000
29	400	50	10	253648	191242	253648	24.60	1239	18000
30	400	100	10	325512	255769	325512	21.43	1278	18000
31	500	25	5	106191	75197	98778	23.87	7576	18000
32	500	50	5	189269	159465	177572	10.20	10186	18000
33	500	100	5	261922	216832	241684	10.28	8584	18000
34	500	25	10	214149	153247	214149	28.44	325	18000
35	500	50	10	262379	196930	262379	24.94	315	18000
36	500	100	10	323275	249201	323275	22.91	564	18000

TABLE 5.1: Results obtained with Cplex on the MIP flow formulation in five hours of time limit.

each of the methods, the best lower bound (LB) and the best upper bound (UB) obtained by Cplex 5h, the duality gap (Gap%), the number of branch and bound nodes (BBn), and the computing time. As it can be seen from Table 5.1, the time limit is reached for all instances but the three smallest ones for which Cplex is able to prove optimality. For the remaining instances, the duality gaps are often quite large and for seven instances Cplex is not even able to improve the initial solution.

In order to measure the performance of the proposed method, we show in Figure 5.4 the comparison between the results we obtain in 900 seconds of time limit and the results obtained by Cplex 5h and by Cplex Pol. 900s. The detailed results are reported in

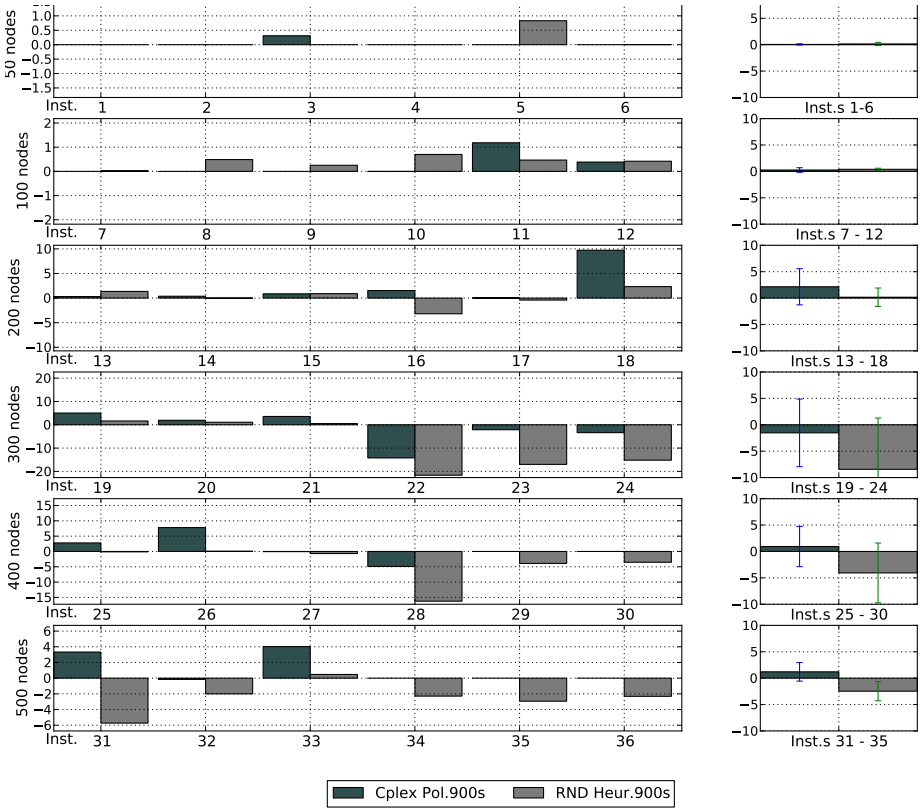


FIGURE 5.4: Comparison of the proposed heuristic RND (time limit of 15 minutes) with Cplex Pol. 900s and with Cplex 5h.

Table 5.3 in Section 5.7. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented with respect to the solutions obtained by Cplex 5h and we show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur. On the right of the figure, we also show the average and standard deviation for each class of instances.

As it is evident from Figure 5.4, the three methods obtain comparable results for instances with up to 100 nodes. However, as the instances get larger, the proposed method becomes more effective than the other ones, and it is able to improve the results obtained by the other two methods. In particular, compared to Cplex Pol. 900s that has the same time limit, the proposed method always obtains better solutions for instances with at least 300 nodes. It obtains solutions with a cost less or equal than those obtained by Cplex Pol. 900s for 27 out of 36 instances, and is at most 1.05% worse for a single instance. The improvement is significant (between 3% and more than 14%) for 14 out of 36 instances. Even compared to Cplex run for five hours, the proposed method performs on average better on instances with at least 200 nodes, especially when we have 10 scenarios. It is able to obtain better or equal solutions for 20 out of 36 instances. The average percentage improvement with respect to Cplex 5h and Cplex Pol. 900s is 2.38% and 2.90%, respectively.

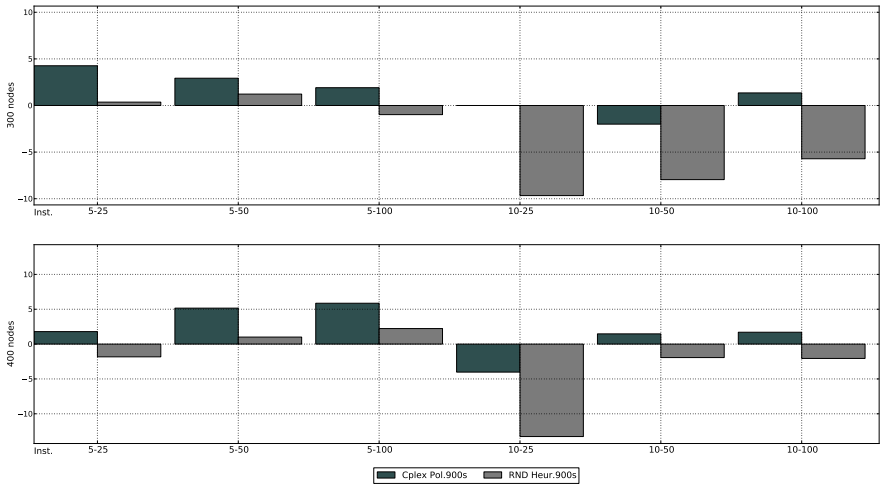


FIGURE 5.5: Comparison of the three methods on additional instances with 300 and 400 nodes.

In order to further validate the results presented in Figure 5.4, we performed extensive computational experiments on instances with $n = 300$ and $n = 400$ nodes. In particular, we considered five instances for each sub-class, defined by selecting $t \in \{25\%, 50\%, 100\%\}$ and $k \in \{5, 10\}$. This gives a total testbed of 60 instances. In Figure 5.5, we show the comparison between the three methods. The comparison is presented with respect to the solutions obtained by Cplex in five hours. We show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur in 900s. Compared to Cplex Pol. 900s, that has the same time limit, the proposed method always obtains better solutions, and, compared to Cplex 5h, performs better on all instances with 10 scenarios, confirming the effectiveness of the proposed approach.

5.6. Conclusions and Future Research

We have presented a single-commodity robust network design problem and we have shown complexity results for special classes of instances, including hypercubes. By the complexity analysis, we have shown that instances with random integer balances different from 1 and -1 are NP-hard, even if computationally easy [see Buchheim et al., 2011, Álvarez-Miranda et al., 2012]. In order to explain why, we have shown that instances defined on hypercubes with balances in $\{-r, 0, r\}$ (r integer, $r > 1$) are theoretically easy, while instances defined on hypercubes with balances in $\{-1, 0, 1\}$ are structurally hard. This has motivated us to study instances (defined on random graphs) with balances of $\{-1, 0, 1\}$. We have developed a heuristic algorithm composed of three phases. The first one reduces the instance graph and constructs a feasible

solution, the second one solves an MIP flow-formulation of the problem on the reduced graph for a given time limit, in order to improve the solution found, and the last phase applies a modified version of the recent technique of proximity search to further improve the solution. We have tested the proposed method on randomly generated instances with balances of $\{-1, 0, 1\}$, and we have compared the obtained results with those obtained by Cplex both in 5 hours (default version) or by using the *polishing* algorithm to enhance its heuristic behavior (for 900 seconds). The results show that our method is comparable with the other ones for instances with up to 100 nodes, but obtains better solutions for larger instances. Future research can be devoted to extend the proposed algorithm to the multi-commodity case. In addition, the proposed method takes into account the balances of all the scenarios, but a less conservative approach could be considered, for example, by taking into account the probability of each scenario. Other extensions could be to tackle related variants of robust network design, such as Survivable Network Design: mostly the constructive phase needs to be modified, as long as a good MIP formulation exists. Additional parameter tuning might be necessary as well.

5.7. Complementary Results

In Table 5.2, we show the results obtained by the proposed method after each of the three phases described in Section 5.4. In particular, we show the instance name (Inst.), the number n of nodes in the graph G, the percentage $t\%$ of nodes that are terminals, the number K of considered scenarios, the solution (u^{CP}) obtained at the end of the constructive phase, the solution u^{NSP} obtained after the neighborhood search phase (and the corresponding percentage improvement $Impr_{u^{CP}}\%$ with respect to u^{CP}) and the final solution u^{PSP} provided by our method by applying proximity search (and the corresponding percentage improvement $Impr_{u^{NSP}}\%$ with respect to u^{CP}). We do not report the computing times, as the time limit of 900 seconds is reached for all instances.

In Table 5.3, we report the value of the best solution obtained by each method, and, for Cplex Pol. 900s and for RND Heur., we show the percentage gap $\text{Gap}_{C^{5h}}\%$ to the best upper bound computed by Cplex 5h. In the last column, we also show the percentage gap $\text{Gap}_{C^{900s}}\%$ between the solutions obtained by RND Heur. and Cplex Pol. 900s. Finally, in the last rows of the table, we show the average (Avg.), the median (Median) and the standard deviation (StDev.) of the percentage gaps, as well as the minimum (Min) and the maximum (Max) percentage gap.

Inst.	n	$t\%$	K	u^{CP}	u^{NSP}	$Impr_{u^{CP}}\%$	u^{PSP}	$Impr_{u^{NSP}}\%$
1	50	25	5	22904	22904	0.00	22438	-2.03
2	50	50	5	60921	53443	-12.27	52952	-13.08
3	50	100	5	79487	67250	-15.39	66340	-16.54
4	50	25	10	52835	47419	-10.25	47272	-10.53
5	50	50	10	66781	58928	-11.76	58346	-12.63
6	50	100	10	88323	73352	-16.95	71530	-19.01
7	100	25	5	39255	37624	-4.15	37041	-5.64
8	100	50	5	89264	80139	-10.22	79088	-11.40
9	100	100	5	81126	76247	-6.01	75012	-7.54
10	100	25	10	86929	72399	-16.71	71694	-17.53
11	100	50	10	115437	99155	-14.10	97703	-15.36
12	100	100	10	132233	116100	-12.20	115107	-12.95
13	200	25	5	98497	87562	-11.10	86855	-11.82
14	200	50	5	142509	122543	-14.01	122032	-14.37
15	200	100	5	169962	148214	-12.80	145826	-14.20
16	200	25	10	134999	113380	-16.01	111439	-17.45
17	200	50	10	173335	148397	-14.39	147487	-14.91
18	200	100	10	219903	190824	-13.22	189406	-13.87
19	300	25	5	92259	85518	-7.31	84681	-8.21
20	300	50	5	139954	129723	-7.31	129709	-7.32
21	300	100	5	183689	164206	-10.61	163699	-10.88
22	300	25	10	148349	122424	-17.48	121953	-17.79
23	300	50	10	201301	172539	-14.29	170487	-15.31
24	300	100	10	271340	235728	-13.12	232706	-14.24
25	400	25	5	109241	98219	-10.09	98176	-10.13
26	400	50	5	217300	190677	-12.25	190492	-12.34
27	400	100	5	291469	253378	-13.07	251291	-13.78
28	400	25	10	158033	136413	-13.68	135968	-13.96
29	400	50	10	253648	253648	0.00	244109	-3.76
30	400	100	10	325512	325512	0.00	314428	-3.41
31	500	25	5	106191	93433	-12.01	93425	-12.02
32	500	50	5	189269	174540	-7.78	174082	-8.02
33	500	100	5	261922	245907	-6.11	242828	-7.29
34	500	25	10	214149	214149	0.00	209360	-2.24
35	500	50	10	262379	262379	0.00	254891	-2.85
36	500	100	10	323275	323275	0.00	315955	-2.26
Avg.						-9.91		-11.02

TABLE 5.2: Results obtained by the proposed method within 900 seconds of time limit.

				Cplex 5h	Cplex Pol. 900s		RND Heur. 900s		
Inst.	n	$t\%$	K	UB	UB	$\text{Gap}_{C^{5h}}\%$	u^{PSP}	$\text{Gap}_{C^{5h}}\%$	$\text{Gap}_{C^{900s}}\%$
1	50	25	5	22438	22438	0.00	22438	0.00	0.00
2	50	50	5	52952	52952	0.00	52952	0.00	0.00
3	50	100	5	66340	66546	0.31	66340	0.00	-0.31
4	50	25	10	47272	47272	0.00	47272	0.00	0.00
5	50	50	10	57861	57861	0.00	58346	0.83	0.83
6	50	100	10	71526	71526	0.00	71530	0.01	0.01
7	100	25	5	37031	37031	0.00	37041	0.03	0.03
8	100	50	5	78702	78702	0.00	79088	0.49	0.49
9	100	100	5	74822	74822	0.00	75012	0.25	0.25
10	100	25	10	71192	71189	0.00	71694	0.70	0.70
11	100	50	10	97246	98409	1.18	97703	0.47	-0.72
12	100	100	10	114624	115068	0.39	115107	0.42	0.03
13	200	25	5	85676	85947	0.32	86855	1.36	1.05
14	200	50	5	122062	122522	0.38	122032	-0.02	-0.40
15	200	100	5	144508	145770	0.87	145826	0.90	0.04
16	200	25	10	114995	116786	1.53	111439	-3.19	-4.80
17	200	50	10	148087	148138	0.03	147487	-0.41	-0.44
18	200	100	10	184992	204936	9.73	189406	2.33	-8.20
19	300	25	5	83302	87723	5.04	84681	1.63	-3.59
20	300	50	5	128296	130825	1.93	129709	1.09	-0.86
21	300	100	5	162860	168882	3.57	163699	0.51	-3.17
22	300	25	10	148349	129877	-14.22	121953	-21.64	-6.50
23	300	50	10	199456	195300	-2.13	170487	-16.99	-14.55
24	300	100	10	268072	259317	-3.38	232706	-15.20	-11.44
25	400	25	5	98297	101115	2.79	98176	-0.12	-2.99
26	400	50	5	190328	206445	7.81	190492	0.09	-8.37
27	400	100	5	252987	252842	-0.06	251291	-0.67	-0.62
28	400	25	10	158033	150661	-4.89	135968	-16.23	-10.81
29	400	50	10	253648	253648	0.00	244109	-3.91	-3.91
30	400	100	10	325512	325512	0.00	314428	-3.53	-3.53
31	500	25	5	98778	102182	3.33	93425	-5.73	-9.37
32	500	50	5	177572	177292	-0.16	174082	-2.00	-1.84
33	500	100	5	241684	251807	4.02	242828	0.47	-3.70
34	500	25	10	214149	214149	0.00	209360	-2.29	-2.29
35	500	50	10	262379	262379	0.00	254891	-2.94	-2.94
36	500	100	10	323275	323275	0.00	315955	-2.32	-2.32
Avg.						0.51		-2.38	-2.90
Median						0.00		0.00	-1.35
StDev.						3.67		5.75	3.96
Min						-14.22		-21.64	-14.55
Max						9.73		2.33	1.05

TABLE 5.3: Comparison of the proposed heuristic (time limit of 15 minutes) with Cplex (time limit of 5 hours or 15 minutes).

Chapter 6

On Exact Solutions for the Minmax Regret Spanning Tree Problem

6.1. Introduction

The classical (deterministic) Minimum Spanning Tree problem (MST) is a fundamental problem in combinatorial optimization, and it can be applied in several areas like logistics or telecommunications. It consists of finding a spanning tree of minimum total cost in a connected and undirected graph with non-negative edge costs. Very simple and fast greedy algorithms are able to solve large MST instances in a few seconds. We refer the reader to [Ahuja et al., 1993] for algorithms and applications of the MST.

The purpose of this work is to present exact approaches for the Minmax Regret Spanning Tree problem (MMR-ST), a generalization of the MST, where the problem is to find a feasible solution that is ϵ-optimal for any possible realization of the vector of the objective function parameters, with ϵ as small as possible. The objective function parameters are the costs of the edges of the graph and each of them is associated with a real cost interval. It is supposed that there is independence among the different cost intervals and that the uncertainty is only considered in the cost function. Problems with this type of data uncertainty are known as Interval Data Minmax Regret problems; for other types of uncertain data [see Aissi et al., 2009, Candia-Véjar et al., 2011].

It is known that many MMR combinatorial optimization problems are NP-Hard even if the corresponding deterministic version is polynomially solvable; for example, the

Shortest Path problem and the Assignment problem are NP-Hard in their MMR versions (MMR-P and MMR-A, respectively). Only for few problems, the corresponding MMR counterpart is polynomially solvable [see Candia-Véjar et al., 2011]. Several exact and heuristic approaches have been proposed for different MMR problems including MMR-ST [Yaman et al., 2001, Montemmani and Gambardella, 2005, Montemmani, 2006, Nikulin, 2008, Kasperski, 2008, Kasperski et al., 2012], MMR-P [Karasan et al., 2004, Montemmani, 2005, Kasperski, 2008], MMR-A [Pereira and Averbakh, 2011a], MMR Set Covering [Pereira and Averbakh, 2011b], MMR-TSP [Montemmani et al., 2007].

Literature Review It is known that the MMR-ST is also an NP-Hard problem [Averbakh and Lebedev, 2004, Aron and Van Hentenryck, 2004]; therefore, the existing exact algorithms are able to solve only small instances. In [Yaman et al., 2001], a compact formulation is presented and a set of instances (`Ya`) comprised by up to 25 nodes are solved by using CPLEX.

Later on, in [Aron and Van Hentenryck, 2002], a constraint programming algorithm for the MMR-ST was developed; this method outperformed the one proposed in [Yaman et al., 2001], allowing to solve to optimality instances of a new class (`He1`) with up to 40 nodes. In [Montemmani and Gambardella, 2005], a branch-and-bound algorithm was designed and applied to the `Ya` instances and to a new group of complete graph instances (`Mo`). For both classes of instances, the proposed algorithm outperformed the exact approach developed in [Aron and Van Hentenryck, 2002].

A Benders Decomposition (BD) algorithm for the MMR-ST was proposed in [Montemmani, 2006], and it was used to solve `Ya` and `Mo` instances. For the first group of instances, the BD approach solved all the instances to optimality, outperforming the results reported by [Yaman et al., 2001, Aron and Van Hentenryck, 2002, Montemmani and Gambardella, 2005]. For the second set of instances, the author considered a parameter p to control the width of the cost intervals; this allowed to conclude that the performance of the algorithm depended strongly on the value of p (the larger p was, the more difficult the optimization task became).

With respect to the heuristic approaches for the MMR-ST, three classes of algorithms are found in the literature: (i) the two "one-scenario" heuristics HM and HU, the first proposed in [Kasperski and Zieliński, 2006] (where it is shown that it has an approximation ratio 2) and the second proposed in [Montemmani et al., 2007]; (ii) a simulated annealing (SA) proposed in [Nikulin, 2008]; and (iii) a tabu search (KMZ-TS) proposed in [Kasperski et al., 2012]. The SA approach was applied to small instances of the MMR-ST (up to 30 nodes) and reasonable results were obtained; the author pointed out that the approach should also work properly for large instances due to the search scheme used in the algorithm. In [Kasperski et al., 2012], the KMZ-TS algorithm was extensively tested on different sets of instances (`Ya`, `He1`, `Mo` and others),

and it is shown that its performance is remarkably better than the one reported for the SA.

Our Contribution and Paper Outline Different algorithmic strategies for solving the MMR-ST to optimality are proposed. More precisely, a BD and a branch-and-cut approach are designed to solve benchmark instances that extend the size of instances for which an exact algorithm gets optimal solutions or small gaps. Additionally, the obtained lower bounds allow to improve the knowledge about the quality of the solutions given by the approach proposed in [Kasperski et al., 2012].

In Section 6.2 basic notation and special results for the MMR-ST are presented. Section 6.3 presents two mathematical programming formulations which will be used later. The proposed algorithms are described in detail in Section 6.4. Computational results and their analysis are presented in Section 6.5. Conclusions and future work are presented in Section 6.6.

6.2. Minmax Regret Spanning Tree (MMR-ST)

Let $G = (V, E)$, where $|V| = n$ and $|E| = m$, be an undirected connected graph with V being the set of nodes and E being the set of edges. Suppose that for every edge $e \in E$ an interval $[c_e^-, c_e^+]$ is given ($0 \leq c_e^- \leq c_e^+$). The values c_e^+ and c_e^- will be referred as the upper and lower limit, respectively, of the corresponding interval. It is assumed that the cost of edge $e \in E$ can take any value on its corresponding interval, independently of the values taken by the cost of other edges. Let Γ be defined as $\Gamma = \otimes_{e \in E} [c_e^-, c_e^+]$, i.e., the set of all possible realizations of edge costs. Thus, an element $s \in \Gamma$ is a so-called *scenario*, because it represents a particular realization of edge costs; these costs will be denoted by c_e^s for each $e \in E$. Let $\mathbf{X} \in \{0,1\}^{|E|}$ be a binary vector such that $X_e = 1$ if $e \in E$ belongs to a spanning tree of G and $X_e = 0$ otherwise. For a given scenario s and a given vector $\mathbf{X}$, the cost of the corresponding spanning tree is given by $F_s(\mathbf{X}) = \sum_{e \in E(\mathbf{X})} c_e^s$, where $E(\mathbf{X})$ corresponds to the subset of edges such that $X_e = 1 \ \forall e \in E(\mathbf{X})$ and $X_e = 0$ otherwise. The classical MST for a fixed scenario $s \in \Gamma$ is:

$$F_s^* = \min \{F_s(\mathbf{X}) \mid \mathbf{X} \in \Phi\}, \tag{MST}$$

where Φ is the set of all binary vectors associated with spanning trees of G.

For a fixed $\mathbf{X} \in \Phi$ and $s \in \Gamma$, the function $R(s, \mathbf{X}) = F_s(\mathbf{X}) - F_s^*$ is called the *regret* for $\mathbf{X}$ under scenario s. For a given $\mathbf{X} \in \Phi$, the *worst-case regret* or *robust deviation* is defined as:

$$Z(\mathbf{X}) = \max \{R(s, \mathbf{X}) \mid s \in \Gamma\}. \tag{MR}$$

The minmax regret version of the MST problem (MMR-ST) is given by:

$$Z^* = \min \{Z(\mathbf{X}) \mid \mathbf{X} \in \Phi\}. \tag{MMR}$$

In [Yaman et al., 2001], it is shown that an optimal solution for the right-hand-side of (MR) (the worst-case scenario for a given $\mathbf{X}$) holds the following property.

Theorem 6.1. *[Yaman et al., 2001] The worst-case scenario for a solution $\mathbf{X}$, $s^{\mathbf{X}}$, is obtained when the cost of the edges in $E(\mathbf{X})$ are set to the corresponding upper limits and the cost of all other edges to the corresponding lower limits, i.e., $c_e^{s^{\mathbf{X}}} = c_e^+$ $\forall e \in E(\mathbf{X})$ and $c_e^{s^{\mathbf{X}}} = c_e^-$ $\forall e \in E \setminus E(\mathbf{X})$.*

Combining the previous property with (MMR), one can derive the following formulation for the MMR-ST.

$$Z^*_{MMR} = \min \sum_{e \in E(\mathbf{X})} c_e^+ - \theta \tag{6.1}$$

$$\text{s.t.} \quad \theta \leq \sum_{e \in E(\mathbf{Y})} c_e^- + \sum_{e \in E(\mathbf{Y}) \cap E(\mathbf{X})} (c_e^+ - c_e^-), \quad \forall \mathbf{Y} \in \Phi \tag{6.2}$$

$$\theta \in \mathbb{R}_{\geq 0} \text{ and } \mathbf{X} \in \Phi. \tag{6.3}$$

Note that this formulation has an exponential number of constraints (6.2) (one per each spanning tree of G).

Let s^M be the scenario defined by $c_e^{s^M} = 1/2(c_e^- + c_e^+)$, $\forall e \in E$. An important algorithmic result for a wide class of MMR problems (including MMR-ST) was provided by [Kasperski and Zieliński, 2006] using s^M, where an approximation algorithm of ratio 2 was designed; the result reads as follows:

Lemma 1. [Kasperski and Zieliński, 2006] Let $\mathbf{X}^M$ be a minimum spanning tree for the midpoint scenario s^M. This solution holds $Z(\mathbf{X}^M) \leq 2Z^*_{MMR}$.

Thus, a solution with an approximation ratio at most 2, X^M, is obtained by simply solving a classical MST problem on G with edge costs defined by s^M. In practice, these approximate solutions have shown a good performance [see, e.g., Montemmani et al., 2007, Kasperski, 2008].

The solution obtained for the scenario s^+ defined by the upper limits of the intervals, i.e., $c_e^{s^+} = c_e^+$, has also shown an interesting performance [psee, e.g., Montemmani et al., 2007, Kasperski, 2008, Kasperski et al., 2012], although it has been proved that this solution can be arbitrarily bad [see Kasperski, 2008]. Both solutions, X^M and X^+, will be used as part of the exact approaches proposed in this work.

6.3. MIP Formulations for the MMR-ST

Notation Let $r \in V$ be an arbitrary node of V which we will denote as the *root* node. Let A be the set of arcs of the bi-directed counterpart of G, $G_A = (V, A)$, such that $A = \{(i,j),(j,i) \mid e : \{i,j\} \in E\}$; likewise, $c^-_{ij} = c^-_{ji} = c^-_e$ and $c^+_{ij} = c^+_{ji} = c^+_e$ $\forall e : \{i,j\} \in E$.

6.3.1 Formulation#1

This first formulation is based on directed cut-set inequalities. The Linear Programming relaxation of this type of formulations usually provides good quality lower bounds, since many facet-inducing inequalities can be projected out of the directed model for optimal tree problems [Grötschel et al., 1992]. Consequently, instead of looking for a spanning tree of G we look for a spanning arborescence of G_A.

Let $\mathbf{x} \in \{0,1\}^{|A|}$ be a binary vector such that $x_{ij} = 1$ if arc $(i,j) \in A$ belongs to a spanning arborescence of G_A and $x_{ij} = 0$ otherwise. We will use the following notation: A set of nodes $S \subseteq V$ $(S \neq \emptyset)$ and its complement $\overline{S} = V \backslash S$, induce two directed cuts: $\delta^+(S) = \big\{(i,j) \mid i \in S, j \in \overline{S}\big\}$ and $\delta^-(S) = \big\{(i,j) \mid i \in \overline{S}, j \in S\big\}$.

A vector $\mathbf{x}$ is associated with a directed spanning tree of G_A (spanning arborescence) rooted at r if it satisfies the following set of inequalities:

$$\sum_{(i,j)\in\delta^-(S)} x_{ij} \geq 1, \qquad \forall S \subseteq V \backslash \{r\} \;\; S \neq \emptyset \tag{6.4}$$

$$\sum_{(i,j)\in\delta^-(j)} x_{ij} = 1, \qquad \forall j \in V \setminus \{r\}. \tag{6.5}$$

Constraints (6.4), which are exponential in number, are known as *cut-set* or *connectivity* inequalities and they ensure that there is a directed path from the root r to each node $v \in V \setminus \{r\}$. This type of constraints is usually used in the context of effective branch-and-cut procedures [see, e.g., Koch and Martin, 1998]. Its separation can be performed in polynomial time using a maximum-flow algorithm on a *support graph* with arc-capacities given by the current fractional solution $\tilde{\mathbf{x}}$. Constraints (6.5), commonly referred ad *in-degree* constraints, ensure the solution to be cycle-free.

The connection between $\mathbf{X}$ and $\mathbf{x}$ variables is given by

$$X_e = x_{ij} + x_{ji}, \quad \forall e : \{i,j\} \in E. \tag{6.6}$$

Therefore, the set Φ can be defined as:

$$\Phi = \{\mathbf{X} \in \{0,1\}^{|E|} \mid (\mathbf{X}, \mathbf{x}) \text{ satisfies (6.4)-(6.6) and } \mathbf{x} \in \{0,1\}^{|A|}\}. \tag{6.7}$$

By replacing this definition of Φ in (6.1)-(6.3) we obtain a MILP formulation for the MMR-ST. In the resulting formulation, each of the constraints of type (6.2) contains an exponential number of connectivity constraints of type (6.4); furthermore, constraint (6.3) also contains an exponential number of connectivity constraints. In the following section we describe two exact methods that tackle this formulation by using Benders decomposition combined with branch-and-cut strategies.

6.3.2 Formulation#2

Instead of using cut-set inequalities to model set Φ, which are exponential in number, one can use a polynomial size representation of Φ in order to derive a compact MILP formulation for (6.1)-(6.3). Using a multi-commodity flow formulation to characterize $\mathbf{X} \in \Phi$ and the dual of a single-commodity flow formulation of the nested maximization problem embodied by θ, Yaman et. al [see Yaman et al., 2001] designed the following compact MILP reformulation of (6.1)-(6.3).

$$Z^*_{MMR} = \min \sum_{e \in E} c^+_e X_e - \sum_{k \in V, k \neq r} \left(\alpha^k_k - \alpha^k_r \right) - (n-1)\,\mu \tag{6.8}$$

$$\text{s.t.} \quad \sigma^k_{ij} \geq \alpha^k_j - \alpha^k_i, \quad \forall (i,j) \in A,\ \forall k \in V \setminus \{r\} \tag{6.9}$$

$$\sum_{k \neq r} \sigma^k_{ij} + \mu \leq c^-_{ij} + \left(c^+_{ij} + c^-_{ij} \right) X_{ij}, \quad \forall \{i,j\} \in E \tag{6.10}$$

$$\sum_{k \neq r} \sigma^k_{ji} + \mu \leq c^-_{ij} + \left(c^+_{ij} + c^-_{ij} \right) X_{ij}, \quad \forall \{i,j\} \in E \tag{6.11}$$

$$\sum_{(i,j) \in A} f_{ij} - \sum_{(i,j) \in A} f_{ji} = \begin{cases} n-1, & \text{if } i = r \\ -1, & \forall i \in V \setminus \{r\} \end{cases} \tag{6.12}$$

$$f_{ij} \leq (n-1)\, X_{ij}, \quad \forall \{i,j\} \in E \tag{6.13}$$

$$f_{ji} \leq (n-1)\, X_{ij}, \quad \forall \{i,j\} \in E \tag{6.14}$$

$$\sum_{e \in E} X_e = n - 1 \tag{6.15}$$

$$\mathbf{f}, \sigma \in \mathbb{R}^{|A|}_{\geq 0},\ \alpha \in \mathbb{R}^{|V| \times |V|}_{\geq 0},\ \mu \text{ unrestricted and } \mathbf{X} \in \{0,1\}^{|E|}. \tag{6.16}$$

Despite the fact that this MILP formulation contains a larger number of variables, any standard MILP solver can be used to solve it directly (up to a limited size). In the remainder we will refer to this model as simply "MILP", and it will be used as an algorithmic strategy in our computational experiments. Computational results using this formulation are presented in [Yaman et al., 2001, Montemmani and Gambardella, 2005, Montemmani, 2006, Kasperski et al., 2012].

6.4. Exact Approaches for the MMR-ST

6.4.1 Benders Decomposition Approaches

As said before, formulation (6.1)-(6.3) is comprised by an exponential number of constraints of type (6.2). In order to tackle the resulting model, we have designed an ad-hoc Benders decomposition approach: at each iteration, a relaxed version of (6.1)-(6.3) with only a subset of constraints (6.2) (Master Problem) is solved; using the obtained solution, the so-called Slave Problem seeks for a violated constraint (6.2) (a Benders cut) which is added to the Master for the next iteration.

The outline of the basic version of the Benders Decomposition (BBD) is given in Algorithm 6. Note that in this approach, the Master Problem (MP.1)-(MP.4) is modeled using (6.7) to characterize the feasibility of $\mathbf{X}$. Therefore, at each iteration, problem (MP.1)-(MP.4) is solved (see line 3) within a branch-and-cut framework which has as main feature the separation of cut-set inequalities (6.4). In the separation, a random node $i \in V \setminus \{r\}$ is selected, the maximum flow between r and i is calculated and the corresponding constraint (6.4) is added (if violated). In our implementation, we use *nested*, *back-flow* and *minimum cardinality cuts* to add as many violated cuts as possible through the resolution of a single auxiliary maximum-flow problem [see, e.g., Koch and Martin, 1998, Ljubić et al., 2006].

Algorithm 6 Basic Benders Decomposition (BBD)

Input: Graph $G_A = (V, A)$, intervals $\left[c_{ij}^-, c_{ij}^+\right]$ $\forall (i,j) \in A$, a root node r such that $r \in V$ and `TimeLim`.

1: Solve problem (MST) with $s = s^+$ and $s = s^M$; let Y^+ and Y^M be the corresponding solutions. Set $t := 1$, $\Phi^t := \{Y^+, Y^M\}$ and `STOP`:=`FALSE`;
2: **repeat**
3: (**Master Problem**) Solve the following problem:

$$Z_{MMR}^t = \min \sum_{e \in E} c_e^+ X_e - \theta \tag{MP.1}$$

$$\text{s.t.} \quad \theta \leq \sum_{e \in E} c_e^- Y_e + \sum_{e \in E} X_e (c_e^+ - c_e^-) Y_e, \quad \forall \mathbf{Y} \in \Phi^t \tag{MP.2}$$

$$(6.4)\text{-}(6.6) \tag{MP.3}$$

$$\mathbf{X} \in \{0,1\}^{|E|},\ \mathbf{x} \in \{0,1\}^{|A|} \text{ and } \theta \in \mathbb{R}_{\geq 0}; \tag{MP.4}$$

let $(\mathbf{X}^t, \theta^t)$ be the optimal solution of (MP.1)-(MP.4);
4: (**Slave Problem**) Solve problem (MST) with $s = s^{\mathbf{X}^t}$ (see Theorem 6.1); let Y^t be the corresponding solution;
5: **if** $\min_{t'=1,\ldots,t} Z(\mathbf{X}^{t'}) \leq Z_{MMR}^t$ or `TIME` $\geq$ `TimeLim` **then**
6: A (optimal) solution $\mathbf{X}' = \arg\min_{t'=1,\ldots,t} Z(\mathbf{X}^{t'})$ has been found, set `STOP`:=`TRUE`;
7: **else**
8: Set $\Phi^{t+1} := \Phi^t \cup \{Y^t\}$ and $t := t + 1$;
9: **until** `STOP` = `TRUE`

The cut-set inequalities found by the embedded branch-and-cut, at a given iteration of the decomposition, are kept in the model for the following iterations. This avoids to solve unnecessarily many maximum flow problems.

Clearly, the corresponding Slave Problem (line 4) is nothing but a simple MST problem with edge costs defined by the scenario induced by the optimal solution $\mathbf{X}^t$ of the corresponding Master Problem.

Note that the set of Benders cuts, derived from set Φ^t, is initialized in line 1 with two cuts, those corresponding to Y^+ and Y^M respectively. Afterward, a single Benders cut is added to the model at each iteration (see line 8).

The algorithm terminates either when an optimal solution is found or when the time limit `TimeLim` (3600 sec in our case) is reached (see line 5).

Enhancements to the Benders Decomposition

As described before, at each iteration of loop 2-9 a non-trivial Master Problem has to be solved in order to find a single Benders cut. Nevertheless, a common practice when using Benders Decomposition is including heuristic procedures in order to find additional Benders cuts. We have implemented two strategies that (heuristically) produce alternative slave solutions $\hat{\mathbf{Y}}^t$ and thus increase the pool of cuts induced by Φ^t.

The first strategy is to apply a local-search approach to the solution $\mathbf{Y}^t$ to obtain an alternative solution $\hat{\mathbf{Y}}^t$. The used procedure, a *3-OPT*-based approach, works as follows: we first identify up to 3 edges of $\mathbf{Y}^t$ that after their removal the graph remains connected; then, the cost of remaining edges is set to a uniformly randomly generated value in the interval $[c_e^-, c_e^+]$ and finally a solution $\hat{\mathbf{Y}}^t$ is calculated. If the obtained solution is better than the original one, it is used (as done in line 8) to create a new constraint. Five attempts are performed, which means that up to six new Benders cuts are added in a given iteration. We refer to this approach as Heuristic Benders Decomposition (HBD). A similar idea is used in [Pereira and Averbakh, 2011a] in the context of the MMR Assignment Problem.

The second enhancement to the BBD corresponds to an idea originally proposed by [Fischetti et al., 2010] and later used in [Pereira and Averbakh, 2011b] also in the context of MMR. Roughly speaking, the idea is the following: each of the incumbent solutions $\hat{\mathbf{X}}^t$ found while solving (MP.1)-(MP.4) is used to generate a Benders cut (induced by the corresponding $\hat{\mathbf{Y}}^t$ solution) that will be added to the model in the next iteration. This means that potentially many additional cuts are generated with a very reduced algorithmic effort. We refer to this variant as Extended Benders Decomposition (EBD).

6.4.2 Branch-and-Cut Approach

Nowadays, several MILP optimization suites provide branch-and-cut frameworks supported on the use of callbacks. Therefore, an algorithm as the BBD described before can be transformed into a pure branch-and-cut approach by the use of callbacks. This

is done by managing Benders cuts as *Optimality Cuts* that are added to the model each time a potential solution to the Master Problem is found in a given node of the branch-and-bound tree.

In the case of our problem, this approach basically works as follows. Let $(\tilde{\mathbf{X}}, \tilde{\theta})$ be a solution at a given node of the branch-and-bound tree (i.e., obtained by the resolution of an LP problem): if $\tilde{\mathbf{X}}$ is integer and does not violate any cut-set inequality (6.4) then it can be used as Master solution; if the corresponding induced solution $\tilde{\mathbf{Y}}$ (the Slave solution) is such that $Z(\tilde{\mathbf{Y}}) < \tilde{\theta}$, then a violated constraint of type (6.2) has been found and it is added to the model. We refer to this approach as B&C.

On one hand, this approach avoids the need of solving a complex MILP model at each iteration; instead, only a linear programming problem has to be solved at each node of the branch-and-bound tree and a more efficient strategy is performed to search in the space of the solutions. On the other hand, many more cuts will be added to the model (one per each new integer solution) which may increase the overall running time of the algorithm.

Enhancements to the Branch-and-Cut Approach As we described above, we add a Benders cut only when a new feasible solution is found: when the current vector $\tilde{\mathbf{X}}$ is associated with a spanning tree of G. However, even if $\tilde{\mathbf{X}}$ is fractional, one can try to find a valid Benders cut by rounding this fractional solution to a feasible one; to do so, we find a *near* integer vector $\tilde{\mathbf{X}}'$ by solving the MST on G with edge costs defined by

$$\tilde{c}_e = (c_e^- + c_e^+) \min\{1 - \tilde{x}_{ij}, 1 - \tilde{x}_{ji}\}, \ \forall e : \{i, j\} \in E; \tag{6.17}$$

using the obtained vector, $\tilde{\mathbf{X}}'$, an induced solution $\tilde{\mathbf{Y}}'$ is calculated and the corresponding Benders cut is added to the model, if violated. Moreover, using $\tilde{\mathbf{X}}$ (feasible or not) we apply a local-search (similar to that used in the HBD approach) in order to find still more violated constraints of type (MP.2) and add them to the model.

We have also embedded into the B&C a primal heuristic which attempts to provide better upper bounds using the information of the fractional solution $\tilde{\mathbf{X}}$; a feasible vector $\tilde{\mathbf{X}}'$ is calculated by solving the MST on G with edge costs defined by (6.17) and the value $\tilde{\theta}'$ is calculated correspondingly. The obtained pair $(\tilde{\mathbf{X}}', \tilde{\theta}')$ is then a candidate to be a new incumbent solution. This procedure has been also included within the Benders Decomposition approaches (BBD, HBD and EBD) as a sub-routine of the branch-and-cut that solves (MP.1)-(MP.4).

6.5. Computational Results

Benchmark Instances

In this work we used a subset of instances considered in [Kasperski et al., 2012], these are: (i) `Ya(l,u)-n` instances (`n` $\in \{10, \ldots, 100\}$ corresponds to the number of nodes and `l,u` $\subset \{10, 15, 20, 30, 40\}^2$ are parameters that control the interval cost structure), introduced in [Yaman et al., 2001]; (ii) `He1-n` and `He2-n` instances (`n` $\in \{10, \ldots, 100\}$), proposed in [Aron and Van Hentenryck, 2002]; (iii) `Mo(p)-n` instances (`n` $\in \{10, \ldots, 100\}$ and `p` $\in \{0.15, 0.50, 0.85\}$ which controls the intervals width), presented in [Montemmani and Gambardella, 2005]; and (iv) `La-n` instances (`n` $\in \{10, \ldots, 50\}$), introduced in [Averbakh and Lebedev, 2004].

These instances present very different topologies, cost interval structure and entail a different computational difficulty; for further details regarding the description of these instances we refer the reader to [Kasperski et al., 2012]. In that paper, the authors generated 10 instances for a given setting (instance type, number of nodes, cost structure, etc.). In total, we consider 122 different settings, which leads to $120 \times 10 = 1220$ instances. The size of the considered instances ranges from 10 nodes and 45 edges up to 100 nodes and 4950 edges.

Preprocessing and MILP Initialization In [Yaman et al., 2001] different polynomial time reduction and preprocessing procedures were proposed for the MMR-ST. They rely on the identification of edges that will be part of every optimal solution (*strong edges*) and edges that will never participate of any optimal solution (*weak edges*). These procedures have been used in our computations as well as in [Aron and Van Hentenryck, 2002, Montemmani and Gambardella, 2005, Montemmani, 2006, Nikulin, 2008, Kasperski et al., 2012].

For all proposed approaches we have initialized the corresponding mathematical programming models by considering the following constraints:

$$\sum_{(r,j)\in\delta^+(r)} x_{rj} \geq 1 \tag{6.18}$$

$$x_{ij} + x_{ji} \leq 1, \quad \forall (i,j) \in \delta^+(i),\ \forall i \in V \setminus \{r\}; \tag{6.19}$$

Constraint (6.18) imposes that at least one outgoing arc from r has to be active, and constraints (6.19) are the subtour elimination constraints of size 2 that avoid too frequent executions of the maximum flow procedure.

In the case of the B&C approach the best between X^M and X^+ is set as initial solution. Likewise, both solutions are used to induced two Benders cuts that are part of the initial model.

Implementation The proposed approaches were implemented using CPLEX 12.3 and Concert Technology. All CPLEX parameters were set to their default values, except the following ones: (i) CPLEX cuts were turned off, (ii) CPLEX heuristics were turned off, (iii) CPLEX preprocessing was turned off, (iv) the time limit was set to

	BBD	EBD	HBD	MILP	B&C		BBD	EBD	HBD	MILP	B&C
Ya(l-u)-10	40	40	40	40	40	He2-10	10	10	10	10	10
Ya(l-u)-20	28	30	30	40	40	He2-20	10	10	10	10	10
Ya(l-u)-30	5	9	5	40	33	He2-30	7	7	7	10	10
Ya(l-u)-40	0	0	0	17	24	He2-40	0	1	0	9	5
Ya(l-u)-50	0	0	0	0*	12	He2-50	0	0	0	8	1
Ya(l-u)-60	0	0	0	0*	1	He2-60	0	0	0	2	0
He1-10	10	10	10	10	10	Mo(p)-10	30	30	30	30	30
He1-20	10	10	10	10	10	Mo(p)-20	30	30	30	30	30
He1-30	6	6	6	0	10	Mo(p)-30	26	26	26	30	30
He1-40	0	1	0	0	7	Mo(p)-40	17	22	17	22	27
						Mo(p)-50	10	10	10	11*	17
						Mo(p)-60	10	10	10	10*	10
						Mo(p)-80	6	6	6	8*	10
						Mo(p)-100	2	4	5	9*	10

TABLE 6.1: Number of instances solved to optimality within the time limit by the different approaches.

3600 seconds. All the experiments were performed on a Intel Core i7-3610QM machine with 8 GB RAM, where each execution was run on a single processor.

6.5.1 Algorithmic Performance

For comparing all the described approaches (MILP, BBD, HBD, EBD and B&C) we have first considered those classes of instances that appear more frequently in the literature: `Mo`, `Ya`, `He1` and `He2`. In Table 6.1 we report the number of instances that are solved to optimality (within the time limit) for those groups of instances for which at least one approach is able to prove optimality for at least one instance. In this table, for a given value `n`, we report the results corresponding to: 40 `Ya` instances (given by 4 combinations of `l,u`), 10 `He1` instances, 10 `He2` instances, and 30 `Mo` instances (given by three values of `p`). Those entries of the MILP approach that are marked with an asterisk correspond to cases in which at least one problem couldn't be solved due to memory failure (CPLEX ran out-of-memory). From the reported results, the first observation is that different instances entail a quite different computational difficulty; while instances with up to 100 nodes can be solved by all approaches in the case of class `Mo`, for the class `He1` the limit for proving optimality is 40 nodes. The second observation is that among the Benders Decomposition approaches, the EBD turns out to be, in general, the most effective one. Although the MILP approach seems to be quite effective for most of the reported cases, the combinatorial explosion appears already for instances with 50 nodes (`Ya` and `Mo` instances); moreover, although not reported, the MILP approach ran out-of-memory for all instances with more than 70 nodes for the case of `Ya`, `He1` and `He2` instances.

As conclusion of the results presented in Table 6.1, one can say that only up to 40 nodes a deeper analysis comparing the running times between MILP, EBD and B&C might

	MILP				EBD				B&C			
	Min	Av.	Max	#Opt	Min	Av.	Max	#Opt	Min	Av.	Max	#Opt
He1-20	4.41	25.38	142.29	10	0.64	46.63	214.56	10	0.08	1.93	5.68	10
He1-30	-	-	-	0	104.71	1089.13	2403.56	6	4.98	114.67	578.11	10
He1-40	-	-	-	0	2220.6	2220.6	2220.6	1	189.28	1759.71	3069.07	7
He2-20	0.30	3.10	8.92	10	1.46	34.12	220.6	10	0.19	1.41	7.07	10
He2-30	4.28	38.88	145.24	10	56.78	811.69	2740.43	7	3.39	306.03	1487.95	10
He2-40	21.81	312.20	1227.26	9	600.24	600.24	600.24	1	46.3	1218.34	2154.09	5
Mo(p)-20	0.03	11.46	177.84	30	0.02	57.80	1502.53	30	0.02	0.93	16.97	30
Mo(p)-30	0.06	208.09	1699.90	30	0.03	151.03	1672.29	26	0.02	14.50	104.47	30
Mo(p)-40	0.19	564.79	3533.03	21	0.11	260.53	2270.13	19	0.03	135.94	1131.38	27
Ya(l,u)-20	2.71	25.21	78.56	40	17.58	483.77	2385.05	30	1.22	48.64	359.60	40
Ya(l,u)-30	26.41	434.75	887.80	40	332.93	1463.18	3089.89	9	10.20	433.13	3414.71	33
Ya(l,u)-40	716.18	2110.20	3552.50	17	-	-	-	0	52.07	1241.10	3487.28	24

TABLE 6.2: Statistics of the running times for instances in which optimality is attained by at least one approach for at least one, out of ten, problem.

make sense. For larger instances, either the time limit is always reached or memory problems appear. In Table 6.2 we report basic statistics about the running times of those cases for which optimality is proven; note that instances with 20, 30 and 40 nodes are considered. Columns "Min" show the minimum running time, columns "Av." the average running time, columns "Max" the maximum running time and columns "#Opt" report the number of instances for which optimality is achieved. Those cases marked with "–" correspond to the cases where no optimal solution is found within the time limit. It is clear that in most cases the EBD approach is outperformed by both the MILP and the B&C approaches. Albeit in the case of `Mo` instances the average running times reported for the EBD are smaller than those reported for the MILP approach, the corresponding number of instances solved to optimality is smaller, meaning that the effectiveness is minor. When comparing only the MILP and the B&C approaches, we see that the B&C approach outperforms the MILP approach except for the `He2` instances where the MILP approach not only solves more instances to optimality but also solves them faster. On the contrary, for `He1` and `Mo` instances, the B&C approach is clearly more effective than the MILP approach.

As a complement of the results shown in Table 6.1, in Table 6.3 we present some measures of the quality of the solutions attained for those cases in which optimality is not reached within the time limit. Columns *Gap* correspond to the average deviation attained by a given algorithm, calculated with respect to its Upper bound (UB) and Lower bound (LB); columns Gap^* correspond to the average deviation of the UB reached by the algorithm and the best UB obtained among the three approaches; columns "#NOpt" show the number of instances that are not solved to optimality. Again, it is clear that the EBD approach is beaten in most cases by the other two approaches. Nevertheless, the MILP approach is the one that presents very unusual outliers, in particular for the `He1` instances. In the case of `He1-30` instances, the average *Gap* of the MILP approach is 117%; however, the average Gap^* is 0% (note that both measures are 0% for the B&C approach): this means that the UB's are the optimal ones but the LB's are still quite far when reaching the time limit. For the

	MILP			EBD			B&C		
	Gap	*Gap**	#NOpt	*Gap*	*Gap**	#NOpt	*Gap*	*Gap**	#NOpt
`He1-30`	117%	0%	10	2%	1%	4	0%	0%	0
`He1-40`	207%	42%	10	27%	21%	9	3%	0%	3
`He2-30`	0%	0%	0	3%	1%	3	0%	0%	0
`He2-40`	0%	0%	1	21%	11%	9	4%	2%	5
`Mo(0.50)-40`	0%	0%	1	6%	6%	3	0%	0%	0
`Mo(0.85)-30`	0%	0%	0	1%	0%	4	0%	0%	0
`Mo(0.85)-40`	15%	14%	8	22%	11%	8	2%	0%	3
`Ya(10-10)-20`	0%	0%	0	3%	1%	7	0%	0%	0
`Ya(10-10)-30`	0%	0%	0	21%	17%	10	2%	0%	4
`Ya(10-10)-40`	4%	0%	8	34%	20%	10	7%	0%	8
`Ya(15-15)-20`	0%	0%	0	2%	1%	3	0%	0%	0
`Ya(15-15)-30`	0%	0%	0	17%	13%	9	1%	0%	3
`Ya(15-15)-40`	0%	0%	2	30%	24%	10	5%	5%	6
`Ya(10-20)-30`	0%	0%	0	10%	10%	8	0%	0%	0
`Ya(10-20)-40`	7%	3%	7	23%	20%	10	3%	0%	1
`Ya(15-30)-30`	0%	0%	0	6%	6%	4	0%	0%	0
`Ya(15-30)-40`	9%	5%	6	30%	28%	10	0%	0%	1

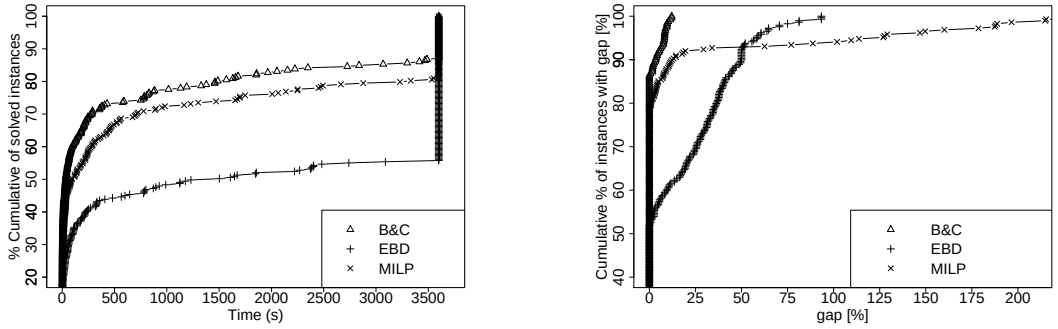
TABLE 6.3: Statistics of the gaps attained by each algorithm when reaching the time limit. *Gap* is calculated using the LB and UB obtained by each approach, while *Gap** compares the best known UB among the three approaches and the UB of the corresponding approach.

`He1-40` instances, the situation is a little bit different because the UB's are also quite far (42% in average) from the best known ones. For the remaining instances (`He2`, `Mo`, `Ya`), one can conclude that up to this size of instances (40 nodes) the MILP approach is competitive with respect to the B&C.

In Figure 6.1(a) and 6.1(b) we provide a clearer comparison between the MILP, EBD and B&C approaches for instances with 20, 30 and 40 nodes; the first graphic corresponds to the performance profile of the percentage (%) number of solved instances, while the second one corresponds to the performance profile of the attained gaps, both with respect to the running time. From these figures one can conclude that the B&C approach is the most effective one: on one hand it allows to solve to optimality more instances than the other two approaches (Figure 6.1(a)); and on the other hand, the obtained gaps (when not proving optimality) are remarkably smaller than those produced by the other two approaches (Figure 6.1(b)).

Tables 6.2 and 6.3 show that both the MILP and the B&C approach are valid strategies to tackle instances with up to 40 nodes; however, in Table 6.1 we have seen that when increasing the number of nodes the difficulty of the problems turns the MILP approach unpractical; a similar observation is outlined in [Kasperski et al., 2012].

In Table 6.4 we show the gaps attained by the B&C approach for instances with more than 40 nodes. First of all, it is clear that the instances of classes `He1` and `He2` are very hard and within 1 hour the B&C is not able to obtain reasonable gaps. The



(a) Performance profile of the running times.

(b) Performance profile of the attained gaps.

FIGURE 6.1: Performance profile of the running times (a) and attained gaps (b) comparing MILP, EBD and B&C approaches. All instances of classes `Ya`, `He1`, `He2`, `Mo` (20, 30 and 40 nodes)

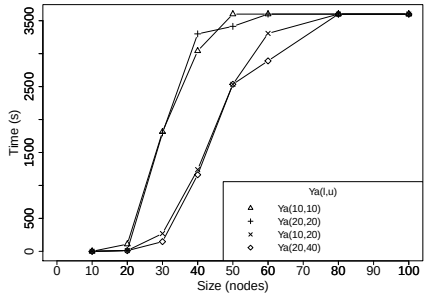
practical difficulty of these instances can be explained by fact the that these are two-level networks, where each level is comprised by "clusters" of `Ya(10,10)-5` instances; this particular topology entails higher efforts due to the presence of many symmetries among the feasible solutions. For the remaining instances, the algorithmic performance is strongly influenced by the cost structure, i.e., parameters `l` and `u` in the case of `Ya` instances, and parameter `p` in the case of `Mo` instances. For `Ya` instances one can see that when $\texttt{l} = \texttt{u}$ the problem is harder than when $\texttt{l} < \texttt{u}$. In the first case, the produced intervals are very similar among each other, which increases the symmetry among the solutions; while in the second case, the intervals are more diverse which allows to quickly detect sub-optimality during the exploration of the search tree and so to reduce the computational effort. Likewise, for `Mo` instances we see that the larger the value of `p` the harder the instances become; this behavior is explained by noticing that parameter `p` controls the interval width, which means that an increase of its value corresponds to an increase of the level of uncertainty in the corresponding instance and therefore of the difficulty of the problem [see, e.g., Montemmani and Gambardella, 2005, Montemmani, 2006, Kasperski et al., 2012].

Figures 6.2(a) and 6.2(b) complement the analysis about the influence of the interval cost structure on the algorithmic performance. From Figure 6.2(a) we can see how the difficulty of the problem changes between those instances where $\texttt{l} = \texttt{u}$ and those where $\texttt{l} < \texttt{u}$. Likewise, from Figure 6.2(b) we see that increasing the value `p` has as a consequence a decrease of the algorithm effectiveness. In both cases, the combinatorial explosion appears in the range of 40-60 nodes.

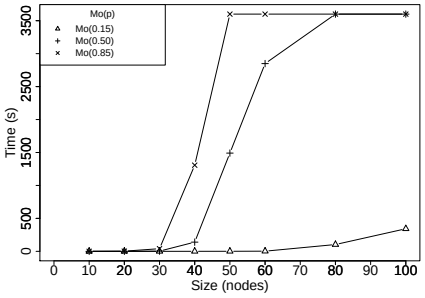
In the analysis presented so far we have excluded `La` instances which are the hardest ones from the computational point of view [see Kasperski et al., 2012]. These instances are comprised by three layers of nodes and all cost intervals are $[0, 1]$. In Table 6.5 we summarize the average gaps obtained by the B&C approach for different values of n. We can see that already from 30 nodes the gaps attained within 3600 seconds are

	#Opt	Min	Av.	Max		#Opt	Min	Av.	Max
Ya(10-10)-50	0	4%	9%	13%	Ya(10-20)-50	5	0%	2%	5%
Ya(10-10)-60	0	9%	11%	14%	Ya(10-20)-60	1	0%	5%	10%
Ya(10-10)-80	0	8%	10%	13%	Ya(10-20)-80	0	4%	6%	8%
Ya(10-10)-100	0	7%	9%	11%	Ya(10-20)-100	0	4%	6%	7%
Ya(20-20)-50	1	0%	9%	16%	Ya(15-15)-50	0	7%	9%	13%
Ya(20-20)-60	0	9%	11%	15%	Ya(15-15)-60	0	2%	10%	15%
Ya(20-20)-80	0	7%	11%	13%	Ya(15-15)-80	0	7%	10%	14%
Ya(20-20)-100	0	8%	10%	11%	Ya(15-15)-100	0	8%	10%	12%
Ya(15-30)-50	7	0%	1%	5%	Ya(20-40)-50	5	0%	2%	5%
Ya(15-30)-60	0	1%	4%	8%	Ya(20-40)-60	3	0%	4%	7%
Ya(15-30)-80	0	2%	5%	8%	Ya(20-40)-80	0	4%	6%	9%
Ya(15-30)-100	0	4%	6%	7%	Ya(20-40)-100	0	3%	6%	7%
He2-50	1	0%	10%	23%	He1-50	0	4%	11%	17%
He2-60	0	10%	17%	23%	He1-60	0	4 %	16%	23%
He2-80	0	22%	26%	30%	He1-80	0	19%	22%	29%
He2-100	0	25%	29%	31%	He1-100	0	24%	29%	32%
Mo(0.85)-50	0	2%	7%	12%	Mo(0.5)-50	7	0%	1%	8%
Mo(0.85)-60	0	7%	12%	17%	Mo(0.5)-60	5	0%	3%	13%
Mo(0.85)-80	0	12%	18%	21%	Mo(0.5)-80	0	15%	19%	24%
Mo(0.85)-100	0	17%	20%	23%	Mo(0.5)-100	0	17%	22%	27%

TABLE 6.4: Gaps attained by the B&C approach for larger instances within the time limit.



(a) Running times v/s Instance size (Ya)



(b) Running times v/s Instance size (Mo)

FIGURE 6.2: Impact of the instance size and the cost structure on the algorithmic performance in classes Ya and Mo.

quite large and the proposed approach seems to be impractical for larger instances. Again, it seems that the remarked presence of symmetries, due to the topology and the intervals structure, is the responsable for the practical difficulty of the resulting problems.

6.5.2 Comparing the B&C and the KMZ-TS Approach

The best known upper bounds for the instances considered in this work have been provided by the algorithm KMZ-TS proposed in [Kasperski et al., 2012], where a

	#Opt	Min	Av.	Max
La-10	10	0%	0%	0%
La-20	1	0%	8%	11%
La-30	0	12%	15%	21%
La-40	0	20%	24%	27%
La-50	0	25%	28%	30%

TABLE 6.5: Gaps attained by the B&C approach for La instances within the time limit.

sophisticated Tabu Search is designed and extensively tested. In Table 6.6 we present a detailed comparison between our B&C approach and the KMZ-TS algorithm; in columns Gap we report the average gaps attained by our B&C and in columns Gap_{KMZ} the average gaps obtained by [Kasperski et al., 2012]. Both average gaps are computed with respect to the LB provided by the B&C approach. Two interesting remarks can be pointed out from the reported values. First, the values Gap and Gap_{KMZ} are quite similar, which means that the upper bounds calculated by both approaches are quite similar. Second, the values in the Gap_{KMZ} column are usually smaller than those in the Gap column, meaning that the upper bounds provided by the KMZ-TS approach are better than the ones obtained by the B&C approach. These two remarks allow the reader to understand better the real quality of the proposed approaches: on one side, it is clearer now that the algorithm proposed in [Kasperski et al., 2012] is able to find very good, or even optimal, solutions for an important portion of the instances; and on the other side, we can see that although our approach is not "UB oriented", as the KMZ-TS heuristic, it is still able to attain good UB due to an effective exploration of the polyhedron through the improvements of the lower bounds.

6.6. Conclusions and Future Work

Different exact approaches have been presented for getting exact solutions for the Minmax Regret Spanning Tree problem, a generalization of the known Minimum Spanning Tree. It was shown that the branch-and-cut (B&C) approach outperforms previously proposed approaches for minmax regret optimization combinatorial problems. A broad set of benchmark instances was used for studying the performance of the algorithms.

Two important conclusions can be established after the extensive computational experience: (i) The B&C approach was able to extend the limits for which an algorithm gets optimal solutions; it achieved optimal solutions for several instances with 40 nodes or more, while the traditional approaches were not able to obtain these results due to time constraints or memory failure. (ii) The B&C approach reached relatively small gaps for all the instances where it was not able to prove optimality. In particular, this fact also allows to prove the good quality of the feasible solutions calculated by the heuristic presented in [Kasperski et al., 2012].

	Gap	Gap_{KMZ}		Gap	Gap_{KMZ}
Yaman(10,10)-10	0.00%	0.00%	He1-10	0.00%	0.00%
Yaman(10,10)-20	0.01%	0.01%	He1-20	0.00%	0.00%
Yaman(10,10)-30	1.95%	1.94%	He1-30	0.01%	0.01%
Yaman(10,10)-40	6.75%	6.71%	He1-40	2.68%	2.64%
Yaman(10,10)-50	8.79%	8.64%	He1-50	10.63%	10.16%
Yaman(10,10)-60	11.02%	10.86%	He1-60	15.92%	15.28%
Yaman(10,10)-80	10.39%	10.16%	He1-80	22.49%	21.41%
Yaman(10,10)-100	9.48%	9.37%	He1-100	28.95%	27.29%
Yaman(15,15)-10	0.00%	0.00%	He2-10	0.00%	0.00%
Yaman(15,15)-20	0.01%	0.01%	He2-20	0.00%	0.00%
Yaman(15,15)-30	1.17%	1.16%	He2-30	0.01%	0.01%
Yaman(15,15)-40	4.29%	4.21%	He2-40	3.82%	3.77%
Yaman(15,15)-50	9.47%	9.33%	He2-50	10.31%	10.12%
Yaman(15,15)-60	9.73%	9.48%	He2-60	15.93%	15.71%
Yaman(15,15)-80	10.38%	10.10%	He2-80	24.95%	24.30%
Yaman(15,15)-100	9.81%	9.65%	He2-100	27.25%	24.96%
Yaman(20,20)-10	0.00%	0.00%	Mon(0.15)-10	0.00%	0.00%
Yaman(20,20)-20	0.00%	0.00%	Mon(0.15)-20	0.00%	0.00%
Yaman(20,20)-30	1.58%	1.55%	Mon(0.15)-30	0.00%	0.00%
Yaman(20,20)-40	6.25%	6.04%	Mon(0.15)-40	0.00%	0.00%
Yaman(20,20)-50	9.37%	9.14%	Mon(0.15)-50	0.00%	0.00%
Yaman(20,20)-60	11.26%	11.00%	Mon(0.15)-60	0.00%	0.00%
Yaman(20,20)-80	11.45%	11.23%	Mon(0.15)-80	0.00%	0.00%
Yaman(20,20)-100	10.09%	9.93%	Mon(0.15)-100	0.00%	0.00%
Yaman(10,20)-10	0.00%	0.00%	Mon(0.50)-10	0.00%	0.00%
Yaman(10,20)-20	0.00%	0.00%	Mon(0.50)-20	0.00%	0.00%
Yaman(10,20)-30	0.01%	0.01%	Mon(0.50)-30	0.00%	0.00%
Yaman(10,20)-40	0.07%	0.07%	Mon(0.50)-40	0.01%	0.01%
Yaman(10,20)-50	1.95%	1.92%	Mon(0.50)-50	1.49%	1.43%
Yaman(10,20)-60	4.91%	4.83%	Mon(0.50)-60	3.36%	3.08%
Yaman(10,20)-80	5.77%	5.70%	Mon(0.50)-80	19.07%	17.16%
Yaman(10,20)-100	5.55%	5.51%	Mon(0.50)-100	22.46%	19.99%
Yaman(15,30)-10	0.00%	0.00%	Mon(0.85)-10	0.00%	0.00%
Yaman(15,30)-20	0.00%	0.00%	Mon(0.85)-20	0.00%	0.00%
Yaman(15,30)-30	0.01%	0.01%	Mon(0.85)-30	0.01%	0.01%
Yaman(15,30)-40	0.36%	0.36%	Mon(0.85)-40	1.68%	1.68%
Yaman(15,30)-50	0.98%	0.98%	Mon(0.85)-50	6.69%	6.63%
Yaman(15,30)-60	4.48%	4.43%	Mon(0.85)-60	12.22%	11.85%
Yaman(15,30)-80	5.04%	5.00%	Mon(0.85)-80	17.50%	16.61%
Yaman(15,30)-100	5.97%	5.94%	Mon(0.85)-100	19.75%	18.74%
Yaman(20,40)-10	0.00%	0.00%	Yaman(20,40)-50	1.56%	1.56%
Yaman(20,40)-20	0.00%	0.00%	Yaman(20,40)-60	2.64%	2.62%
Yaman(20,40)-30	0.01%	0.01%	Yaman(20,40)-80	5.99%	5.93%
Yaman(20,40)-40	0.81%	0.81%	Yaman(20,40)-100	5.60%	5.56%

TABLE 6.6: Comparisons between the gaps attained by the B&C approach (Gap) and the gap obtained by [Kasperski et al., 2012] (Gap_{KMZ}). In both cases the values are calculated using the LB provided by the B&C approach.

For future work we want to emphasize that the proposed strategy for solving the MMR-ST can be easily adapted for other MMR combinatorial optimization problems with interval data. In particular, if the problem is NP-Hard in its deterministic version (e.g., Steiner Tree), the proposed framework can be modified, for example, by embedding another exact method to solve the Slave Problem. However, such an approach might lead to very high computational effort. In these cases, instead of solving the Slave

Problem exactly, one can use heuristic procedures that although sacrifice guarantee of convergence might yield very good lower bounds in relatively short time. On the top of this, the use of more sophisticated primal heuristics, for example one integrating a Tabu Search as the one designed in [Kasperski et al., 2012], can help to improve the generation of good upper bounds and thus the overall performance of the algorithm.

Chapter 7

A Note on the Bertsimas & Sim Algorithm for Robust Combinatorial Optimization Problems

7.1. Introduction and Motivation

We address a general class of Combinatorial Optimization problems in which both the objective function coefficients and the constraint coefficients are subject to interval uncertainty. When uncertainty has to be taken into consideration, Robust Optimization (RO) arises as methodological alternative to deal with it. The Bertsimas & Sim Robust (B&S) Optimization approach, introduced in [Bertsimas and Sim, 2003], is one of the most important approaches devised to incorporate this type of uncertainty into the decision process. By means of *protection functions*, the obtained solutions are endowed with protection, i.e., they are *robust*, in terms of feasibility and/or optimality for a given *level of conservatism* denoted by a parameter Γ_X, defined by the decision maker. When the coefficients associated with a set of n variables are subject to uncertainty, the *level of conservatism* is interpreted as the number of coefficients that are expected to present uncertainty, i.e., $0 < \Gamma_X \leq n$.

For the case that the uncertain coefficients are only present in the objective function, a well-known result of [Bertsimas and Sim, 2003] states that the robust counterpart of the problem can be computed by solving at most $n + 1$ instances of the original deterministic problem. Thus, the robust counterpart of a polynomially solvable binary optimization problem remains polynomially solvable.

Our Contribution In this work we propose some improvements and extensions to the algorithmic result presented in [Bertsimas and Sim, 2003]. For the case studied in their paper, we show that instead of solving $n+1$ deterministic problems, the robust counterpart can be computed by solving $n-\Gamma_X+2$ deterministic problems (Lemma 1); this improvement is particularly interesting for those cases for which a high level of conservatism, i.e., a large value of Γ_X, is suitable. Additionally, we show that if a knapsack-type constraint is part of a problem and m of its coefficients are affected by uncertainty, an equivalent algorithmic approach can be applied, and the robust counterpart can be computed by solving $m-\Gamma_Y+2$ deterministic problems (Lemma 2), for $0 < \Gamma_Y \leq m$. Likewise, we show that if the uncertain coefficients in the objective function are associated with two disjoint sets of variables, of size n and m respectively, the robust problem can be computed by solving of $(n-\Gamma_X+2)(m-\Gamma_Y+2)$ deterministic problems (Lemma 3), giving to the decision maker the flexibility to define different levels of conservatism to different sets of uncertain parameters. A similar result is also shown for the case that uncertainty is present in a set of n objective function coefficients and in a set of m coefficients of a knapsack-type constraint (Lemma 4). Combining the previous results, we provide a more general result which considers the case in which the uncertain coefficients in the objective function are associated with K disjoint sets of variables and there are L knapsack-type constraints (each of them involving a different set of variables) with uncertain coefficients. For this type of problems, we show that the robust counterpart can be computed by solving a strongly-polynomial number of deterministic problems (Theorem 1), assuming that K and L are constant.

The presented results are important when solving robust counterparts of some well-known combinatorial optimization problems in which different levels of conservatism are associated to disjoint subsets of binary variables. For example, in *Prize-Collecting Network Design Problems* (PCNDPs) (e.g., TSP, Steiner Trees), binary variables are associated to edges and nodes of a graph, and we might associate different levels of conservatism to their corresponding coefficients, costs and prizes, respectively. Besides defining the objective function as the sum of edge costs and node prizes, PCNDPs are frequently modeled using knapsack-type Budget or Quota constraints, and our results can be used in these cases as well, when the coefficients of these constraints are subject to interval uncertainty.

Similarly, in *facility location problems*, location and allocation decisions need to be taken. Each of these decisions involves disjoint sets of variables and, possibly uncertain, coefficients. In these conditions, different levels of conservatism might be suitable for different sets of coefficients. Other prominent examples of problems that fall into this framework are generalizations of the *vehicle routing problem*, involving routing, assignment, location, inventory decision variables and more – for solving the robust counterparts of these problems, the presented result can be used as well.

The viability of the proposed methods strongly relies on the efficacy to solve the deterministic counterparts.

7.2. Main Results

Let us consider the following generic Combinatorial Optimization problem with linear objective function and binary variables $\mathbf{x} \in \{0,1\}^n$:

$$OPT_{P1} = \min \left\{ \sum_{i \in I} c_i x_i \mid \mathbf{x} \in \Pi \right\}, \tag{P1}$$

where $\mathbf{c} \geq 0$, $I = \{1, 2, \ldots, n\}$ and Π is a generic polyhedral region.

Let us assume now that instead of having known and deterministic parameters c_i, $\forall i \in I$, we are actually given uncertain intervals $[c_i, c_i + d_i]$, $\forall i \in I$. Assume that variables $\mathbf{x}$ are ordered so that $d_i \geq d_{i+1}$, $\forall i \in I$, and $d_{n+1} = 0$.

For a given *level of conservatism* $\Gamma_X \in \{1, \ldots, n\}$, the robust formulation of (P1) is defined in [Bertsimas and Sim, 2003] as:

$$ROPT_{P1}(\Gamma_X) = \min \left\{ \sum_{i \in I} c_i x_i + \beta_X^* \left(\Gamma_X, \mathbf{x}\right) \mid \mathbf{x} \in \Pi \right\}, \tag{RP1}$$

where $\beta_X^* \left(\Gamma_X, \mathbf{x}\right)$ is the corresponding *protection function* defined as:

$$\beta_X^* \left(\Gamma_X, \mathbf{x}\right) = \max \left\{ \sum_{i \in I} d_i x_i u_i \mid \sum_{i \in I} u_i \leq \Gamma_X \text{ and } u_i \in [0,1] \, \forall i \in I \right\}. \tag{7.1}$$

This protection function endows robustness to the solutions in terms of protection of optimality in presence of a given level of data uncertainty, represented by Γ_X.

In the context of RO, (P1) is referred to as the *nominal problem* and (RP1) as the corresponding *robust counterpart*.

After applying strong duality to (7.1), problem (RP1) can be rewritten as

$$ROPT_{P1}(\Gamma_X) = \min \sum_{i \in I} c_i x_i + \Gamma_X \theta + \sum_{i \in I} h_i \tag{7.2}$$

$$\text{s.t.} \quad h_i + \theta \geq d_i x_i, \quad \forall i \in I \tag{7.3}$$

$$h_i \geq 0, \ \forall i \in I \text{ and } \theta \geq 0 \tag{7.4}$$

$$\mathbf{x} \in \Pi. \tag{7.5}$$

The previous formulation of the robust counterpart of (P1) has been presented in [Bertsimas and Sim, 2003] and the authors provide a combinatorial framework that computes

$ROPT_{P1}(\Gamma_X)$ by solving $n+1$ nominal problems (Theorem 3, p. 56). The following lemma provides an improvement to this result by reducing the number of iterations of the algorithmic procedure.

Lemma 1. Given $\Gamma_X \in \{1,\ldots,n\}$, the problem (RP1), the robust counterpart of problem (P1), can be computed by solving $(n-\Gamma_X+2)$ nominal problems in the following scheme:

$$ROPT_{P1}(\Gamma_X) = \min_{r\in\{\Gamma_X,\ldots,n+1\}} G^r,$$

where for $r \in \{\Gamma_X,\ldots,n+1\}$:

$$G^r = \Gamma_X d_r + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r}(d_i - d_r)\, x_i\right).$$

Proof. The first part of the proof consists of proving that any optimal solution of (RP1), given by $(\mathbf{x}^*,\mathbf{h}^*,\theta^*)$, satisfies: $\theta^* \in [0, d_{\Gamma_X}]$.

Given the structure of constraints $h_i+\theta \geq d_i x_i$, $\forall i \in I$, it follows that any optimal solution $(\mathbf{x}^*,\mathbf{h}^*,\theta^*)$ satisfies:

$$h_i^* = \max\left(d_i x_i^* - \theta^*, 0\right),$$

and since $x_i \in \{0,1\}$, then it is true that

$$\max\left(d_i x_i^* - \theta^*, 0\right) = \max\left(d_i - \theta^*, 0\right) x_i^*.$$

Therefore, the objective function of (7.2)-(7.5) can be rewritten as

$$ROPT_{P1}(\Gamma_X) = \min \sum_{i\in I} c_i x_i + \Gamma_X\theta + \sum_{i\in I}\max\left(d_i-\theta,0\right) x_i.$$

Let $\mathbf{x}$ be a feasible solution for a given Γ_X. Let $N_{\mathbf{x}}$ be the set of indices $i \in I$ such that $x_i = 1$. Let $I(N_{\mathbf{x}},\Gamma_X)$ be a subset of $N_{\mathbf{x}}$ associated with (at most) the Γ_X largest d_i values.

Let us assume that $|N_{\mathbf{x}}| \leq \Gamma_X$, then we have $I(N_{\mathbf{x}},\Gamma_X) = N_{\mathbf{x}}$, which implies that the cost of each element corresponding to an index $i \in N_{\mathbf{x}}$ will be set to its corresponding upper bound c_i+d_i. This means that if $\mathbf{x}$ is optimal, the minimum value $ROPT_{P1}(\Gamma_X)$ can be calculated as $\sum_{i\in N_{\mathbf{x}}}(c_i+d_i)$, which implies that $\theta^* = d_{n+1} = 0$. Let us now assume that $|N_{\mathbf{x}}| \geq \Gamma_X + 1$. Then, by definition, we have $|I(N_{\mathbf{x}},\Gamma_X)| = \Gamma_X$. Let r^* be the index of the Γ_X-th largest d_i value taken into the solution, i.e., $r^* =$

$\max\{i \,|i \in I(N_{\mathbf{x}}, \Gamma_X)\}$. Then we have:

$$\begin{aligned}\sum_{i\in N_{\mathbf{x}}} c_i + \sum_{i\in I(N_{\mathbf{x}},\Gamma_X)} d_i &= \sum_{i\in N_{\mathbf{x}}} c_i + \sum_{\{i\in N_{\mathbf{x}}: i\le r^*\}} d_i - \sum_{\{i\in N_{\mathbf{x}}: i\le r^*\}} d_{r^*} + \sum_{\{i\in N_{\mathbf{x}}: i\le r^*\}} d_{r^*} \\ &= \sum_{i\in N_{\mathbf{x}}} c_i + \sum_{\{i\in N_{\mathbf{x}}: i\le r^*\}} (d_i - d_{r^*}) + \Gamma_X d_{r^*} \\ &= \sum_{i\in I} c_i x_i + \sum_{i=1}^{r^*} (d_i - d_{r^*}) x_i + \Gamma_X d_{r^*}.\end{aligned}$$

Note that $r^* \ge \Gamma_X$ since $|N_{\mathbf{x}}| \ge \Gamma_X + 1$. Therefore, the minimum value $ROPT_{P1}(\Gamma_X)$ will be reached for $\theta^* = d_r$, where $r \ge \Gamma_X$, and hence, $\theta^* \in [0, d_{\Gamma_X}]$, which completes the first part of the proof.

We now present the second part of the proof, where the previous result is plugged into the procedure devised in [Bertsimas and Sim, 2003], and we find the optimal values of θ by using an equivalent decomposition approach. We decompose the real interval $[0, d_{\Gamma_X}]$ into $[0, d_n]$, $[d_n, d_{n-1}]$, ..., $[d_{\Gamma_X+1}, d_{\Gamma_X}]$. Observe that for an arbitrary $\theta \in [d_r, d_{r-1}]$ we have:

$$\sum_{i\in I} \max(d_i - \theta, 0) x_i = \sum_{i=1}^{r-1} (d_i - \theta) x_i.$$

Therefore, $ROPT_{P1}(\Gamma_X) = \min_{r\in\{\Gamma_X,\ldots,n+1\}} G^r$ where for $r \in \{\Gamma_X, \ldots, n+1\}$

$$G^r = \min \sum_{i\in I} c_i x_i + \Gamma_X \theta + \sum_{i=1}^{r-1} (d_i - \theta)\, x_i,$$

where $\theta \in [d_r, d_{r-1}]$ and $\mathbf{x} \in \Pi$. Since we are optimizing a linear function of θ over the interval $[d_r, d_{r-1}]$, the optimal value of G^r is obtained either by $\theta = d_r$ or by $\theta = d_{r-1}$. So, for $r \in \{\Gamma_X, \ldots, n+1\}$:

$$\begin{aligned}G^r &= \min\left[\Gamma_X d_r + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1} (d_i - d_r)\, x_i\right),\ \Gamma_X d_{r-1} + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1} (d_i - d_{r-1})\, x_i\right)\right] \\ &= \min\left[\Gamma_X d_r + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r} (d_i - d_r)\, x_i\right),\ \Gamma_X d_{r-1} + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1} (d_i - d_{r-1})\, x_i\right)\right].\end{aligned}$$

Therefore,

$$\begin{aligned}ROPT_{P1}(\Gamma_X) = \min\Bigg[&\Gamma_X d_{\Gamma_X} + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i\right), \ldots,\ \Gamma_X d_r + \min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r} (d_i - d_r)\, x_i\right), \ldots, \\ &\min_{\mathbf{x}\in\Pi}\left(\sum_{i\in I} c_i x_i + \sum_{i\in I} d_i x_i\right)\Bigg],\end{aligned}$$

which completes the proof. □

Consider now the following problem that we will refer to as (P2):

$$OPT_{P2} = \min \left\{ \sum_{i \in I} c_i x_i \mid \sum_{j \in J} b_j y_j \leq B \text{ and } (\mathbf{x}, \mathbf{y}) \in \Psi \right\}, \tag{P2}$$

where $\mathbf{y} \in \{0,1\}^m$ are decision variables, $B \in \mathbb{R}^{\geq 0}$ is a constant, $\mathbf{b} \geq 0$, $J = \{1, 2, \ldots, m\}$, and Ψ is a generic polyhedral region.

Let us assume that $\mathbf{c}$ is known with certainty, but instead, the elements of $\mathbf{b}$ are given as uncertain intervals $[b_j, b_j + \delta_j]$, $\forall j \in J$, and that the variables are ordered so that $\delta_j \geq \delta_{j+1}$, $\forall j \in J$, and $\delta_{m+1} = 0$. Given $\Gamma_Y \in \{1, \ldots, m\}$, the robust counterpart of the nominal problem (P2), given the interval uncertainty of vector $\mathbf{b}$, is:

$$ROPT_{P2}(\Gamma_Y) = \min \left\{ \sum_{i \in I} c_i x_i \mid \sum_{j \in J} b_j y_j + \beta_Y^* (\Gamma_Y, \mathbf{y}) \leq B \text{ and } (\mathbf{x}, \mathbf{y}) \in \Psi \right\}. \tag{RP2}$$

In this case, $\beta_Y^* (\Gamma_Y, \mathbf{y})$ provides protection of feasibility in presence of a level of conservatism given by Γ_Y. This problem can be rewritten as

$$ROPT_{P2}(\Gamma_Y) = \min \sum_{i \in I} c_i x_i \tag{7.6}$$

$$\text{s.t} \quad \sum_{j \in J} b_j y_j + \Gamma_Y \lambda + \sum_{j \in J} k_j \leq B \tag{7.7}$$

$$k_j + \lambda \geq \delta_j y_j, \quad \forall j \in J \tag{7.8}$$

$$k_j \geq 0, \ \forall j \in J \text{ and } \lambda \geq 0 \tag{7.9}$$

$$(\mathbf{x}, \mathbf{y}) \in \Psi. \tag{7.10}$$

The following lemma extends for (RP2) the result of Theorem 3 in [Bertsimas and Sim, 2003], and adapts the result of Lemma 1.

Lemma 2. Given $\Gamma_Y \in \{1, \ldots, m\}$, the problem (RP2), the robust counterpart of problem (P2), can be computed by solving $(m - \Gamma_Y + 2)$ nominal problems, in the following scheme:

$$ROPT_{P2}(\Gamma_Y) = \min_{s \in \{\Gamma_Y, \ldots, m+1\}} H^s,$$

where for $s \in \{\Gamma_Y, \ldots, m+1\}$:

$$H^s = \min_{(\mathbf{x}, \mathbf{y}) \in \Psi} \left(\sum_{i \in I} c_i x_i \mid \sum_{j \in J} b_j y_j + \sum_{j=1}^{s} (\delta_j - \delta_s) y_j + \Gamma_Y \delta_s \leq B \right).$$

Proof. The core of the proof consists of showing that for any feasible solution of (7.6)-(7.10) we have $\lambda \in [0, \delta_{\Gamma_Y}]$.

For any feasible solution of (7.6)-(7.10) holds that $k_j = \max(\delta_j y_j - \lambda, 0)$; thus, constraint (7.7) can be written as

$$\sum_{j\in J} b_j y_j + \Gamma_Y \lambda + \sum_{j\in J} \max(\delta_j - \lambda, 0)\, y_j \leq B. \tag{7.11}$$

Let $(\mathbf{x}, \mathbf{y})$ be a feasible solution for a given Γ_X and a given Γ_Y. Let $M_{\mathbf{y}}$ be a set of indices $j \in J$ such that $y_j = 1$. Let $J(M_{\mathbf{y}}, \Gamma_Y)$ be a subset of $M_{\mathbf{y}}$ associated with (at most) the Γ_Y largest values δ_j. Since $(\mathbf{x}, \mathbf{y})$ is a feasible solution, then the following holds:

$$\sum_{j\in M_{\mathbf{y}}} b_j + \sum_{j\in J(M_{\mathbf{y}},\Gamma_Y)} \delta_j \leq B.$$

Let us assume that $|M_{\mathbf{y}}| \leq \Gamma_Y$, then we have $J(M_{\mathbf{y}}, \Gamma_Y) = M_{\mathbf{y}}$, which implies that the cost of each element corresponding to index $j \in M_{\mathbf{y}}$ will be set to its corresponding upper bound $b_j + \delta_j$, and hence constraint (7.11) is satisfied for $\lambda = d_{m+1} = 0$.

Let us now assume that $|M_{\mathbf{y}}| \geq \Gamma_Y + 1$. Then, by definition, we have $|J(M_{\mathbf{y}}, \Gamma_Y)| = \Gamma_Y$. Let $s^* = \max\{j \,|\, j \in J(M_{\mathbf{y}}, \Gamma_Y)\}$. So

$$\begin{aligned}
\sum_{j\in M_{\mathbf{y}}} b_j + \sum_{j\in J(M_{\mathbf{y}},\Gamma_Y)} \delta_j &= \sum_{j\in M_{\mathbf{y}}} b_j + \sum_{\{j\in M_{\mathbf{y}}: j\leq s^*\}} \delta_j - \sum_{\{j\in M_{\mathbf{y}}: j\leq s^*\}} \delta_{s^*} + \sum_{\{j\in M_{\mathbf{y}}: j\leq s^*\}} \delta_{s^*} \\
&= \sum_{j\in M_{\mathbf{y}}} b_j + \sum_{\{j\in M_{\mathbf{y}}: j\leq s^*\}} (\delta_j - \delta_{s^*}) + \Gamma_Y \delta_{s^*} \\
&= \sum_{j\in J} b_j y_j + \sum_{j=1}^{s^*} (\delta_j - \delta_{s^*}) y_j + \Gamma_Y \delta_{s^*} \leq B.
\end{aligned}$$

Note that $s^* \geq \Gamma_Y$ since $|M_{\mathbf{y}}| \geq \Gamma_Y + 1$, and therefore constraint (7.7) will be satisfied for all $\lambda = \delta_s$ such that $s \geq \Gamma_Y$. Therefore for any feasible solution we have $\lambda \in [0, \delta_{\Gamma_Y}]$.

By following similar arguments as those presented in the decomposition approach of the proof of Lemma 1, it holds that

$$\begin{aligned}
ROPT_{P2}(\Gamma_Y) = \min \Bigg[&\min_{(\mathbf{x},\mathbf{y})\in\Psi} \left(\sum_{i\in I} c_i x_i \mid \sum_{j\in J} b_j y_j + \Gamma_Y \delta_{\Gamma_Y} \leq B \right), \ldots, \\
&\min_{(\mathbf{x},\mathbf{y})\in\Psi} \left(\sum_{i\in I} c_i x_i \mid \sum_{j\in J} b_j y_j + \sum_{j=1}^{s} (\delta_j - \delta_s)\, y_j + \Gamma_Y \delta_s \leq B \right), \ldots, \\
&\min_{(\mathbf{x},\mathbf{y})\in\Psi} \left(\sum_{i\in I} c_i x_i \mid \sum_{j\in J} b_j y_j + \sum_{j\in J} \delta_j y_j \leq B \right) \Bigg],
\end{aligned}$$

and the proof is completed. □

We now present a second extension of the algorithm proposed in [Bertsimas and Sim, 2003]. Let us consider now the following nominal problem:

$$OPT_{P3} = \min \left\{ \sum_{i \in I} c_i x_i + \sum_{j \in J} b_j y_j \mid (\mathbf{x}, \mathbf{y}) \in \Psi \right\}. \tag{P3}$$

In case that the elements of both vectors $\mathbf{c}$ and $\mathbf{b}$ are given in terms of closed intervals, the corresponding robust counterpart (for a pair (Γ_X, Γ_Y)) is given by

$$ROPT_{P3}(\Gamma_X, \Gamma_Y) = \min \sum_{i \in I} c_i x_i + \Gamma_X \theta + \sum_{i \in I} h_i + \sum_{j \in J} b_j y_j + \Gamma_Y \lambda + \sum_{j \in J} k_j \tag{7.12}$$

$$\text{s.t.} \quad (7.3),(7.4),(7.8),(7.9) \text{ and } (\mathbf{x}, \mathbf{y}) \in \Psi. \tag{7.13}$$

The following result extends Lemma 1 and provides an algorithmic procedure to solve (7.12)-(7.13).

Lemma 3. Given $\Gamma_X \in \{1, \ldots, n\}$ and $\Gamma_Y \in \{1, \ldots, m\}$, the robust counterpart of problem (P3) can be computed by solving $(n - \Gamma_X + 2)(m - \Gamma_Y + 2)$ nominal problems as follows:

$$ROPT_{P3}(\Gamma_X, \Gamma_Y) = \min_{\substack{r \in \{\Gamma_X, \ldots, n+1\} \\ s \in \{\Gamma_Y, \ldots, m+1\}}} G^{r,s},$$

where for $r \in \{\Gamma_X, \ldots, n+1\}$ and $s \in \{\Gamma_Y, \ldots, m+1\}$:

$$G^{r,s} = \Gamma_X d_r + \Gamma_Y \delta_s + \min_{(\mathbf{x}, \mathbf{y}) \in \Psi} \left(\sum_{i \in I} c_i x_i + \sum_{i=1}^{r} (d_i - d_r) x_i + \sum_{j \in J} b_j y_j + \sum_{j=1}^{s} (\delta_j - \delta_s) y_j \right).$$

Proof. Using an analogous analysis to the one in the proofs of Lemma 1 and 2, we have that for any optimal solution $(\mathbf{x}^*, \mathbf{y}^*, \theta^*, \lambda^*)$, it holds $\theta^* \in [0, d_{\Gamma_X}]$ and $\lambda^* \in [0, d_{\Gamma_Y}]$. Then, by decomposition, the optimum can be found as $ROPT_{P3}(\Gamma_X, \Gamma_Y) = \min_{\substack{r \in \{\Gamma_X, \ldots, n+1\} \\ s \in \{\Gamma_Y, \ldots, m+1\}}} G^{r,s}$ where for $r \in \{\Gamma_X, \ldots, n+1\}$ and $s \in \{\Gamma_Y, \ldots, m+1\}$

$$G^{r,s} = \min \sum_{i \in I} c_i x_i + \Gamma_X \theta + \sum_{i=1}^{r-1} (d_i - \theta) x_i + \sum_{j \in J} b_j y_j + \Gamma_Y \lambda + \sum_{i=1}^{s-1} (\delta_j - \lambda) y_j, \tag{7.14}$$

for which $\theta \in [d_r, d_{r-1}]$, $\lambda \in [\delta_s, \delta_{s-1}]$ and $(\mathbf{x}, \mathbf{y}) \in \Psi$. Since we are optimizing a linear function of θ over the interval $[d_r, d_{r-1}]$ and also a linear function for λ over the interval $[\delta_s, \delta_{s-1}]$, the optimal value of $G^{r,s}$ is obtained for

$$(\theta, \lambda) \in \{(d_r, \delta_s), (d_{r-1}, \delta_s), (d_r, \delta_{s-1}), (d_{r-1}, \delta_{s-1})\}.$$

So, for $r \in \{\Gamma_X, \ldots, n+1\}$ and $s \in \{\Gamma_Y, \ldots, m+1\}$:

$$
\begin{aligned}
G^{r,s} \;=\; & \min\Bigg[\Gamma_X d_r + \Gamma_Y \delta_s + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_r)\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_s)\,y_j\right),\\
& \Gamma_X d_{r-1} + \Gamma_Y \delta_s + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_{r-1})\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_s)\,y_j\right),\\
& \Gamma_X d_r + \Gamma_Y \delta_{s-1} + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_r)\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_{s-1})\,y_j\right),\\
& \Gamma_X d_{r-1} + \Gamma_Y \delta_{s-1} + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_{r-1})\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_{s-1})\,y_j\right)\Bigg]\\
= \; & \min\Bigg[\Gamma_X d_r + \Gamma_Y \delta_s + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r}(d_i - d_r)\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s}(\delta_j - \delta_s)\,y_j\right),\\
& \Gamma_X d_{r-1} + \Gamma_Y \delta_s + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_{r-1})\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s}(\delta_j - \delta_s)\,y_j\right),\\
& \Gamma_X d_r + \Gamma_Y \delta_{s-1} + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r}(d_i - d_r)\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_{s-1})\,y_j\right),\\
& \Gamma_X d_{r-1} + \Gamma_Y \delta_{s-1} + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r-1}(d_i - d_{r-1})\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s-1}(\delta_j - \delta_{s-1})\,y_j\right)\Bigg].
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
ROPT_{P3}(\Gamma_X,\Gamma_Y) = \min\Bigg[& \Gamma_X d_{\Gamma_X} + \Gamma_Y \delta_{\Gamma_Y} + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{j\in J} b_j y_j\right),\ldots,\\
& \Gamma_X d_r + \Gamma_Y \delta_s + \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r}(d_i - d_r)\,x_i + \sum_{j\in J} b_j y_j + \sum_{j=1}^{s}(\delta_j - \delta_s)\,y_j\right),\ldots,\\
& \min_{(\mathbf{x},\mathbf{y})\in\Psi}\left(\sum_{i\in I} c_i x_i + \sum_{i\in I} d_i x_i + \sum_{j\in J} b_j y_j + \sum_{j\in J} \delta_j y_j\right)\Bigg],
\end{aligned}
$$

which completes the proof. □

As a complementary result, one can observe that if in (P2) the cost vector $\mathbf{c}$ is also subject to interval uncertainty (along with the coefficient vector $\mathbf{b}$), the corresponding robust counterpart is given by

$$
ROPT_{P4}(\Gamma_X,\Gamma_Y) = \min \sum_{i\in I} c_i x_i + \Gamma_X \theta + \sum_{i\in I} h_i \tag{7.15}
$$

$$
\text{s.t.} \quad (7.3),\ (7.4),\ (7.7),\ (7.8),\ (7.9) \text{ and } (\mathbf{x},\mathbf{y}) \in \Psi. \tag{7.16}
$$

Combining the results of Lemma 1 and 2, we have the following result,

Lemma 4. Given $\Gamma_X \in \{1,\ldots,n\}$ and $\Gamma_Y \in \{1,\ldots,m\}$, the robust problem (7.15)-(7.16) can be solved by solving $(n-\Gamma_X+2)(m-\Gamma_Y+2)$ nominal problems as follows:

$$
ROPT_{P4}(\Gamma_X,\Gamma_Y) = \min_{\substack{r\in\{\Gamma_X,\ldots,n+1\}\\ s\in\{\Gamma_Y,\ldots,m+1\}}} H^{r,s},
$$

where for $r \in \{\Gamma_X, \ldots, n+1\}$ and $s \in \{\Gamma_Y, \ldots, m+1\}$:

$$H^{r,s} = \Gamma_X d_r + \min_{(\mathbf{x},\mathbf{y})\in\Psi} \left(\sum_{i\in I} c_i x_i + \sum_{i=1}^{r} (d_i - d_r)\, x_i \mid \sum_{j\in J} b_j y_j + \sum_{j=1}^{s} (\delta_j - \delta_s)\, y_j + \Gamma_Y \delta_s \leq B \right).$$

We omit the proof of this result as it follows from the proofs of Lemma 2 and 3.

7.3. General Result

In light of Lemmas 3 and 4, we now generalize the previous results considering a more general Combinatorial Optimization problem under interval uncertainty and propose a combinatorial framework to solve its robust counterpart.

Let us consider a case in which the set of binary variables is partitioned into $K+L$ subsets given by $(\mathbf{x}^1, \ldots, \mathbf{x}^K, \mathbf{y}^1, \ldots, \mathbf{y}^L)$, associated with indices $(I^1, \ldots, I^K, J^1, \ldots, J^L)$. Variables $(\mathbf{x}^1, \ldots, \mathbf{x}^K)$ appear in the objective function with non-negative cost vectors $(\mathbf{c}^1, \ldots, \mathbf{c}^K)$, and $(\mathbf{y}^1, \ldots, \mathbf{y}^L)$ variables appear in L disjoint knapsack constraints with non-negative coefficients $(\mathbf{b}^1, \ldots, \mathbf{b}^L)$ and non-negative right-hand-side bounds $(B^1, \ldots, B^L)$. Let Ψ' be a generic polyhedron containing the feasibility conditions for $(\mathbf{x}^1, \ldots, \mathbf{x}^K, \mathbf{y}^1, \ldots, \mathbf{y}^L)$. With these elements we define nominal problem (P5) as

$$OPT_{P5} = \min_{(\mathbf{x}^1,\ldots,\mathbf{y}^L)\in\Psi'} \left\{ \sum_{i\in I^1} c_i^1 x_i^1 + \ldots + \sum_{i\in I^K} c_i^K x_i^K \mid \sum_{j\in J^1} b_j^1 y_j^1 \leq B^1, \ldots, \sum_{j\in J^L} b_j^L y_j^L \leq B^L \right\}. \tag{P5}$$

We assume now that all elements of the cost vectors $(\mathbf{c}^1, \ldots, \mathbf{c}^K)$ and all elements of the knapsack coefficients $(\mathbf{b}^1, \ldots, \mathbf{b}^L)$ are subject to interval uncertainty; the cost coefficient of variable x_i^k is taken from $[c_i^k, c_i^k + d_i^k]$, for each $i \in I^k$ and $k \in \mathcal{K} = \{1, \ldots, K\}$, and the coefficient of variable y_j^l is taken from $[b_j^l, b_j^l + \delta_j^l]$, for each $j \in J^l$ and $l \in \mathcal{L} = \{1, \ldots, L\}$. Assume that variables $(\mathbf{x}^1, \ldots, \mathbf{y}^L)$ are ordered so that $d_i^k \geq d_{i+1}^k$ and $d_{|I^k|+1}^k = 0$, for all $i \in I^k$ and $k \in \mathcal{K}$, and $\delta_j^l \geq \delta_{j+1}^l$ and $\delta_{|J^l|+1}^l = 0$, for all $j \in J^l$ and $l \in \mathcal{L}$.

To each set of cost coefficients we associate a level of conservatism $0 \leq \Gamma_X^k \leq |I^k|$, for all $k \in \mathcal{K}$, and to each knapsack constraint we associate a level of conservatism $0 \leq \Gamma_Y^l \leq |J^l|$, for all $l \in \mathcal{L}$. The following Theorem unifies the previous results.

Theorem 1. For given $0 \leq \Gamma_X^k \leq |I^k|$, for all $k \in \mathcal{K}$, and $0 \leq \Gamma_Y^l \leq |J^l|$, for all $l \in \mathcal{L}$, the robust counterpart of (P5), $ROPT_{P5}(\Gamma_X^1, \ldots, \Gamma_X^K, \Gamma_Y^1, \ldots, \Gamma_Y^L)$, can be computed by solving

$$\prod_{k\in\mathcal{K}} (|I^k| - \Gamma_X^k + 2) \prod_{l\in\mathcal{L}} (|J^l| - \Gamma_Y^l + 2)$$

problems given by

$$ROPT_{P5}(\Gamma_X^1,\ldots,\Gamma_X^K,\Gamma_Y^1,\ldots,\Gamma_Y^L) = \min_{\substack{r^1\in\{\Gamma_X^1,\ldots,|I^1|+1\}\\ \vdots \quad \vdots \\ s^L\in\{\Gamma_Y^L,\ldots,|J^L|+1\}}} F^{(r^1,\ldots,r^K,s^1,\ldots,s^L)},$$

where for $r^1 \in \{\Gamma_X^1,\ldots,|I^1|+1\},\ \ldots, r^K \in \{\Gamma_X^K,\ldots,|I^K|+1\}$ and $s^1 \in \{\Gamma_Y^1,\ldots,|J^1|+1\},\ \ldots, s^L \in \{\Gamma_Y^L,\ldots,|J^L|+1\}$, we have that

$$\begin{aligned} F^{(r^1,\ldots,r^K,s^1,\ldots,s^L)} =& \Gamma_x^1 d_{r^1} + \ldots + \Gamma_x^K d_{r^K} + \min_{(\mathbf{x}^1,\ldots,\mathbf{y}^K)\in\Psi'} \{\varphi^1(r^1) + \ldots + \varphi^K(r^K) \mid \\ & \xi^1(s^1) \le B^1,\ldots,\xi^L(s^L) \le B^L\}, \end{aligned}$$

such that

$$\varphi^l(r^k) = \sum_{i\in I^k} c_i^k x_i^k + \sum_{i=1}^{r^k} \left(d_i^k - d_{r^k}^k\right) x_i^k,\ \forall k \in \mathcal{K},$$

and

$$\xi^l(s^l) = \sum_{j\in J^l} b_j^l y_j^l + \sum_{j=1}^{s^l} \left(\delta_j^l - \delta_{s^l}^l\right) y_j^l,\ \forall l \in \mathcal{L}.$$

Proof. The robust counterpart of (P5) can be written as

$$ROPT_{P5}(\Gamma_X^1,\ldots,\Gamma_X^K,\Gamma_Y^1,\ldots,\Gamma_Y^L) = \min \sum_{k\in\mathcal{K}} \left(\sum_{i\in I^k} c_i^k x_i^k + \Gamma_X^k \theta^k + \sum_{i\in I^k} h_i^k \right) \tag{7.17}$$

$$\text{s.t.} \quad \sum_{j\in J^l} b_j^l y_j^l + \Gamma_Y^l \lambda^l + \sum_{j\in J^l} k_j^l \le B^l, \quad l \in \mathcal{L} \tag{7.18}$$

$$h_i^k + \theta^k \ge d_i^k x_i^k \text{ and } \theta^k \ge 0, \quad \forall i \in I^k,\ k \in \mathcal{K} \tag{7.19}$$

$$k_j^l + \lambda^l \ge \delta_j^l y_j^l \text{ and } \lambda^l \ge 0, \quad \forall l \in J^l,\ l \in \mathcal{L} \tag{7.20}$$

$$h_i^k \ge 0, \quad \forall i \in I^k,\ k \in \mathcal{K} \tag{7.21}$$

$$k_j^l \ge 0, \quad \forall j \in J^l,\ l \in \mathcal{L}. \tag{7.22}$$

From Lemma 1 and 2, one can show by mathematical induction that any optimal solution for (7.17)-(7.22) satisfies $\theta^{k^*} \in [0, d_{\Gamma_X^k}^k]$, for each $k \in \mathcal{K}$, and $\lambda^{l^*} \in [0, \delta_{\Gamma_Y^l}^l]$, for each $l \in \mathcal{L}$. Finally, mathematical induction is applied to the previously used decomposition approach to derive the result for computing $ROPT_{P5}(\Gamma_X^1,\ldots,\Gamma_Y^L)$. □

As stressed in the Introduction, several Combinatorial Optimization problems are particular cases of (P5), and if interval uncertainty in their parameters is brought into play, the algorithmic procedure described by Theorem 1 could be an alternative for solving their robust counterparts.

Chapter 8

Vulnerability Assessment of Spatial Networks: Models and Solutions

8.1. Introduction

Shortest path problems correspond to an old and very known class of problems in combinatorial optimization. A variant of one of these basic problem consists on analyzing the effects of removing arcs from a network. In Wollmer [1964] the problem of removing k arcs that cause the greatest decrease in the maximum flow from a source to a sink in a planar network is studied. This problem is a special case of a broad class of network optimization problems known as *interdiction* problems. Applied to the shortest s,t-path problem, the interdiction problem can be defined in the following way. Given a graph $G = (V, E)$ with a non-negative length function on its arcs $l : E \to \mathbb{R}$ and two terminals $s, t \in V$, the goal is to destroy all (or the *best*) paths from s to t in G by *optimally* eliminating as many arcs of A as possible (usually respecting a so-called *interdiction budget*). Interdiction problems are often used to measure the robustness of solutions of network optimization problems. In Khachiyan et al. [2008] several versions of these problems are studied; they consider the case of *total limited interdiction* when a fixed number of k arcs can be removed, and *node-wise limited interdiction* (for each node $v \in V$ a fixed number $k(v)$ of out-going arcs can be removed). For a complete survey on early interdiction problems with different underlying network properties the reader is referred to Church et al. [2004]. For a more general discussion regarding network vulnerability approaches we suggest to see Murray [2013].

Based on a well-known network interdiction model we formulate a framework of combinatorial optimization problems whose solutions can be used for assessing the vulnerability of spatial networks in the case of disruptions. We design a flexible model of

network disruption based on the geometric characteristics of spatial networks. This model incorporates the nature of the disruptions present in different situations such as military planning Golden [1978], Israeli and Wood [2002], terrorist attacks Salmeron et al. [2009] or emergency control of infectious disease spreading Assimakopoulos [1987]. The proposed problems, along with the model of disruption, span several realizations of network interdiction providing a useful tool to characterize network vulnerability. Our aim is to propose a methodology that uses network optimization problems to characterize the robustness of a network in the presence of multiple failures.

In §8.2 we present the optimization framework for vulnerability assessment; in §3.5 we report computational results on realistic instances; these results show the versatility of the proposed models to characterize the robustness of the network infrastructure. Finally, in §3.6 we draw final conclusions and propose paths for future work.

8.2. Vulnerability Measures as Optimization Problems

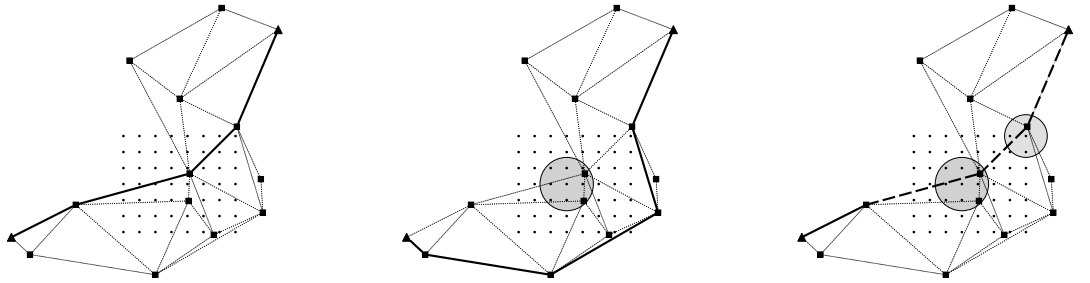
Notation Let $G = (V, E)$ be a spatial network such that $|V| = n$ and $|E| = m$. Let $s, t \in V$ be a source and a target node respectively; l_e, $\forall e : \{i, j\} \in E$, be the cost of edge e (distance between i and j); and ℓ be the cost of the shortest s, t-path on G with edge costs given by l_e, $\forall e \in E$.

Let $\mathcal{X} \subset \mathbb{R}^2$ be an arbitrary sub-region of $\mathbb{R}^2$. An element $x \in \mathcal{X}$ is a *point* in $\mathcal{X}$; for a given point x and a given edge e, let $d(x, e)$ be the minimum distance between x and the line segment defined by e (recall that $e : \{i, j\}$ links node i with node j, whose positions are given). For a given $R \in \mathbb{R}^{>0}$ and a given $x \in \mathcal{X}$, let $E_x = \{e \in E \mid d(x, e) > R\}$ and $\overline{E}_x = \{e \in E \mid d(x, e) \leq R\}$. In other words, E_x is the set of edges that are not *reached* by the disk of radius R centered at x (the *disruption disk* $\rho(x, R)$), and $\overline{E}_x$ is the set of disrupted or interdiced edges. We will refer to $G_x = (V, E_x)$ as the *operating network* with respect to $\rho(x, R)$. Note that G_x might be disconnected.

The model of failure represented by $\rho(x, R)$ embodies a characteristic of disruption produced by many different sources: instead of having isolated failures, we have a set of failures all of them circumscribed within a delimited area. This naturally occurs in the application contexts that we have already mentioned.

8.2.1 The Max-Cost Single-Failure Shortest Path Problem

Let us assume that $\mathcal{X}$ is a finite set of points x in $\mathbb{R}^2$ and that R can take values in $\mathcal{R}$ which is a finite subset of $\mathbb{R}^{>0}$. Given a radius $R \in \mathcal{R}$ and a discrete set $\mathcal{X}$, we are interested in knowing what is the *maximum* length Ω of a *shortest* s, t-path across all possible locations $x \in \mathcal{X}$ of the disruption disk $\rho(x, R)$.



(a) No failure disk. (b) A disk $\rho(x_1, R_1)$. (c) Two disks $\rho(x_1, R_1)$ and $\rho(x_2, R_2)$.

FIGURE 8.1: Example of a network $G = (V, E)$, nodes s and t, a region $\mathcal{X}$ and different interdiction cases.

Knowing Ω is threefold: (i) It tells us how severe can a disruption be by comparing the value of Ω with respect to ℓ; in other words, the increase of the transportation time between s and t induced by a failure located in the worst location $x^* = \arg_{x\in\mathcal{X}}\{\Omega\}$. (ii) From the tactical point of view, *preventive* actions can be taken in order to reduce the chances that a failure can be produced at x^* or the edges $\overline{E}_{x^*}$ can be reinforced to increase their reliability. And (iii) we can know whether the network is *so* vulnerable that s and t might be disconnected, which can be verify if $\Omega = \infty$.

The problem of calculating Ω will be called the Max-Cost Single-Failure Shortest Path Problem (MCSFSPP). Therefore, the MCSFSPP is an optimization problem whose objective function value is a vulnerability measure of the network on which it is solved. Intuitively, the MCSFSPP can be solved as follows. For a given $x \in \mathcal{X}$, let ℓ_x be the cost of the shortest s,t-path on G_x with edge costs l_e^x defined as $l_e^x = l_e$ if $e \in E_x$ and $l_e^x = M$ if $e \in \overline{E}_x$, with $M = O(m \max_{e\in E} l_e)$; therefore, $\Omega = \max_{x\in\mathcal{X}} \ell_x$. If $\Omega > M$ then there is at least one x for which s and t cannot be connected.

In Figure 8.1(a) it is shown a network $G = (V, E)$ where s and t correspond to the nodes represented with triangles and $\mathcal{X}$ is represented by a grid of 8×7 points in the background of part of G; an optimal s,t-path is shown with bold edges. In Figure 8.1(b) we show the case where a disruption disk $\rho(x_1, R_1)$ interdicts the network such that an alternative (an more expensive) s,t-path has to be established ($\Omega < M$). And in Figure 8.1(c) a more complex situation is shown; here two disruption disks, $\rho(x_1, R_1)$ and $\rho(x_2, R_2)$, are simultaneously interdicting the network. In the latter case all possible s,t-paths (one of them is shown in bold dashed lines) have at least one interdicted edge, i.e., $\Omega > M$.

The MCSFSPP is closely related with the network interdiction problems studied from the 70's up to now [Fulkerson and Harding, 1977, Golden, 1978, Phillips, 1993, Cormican et al., 1998, Israeli and Wood, 2002] and [Hemmecke et al., 2003]. In the following, we will use this basic definition to construct generalizations addressing different, but complementary, measures of vulnerability under different models of failure.

Mixed Integer Programming Formulation for the MCSFSPP

Let $\mathbf{f} \in [0,1]^m$ be a vector of $[0,1]$-flow variables. An s,t-path p in G is induced by a given allocation of flows $\mathbf{f}$ if the following constraints are satisfied:

$$\sum_{k \in V | e:\{j,k\} \in E} f_{j,k} - \sum_{i \in V | e:\{i,j\} \in E} f_{i,j} = \begin{cases} 1, & \text{if } j = s \\ 0, & \text{if } j \in V \setminus \{s,t\} \\ -1, & \text{if } j = t. \end{cases} \tag{SP.1}$$

For a given $x \in \mathcal{X}$, the problem of finding ℓ_x can be defined as

$$\ell_x = \min \left\{ \sum_{e \in E} l_e^x f_e \mid \text{(SP.1) and } \mathbf{f} \in [0,1]^m \right\}. \tag{ℓ_x}$$

Let $\mathbf{y} \in \{0,1\}^{|\mathcal{X}|}$ be a vector of binary variables such that $y_x = 1$ if the failure disc is centered at x and $y_x = 0$ otherwise. Now, let $\mathbf{z} \in \{0,1\}^m$ be a set of binary variables such that $z_e = 1$ if edge e is *operative* and $z_e = 0$ otherwise for any given $x \in \mathcal{X}$. Variables $\mathbf{y}$ and $\mathbf{z}$ are related as follows

$$y_x + z_e \leq 1, \ \forall e \in E \mid d(x,e) \leq R, \ \forall x \in \mathcal{X} \tag{YZ.1}$$

$$\sum_{x \in \mathcal{X} | d(x,e) > R} y_x - z_e \leq 0, \ \forall e \in E. \tag{YZ.2}$$

Constraints (YZ.1) and (YZ.2) state that, for any $x \in \mathcal{X}$, an edge e has to be operative ($z_e = 1$) if is not reached by the disruption disk $\rho(x,R)$. Since a single disruption disk affects the network, we have that

$$\sum_{x \in \mathcal{X}} y_x = 1. \tag{YZ.3}$$

Using (YZ.1) and (YZ.2), for a given $x \in \mathcal{X}$ the edge costs l_e^x can be written as $l_e^x = l_e z_e + (1 - z_e)M, \ \forall e \in E$. Hence, the MCSFSPP is as follows

$$\Omega = \max_{x \in \mathcal{X}} \left\{ \ell_x \mid \text{(YZ.1), (YZ.2), (YZ.3) and } (\mathbf{y}, \mathbf{z}) \in \{0,1\}^{|\mathcal{X}|+m} \right\}. \tag{Ω}$$

Problem (Ω), as it is, is non-linear. To linearize it, we will convert the max min objective into a pure max one; to do so, let us consider the dual of (ℓ_x), which is given by

$$\ell_x = \max \left\{ \gamma_t - \gamma_s \mid \gamma_j - \gamma_i \leq l_{ij} z_{ij} + (1 - z_{ij})M, \ \forall e : \{i,j\} \in E \text{ and } \boldsymbol{\gamma} \in \mathbb{R}^n \right\}. \tag{λ}$$

Embedding (λ) into(Ω), we get the next MILP formulation for the MCSFSPP:

$$\Omega = \max \quad \gamma_t - \gamma_s \tag{MCSF.1}$$
$$\text{s.t} \quad \text{(YZ.1), (YZ.2) and (YZ.3)} \tag{MCSF.2}$$
$$\gamma_j - \gamma_i \leq l_{ij} z_{ij} + (1 - z_{ij})M, \ \forall e : \{i, j\} \in E \tag{MCSF.3}$$
$$(\mathbf{y}, \mathbf{z}) \in \{0, 1\}^{|\mathcal{X}|+m} \text{ and } \boldsymbol{\gamma} \in \mathbb{R}^n. \tag{MCSF.4}$$

Note that in our approach we assume that $\rho(x, R)$ can be located in any point $x \in \mathcal{X}$ without any stochastic characterization. That is, any point $x \in \mathcal{X}$ is likely to "host" the center of the failure.

In the proposed setting we assume that if an edge e is disrupted by at least one failure disk $\rho(x, R)$, then it becomes inoperative. However, one can easily extend this to a more general case by defining a coefficient $d_e \geq 0 \ \forall e \in E$ representing the *delay* on edge e in case of interdiction (in our setting $d_e = M \ \forall e \in E$). The MCSFSPP can be redefined by replacing constraint (MCSF.3) with

$$\gamma_j - \gamma_i \leq l_{ij} + (1 - z_{ij})d_{ij}, \ \forall e : \{i, j\} \in E. \tag{MCSF.3b}$$

The Shortest-Path Network Interdiction problem presented in [Israeli and Wood, 2002] is very similar to the definition of the MCSFSPP using (MCSF.3b) instead of (MCSF.3). In that problem, edges can be interdicted without any geometrical pattern among them; instead, they consider interdiction costs so that any feasible disruption of the network should not cost more than a given interdiction *budget*. Later we formally define these concepts and adapt them to our setting.

8.2.2 The Multiple Failures case

As described above, in the MCSFSPP only a single failure $\rho(x, R)$ occurs. However, there are applications in which this characteristic does not hold and, instead, multiple failures occur simultaneously. More precisely, we now have that k failure disks $\rho(x_1, R), \ldots, \rho(x_k, R)$ of radius R are located in $\mathcal{X}$, resulting in an operative network $G_{\mathbf{x}^k} = (V, E_{\mathbf{x}^k})$ where $E_{\mathbf{x}^k} = \{e \in E \mid \min_{x \in \{x_1, \ldots, x_k\}} d(x, e) \leq R\}$. Under these conditions, finding the maximum cost, across all possible $\{x_1, \ldots, x_k\} \in \mathcal{X}^k$, of the shortest s, t-path on $G_{\mathbf{x}^k}$ can be done by modifying MCSFSPP as follows. Instead of (YZ.2), we have

$$\sum_{x \in \mathcal{X}} y_x = k. \tag{YZ.1k}$$

Besides, constraint (YZ.2) should be now adapted in order to impose that $z_e = 1$ if none of the k failure disks reaches e; the new constraint is

$$\sum_{x\in\mathcal{X}|d(x,e)>R} y_x - z_e \leq 1 - \sum_{x\in\mathcal{X}} y_x, \ \forall e \in E, \tag{YZ.2k}$$

clearly if $k = 1$, then (YZ.2k) corresponds to (YZ.2). Therefore, the Max-Cost Multiple-Failure Shortest Path Problem (MCMFSPP) can be formulated as

$$\Omega^k = \max\{\gamma_s - \gamma_t \mid \text{(YZ.1), (YZ.1k), (YZ.2k), (MCSF.3) and (MCSF.4)}\} \tag{MCMF}$$

Note that in formulation (MCMF) it is assumed that $R \in \mathcal{R}$ is known in advance.

Maximal Disruption for an interdiction budget Similar as in [Cormican et al., 1998, Israeli and Wood, 2002, Hemmecke et al., 2003], let us consider that associated with each point $x \in \mathcal{X}$ there is a disruption *cost* $c_x > 0$. Assume that the *interdictors* have a *budget* B of interdiction resources, so that they can disrupt the network using several disks $\rho(x, R)$ as long as the total cost does not exceed B. Formally, the interdiction-budget constraint is given by

$$\sum_{x\in\mathcal{X}} c_x y_x \leq B; \tag{IB}$$

so the Budget Constrained MCMFSPP is formulated as

$$\Omega^k = \max\{\gamma_s - \gamma_t \mid \text{(YZ.1), (IB), (YZ.2k), (MCSF.3) and (MCSF.4)}\} \tag{B}$$

By solving (B) we can know *how* vulnerable the network is if the interdictors are able to optimally use their resources to disrupt it. Models as the one presented in [Israeli and Wood, 2002, Hemmecke et al., 2003] are particular cases of (B) in which $\mathcal{X}$ coincides with the midpoint of every edge $e \in E$ and $R = \epsilon$ (ϵ being infinitesimally small).

Minimum Simultaneity for Complete Vulnerability: Critical k One might be interested in knowing the minimum number of failures (the *critical* k or k^c) that should occur simultaneously in order to have at least one set $\rho(x_1, R), \ldots, \rho(x_k, R)$ that damages the network so that s and t cannot be connected anymore or the shortest length between them is greater than a threshold Θ.

The value k^c and the corresponding collection $\{x_1, \ldots, x_{k^c}\}$ will enable a decision maker to perform more general preventive actions to endure the network not in a single but a in several areas. In many practical contexts, the possibility of multiple and synchronized failures might be the rule, so knowing k^c might play a strategical role. Clearly, for a given R, the larger k^c is the more robust the network is. Mathematically, one can formulate the search for k^c as

$$k^c = \min\{k \mid \text{(YZ.1), (YZ.1k), (YZ.2k), (MCSF.3), (MCSF.4)}\ , \\ \gamma_s - \gamma_t \geq \Theta \text{ and } k \in \mathbb{Z}_{\geq 0}\ \} \tag{k^c}$$

If $\Theta = M$, then (k^c) aims at finding the minimum k such that allocating k disks produces a disconnection between s and t. A similar model is presented in [Hemmecke et al., 2003] in the context of interdiction in stochastic networks.

If instead of k^c one is interested in knowing the minimum *cost* needed to produce a damage represented by Θ, model (k^c) can be easily modified by replacing the objective function of (k^c) with $C^c = \min \sum_{x \in \mathcal{X}} c_x y_x$.

8.3. Computational Results

8.3.1 Instance Benchmark and Solver Setting

Instance Benchmark For our experiments we consider three sets of instances: `ND`, `US` and `Bangladesh`.

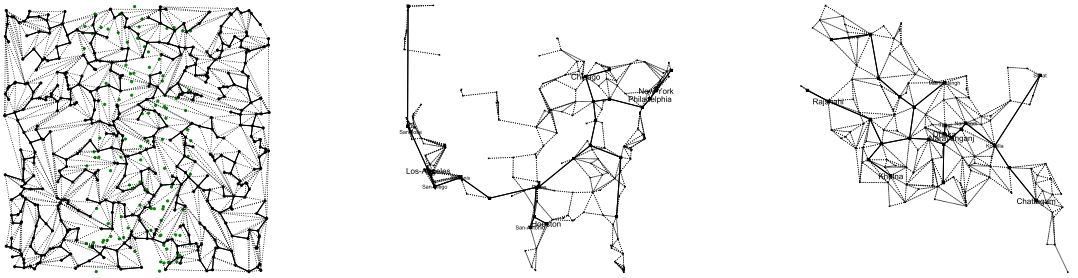
In the first set, the instances are generated as follows: (i) n points are randomly located in a unit Euclidean square; (ii) a minimum spanning tree connecting all points is calculated; (iii) $\beta \times n$ additional edges are added to the network such that an edge is added if l_{ij} (euclidean distance) satisfies $l_{ij} \leq \alpha/\sqrt{n}$ and the planarity of the network is still preserved; (iv) the set $\mathcal{X}$ is created by randomly located K points within the are area defined by points (x_1, y_1), (x_2, y_1), (x_1, y_2) and (x_2, y_2).

For experiments we have considered $n \in \{500, 1000\}$, $\beta = 1.5$, $\alpha = 1.6$, $(x_1, x_2, y_1, y_2) = (0.3, 0.7, 0.0, 1.0)$ ($\mathcal{X}_1$) and $(x_1, x_2, y_1, y_2) = (0.1, 0.9, 0.1, 0.9)$ ($\mathcal{X}_2$), and $K = 100$.

In Figure 8.2(a) it is shown an example of an instance with 500 nodes and $\mathcal{X}$ contained in $(0.3, 0.0)$, $(0.7, 0.0)$, $(0.3, 1.0)$ and $(0.7, 1.0)$.

In the case of groups `US` and `Bangladesh` we consider the geographical coordinates of the most populated cities in each case [see United Nations Statistics Division, 2013] to define the set V. Then, we used an approximation of their highway and interurban road system with the information available in [Google, 2013] to approximate the set of edges E. The set $\mathcal{X}$ is created by randomly located K points within the are area defined by points (x_1, y_1), (x_2, y_1), (x_1, y_2) and (x_2, y_2). In Figures 8.2(b) and 8.2(c) we show the networks used to generate the instances `US` and `Bangladesh` respectively. In the case of `US`, the area $\mathcal{X}$ is given by placing 100 points in the so-called *south* area. With this we intend to represent possible cases of failure produced by hurricanes and other natural disasters. For the `Bangladesh` instances, we have created $\mathcal{X}$ by placing 100 points in squared area in the very center that covers around the 15% of the total area.

In the case of instances `ND`, nodes s and t are selected as those with the longest euclidean distance. In the case of instances `US` we have used $s \in \{\text{NY:New York}, \text{CH:Chicago}\}$



(a) ND with $n = 500$. (b) Network of US. (c) Network of Bangladesh.

FIGURE 8.2: Representation of the instances used for computations.

and $t \in \{\text{LA:Los Ángeles}, \text{HS:Houston}\}$; likewise, in the case of instances Bangladesh we have used $s = \text{Rajshahi}$ and $t = \text{Silhat}$.

Solver Setting Models (MCSF.1)-(MCSF.4), (MCMF) and (k^c) were solved using CPLEX 12.5 (all CPLEX parameters were set to their default values). The experiments were performed on a Intel Core i7-3610QM machine with 8 GB RAM.

8.3.2 Vulnerability Assessment of Spatial Networks: Solutions

From the operative perspective, the value of R corresponds to the *intensity* of a disruption. If we consider the MCSFSPP or the MCMFSPP we would expect that a *vulnerable* network is such that Ω increases quickly (up to M) when R increases marginally. On the other hand, a *reliable* network is such that the cost of the shortest s,t-path does not change *too much* even if R increases considerably.

In Table 8.1 we report solutions for the MCSFSPP for instances of group ND considering different values of n, different compositions of set $\mathcal{X}$ and different values of R (columns 1, 4, 7 and 10). In columns $\Delta\%\Omega$ is reported the relative increase of Ω, for a given $\mathcal{X}$ and a given R, with respect to cost of the shortest s,t-path without any failure. In this column, "-" means that all paths have been disrupted. In columns t[sec] are reported the running times in seconds needed to reach optimality. One can observe from this table that when the area where the failure can occur, $\mathcal{X}$, is such that covers a *stripe* on the network (as $\mathcal{X}_1$) then it is more vulnerable (see the values $\Delta\%\Omega$ for different R) than a network in which the failure area, although larger, still leaves *corridors* where s,t-paths can be constructed, as for $\mathcal{X}_2$. In a warfare context, if we were to be the enemies, this analysis would suggest us that is better to concentrate our resources in a narrower area potentially spanning a complete stripe of the network than in a larger area (which might be more expensive) that does not properly covers the network. On the other hand, who wants to protect the network should concentrate the efforts in protecting at least one corridor connecting s and t.

$n = 500$						$n = 1000$					
$\mathcal{X}_1$			$\mathcal{X}_2$			$\mathcal{X}_1$			$\mathcal{X}_2$		
R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]
0.01	2.17	38.92	0.01	0.00	32.93	0.01	2.48	115.39	0.01	0.14	144.66
0.02	3.93	46.46	0.02	1.64	36.83	0.02	2.93	153.58	0.02	0.69	215.16
0.03	3.93	65.63	0.03	1.64	49.73	0.03	5.38	235.95	0.03	1.52	240.88
0.04	5.15	80.79	0.04	1.64	63.06	0.04	7.31	258.85	0.04	1.52	265.00
0.05	5.15	103.01	0.05	1.64	79.67	0.05	7.17	395.46	0.05	1.52	259.80
0.10	5.15	97.83	0.10	1.64	112.76	0.10	8.69	917.78	0.10	3.87	373.61
0.15	-	53.70	0.15	10.64	111.65	0.15	9.93	587.27	0.15	6.19	843.45

TABLE 8.1: Solutions for the MCSFSPP considering different values of R (Instances ND)

In Tables 8.2 and 8.3, results for (MCMF) and (k^c), respectively, are reported. The analysis is similar as for Table 8.1. From Table 8.2 we can see that the increase of $\Delta\%\Omega$ (due to a larger k), is greater for $\mathcal{X}_1$ than for $\mathcal{X}_2$. Along the same lines, we see from Table 8.3 that the minimum resources needed to disconnect s and t (see columns k^c) are greater for $\mathcal{X}_2$ than for $\mathcal{X}_1$. In Table (k^c), when results for a given R are not reported (e.g., $R = 0.01$ for $n = 500$ and $\mathcal{X}_1$) is because not even $|\mathcal{X}|$ failure disks are enough to make the s, t connectivity collapse. This applies for all the remaining Tables.

From the algorithmic point of view, we can notice in Tables 8.1, 8.2 and 8.3 that the search for an alternative path in a disrupted network is not *for free*. In all cases we see an increase of the algorithmic effort (time) needed to find such a path (if exists). This is due to the high combinatorial nature of the problem when more edges are subject

500								1000							
$\mathcal{X}_1$				$\mathcal{X}_2$				$\mathcal{X}_1$				$\mathcal{X}_2$			
R	k	$\Delta\%\Omega$	t[sec]	R	k	$\Delta\%\Omega$	t[sec]	R	k	$\Delta\%\Omega$	t[sec]	R	k	$\Delta\%\Omega$	t[sec]
0.01	1	2.17	24.15	0.01	1	0.00	22.51	0.01	1	2.48	88.16	0.01	1	0.01	88.64
	2	3.93	25.07		2	0.00	21.96		2	3.03	94.80		2	0.09	96.35
	3	4.74	25.02		3	0.00	22.11		3	3.03	93.90		3	1.63	93.57
	4	5.56	24.87		4	0.00	22.21		4	3.03	95.08		4	1.63	95.08
	5	6.79	24.52		5	0.00	22.14		5	3.03	94.15		5	1.63	94.4
0.1	1	5.15	59.94	0.1	1	1.64	69.61	0.1	1	8.68	889.61	0.1	1	3.87	222.36
	2	-	26.43		2	13.82	142.93		2	17.89	3448.93		2	8.16	1012.06
	3	-	98.05		3	13.92	297.95		3	21.24	75319.9		3	12.45	5434.01
	4	-	526.47		4	-	63.01		4	28.14	26123.3		4	21.99	13130.3
	5	-	147.39		5	-	149.54		5	-	16576.5		5	-	742.63

TABLE 8.2: Solutions for the MCMFSPP considering different values of R and k (Instances ND)

n	$\mathcal{X}$	R	k^c	t[sec]	n	$\mathcal{X}$	R	k^c	t[sec]
500	$\mathcal{X}_1$	0.10	2	170.76	1000	$\mathcal{X}_1$	0.10	5	628.76
		0.15	1	43.09			0.15	2	805.32
	$\mathcal{X}_2$	0.03	10	39.05		$\mathcal{X}_2$	0.02	19	219.38
		0.04	9	109.70			0.03	13	621.60
		0.05	7	161.06			0.04	11	933.85
		0.10	4	274.20			0.05	8	1759.32
		0.15	3	162.47			0.10	5	1277.24
							0.15	3	1214.14

TABLE 8.3: Solutions of (k^c) considering different values of R (Instances ND)

to be interdicted (when R increases and/or when k is either greater than 1 or when it is a variable).

In the case of USA Instances, we report in Table 8.4 results of the MCSFSPP considering different pairs of s and t and different values of R. In this case, we can see that different combinations of s and t yield to different levels of vulnerability in the system. For instance, the network is considerably more vulnerable when it is intended to host a path from Chicago to Los Ángeles than when the path should be established from Chicago to Houston. This is due to the fact that, in our instance, the system of roads connecting the north of the Midwest with the south of the West Coast is composed by relatively few elements. Hence, a single disruption disk (that is optimally placed) is enough to interrupt the communication between the cities. In this case the values of $\Delta\%\Omega$ are particularly important from the tactic point of view; if it is up to the decision maker to decide where to establish both the source and the target of the transportation system, then it might preferable to have New York - Houston than, for instance, Chicago - Los Ángeles. However, this analysis is valid only when a single failure occurs. For an approximate equivalence to real distances, R should be multiply by 1700.

In Figure 8.3(a) we show the solution of the shortest path problem between New York

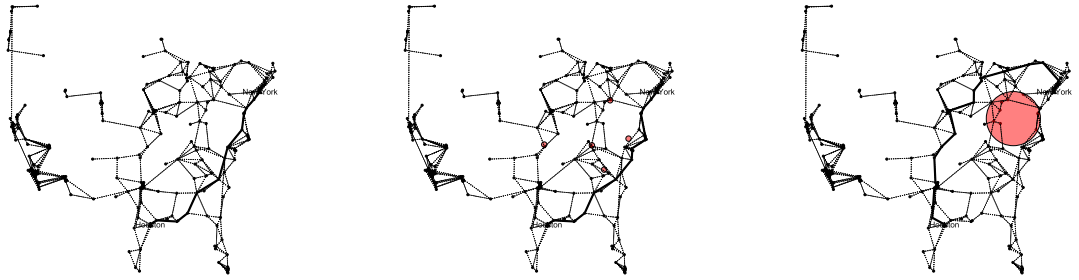
$\{NY, LA\}$			$\{CH, LA\}$			$\{NY, HS\}$			$\{CH, HS\}$		
R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]	R	$\Delta\%\Omega$	t[sec]
0.01	9.00	17.22	0.01	0.00	17.43	0.01	13.00	17.25	0.01	0.00	17.22
0.02	10.00	21.42	0.02	20.00	25.05	0.02	13.00	26.83	0.02	2.00	20.12
0.03	10.00	24.77	0.03	20.00	33.17	0.03	15.00	22.25	0.03	2.00	35.27
0.04	10.00	23.57	0.04	20.00	31.29	0.04	18.00	23.07	0.04	2.00	28.31
0.05	10.00	25.30	0.05	20.00	35.51	0.05	19.00	25.04	0.05	8.00	36.15
0.10	30.00	34.16	0.10	-	33.67	0.10	45.00	42.31	0.10	-	30.67
0.15	-	40.22	0.15	-	28.41	0.15	-	29.69	0.15	-	28.17
0.20	-	28.00	0.20	-	48.14	0.20	-	27.16	0.20	-	37.13

TABLE 8.4: Solutions for the MCSFSPP considering different values of R (Instances USA)

						$R = 0.01$			$R = 0.1$		
R	$\Delta\%\Omega$	t[sec]	R	k^c	t[sec]	K	$\Delta\%\Omega$	t[sec]	K	$\Delta\%\Omega$	t[sec]
0.01	3.76	8.72	0.05	3	10.06	1	3.76	7.69	1	24.17	8.11
0.02	2.93	7.44	0.10	2	28.52	2	5.48	8.75	2	-	38.28
0.03	3.76	8.99	0.15	1	24.01	3	5.48	8.42	3	-	26.75
0.04	3.76	9.53				4	5.48	8.41	4	-	28.32
0.05	4.78	19.03				5	5.48	7.64	5	-	10.55
0.10	24.17	36.97									
0.15	-	11.19									

TABLE 8.5: Solutions for MCSFSPP, MCSFMPP and k^c, s =Rajshahi and t =Silhat (Instances `Bangladesh`)

and Houston when there is no disruption. In Figure 8.3(b) is shown the solution of the MCMFSPP when 5 disruption disks with $R = 0.01$ are optimally located. In Figure 8.3(b) is shown the solution of the MCMFSPP with $k = 1$ and $R = 0.10$. These figures show how different the optimal s, t-paths can be when the network is disrupted by failures of different magnitude.



(a) $k = 0$ (b) $k = 5$ and $R = 0.01$ (c) $k = 1$ and $R = 0.1$

FIGURE 8.3: Solutions for the MCMFSPP for different k and R (Instances `USA`)

Finally, in Table 8.5 we report results for the Instances `Bangladesh`. From the solutions of the MCSFSPP (reported in columns 1-3) we can see that the relatively dense road system of this country is able to *resist* (small values of $\Delta\%\Omega$), reasonably well the optimal location of a single failure disk up to $R = 0.05$. For greater values, the network can be dramatically damaged. This later observation is reinforced by the results reported in columns 4-6 in the same table: a critical k can be found only if $R \geq 0.05$. When looking at the results of the MCSFMPP (columns 7-9 for $R = 0.01$ and 10-12 for $R = 0.1$) we can see that the network resists *well* ($\Delta\%\Omega \approx 5\%$) several failures with $R = 0.01$; however, if $R = 0.1$ then the network collapses even if $k = 2$.

8.4. Conclusions and Future Work

We have presented a collection of combinatorial optimization problems that in combination allow to measure the vulnerability of a network. Vulnerability is represented by the relative increase of the cost of a s, t-shortest path when part of the network is

disrupted. By analyzing the solutions of these problems for different instances, we have highlighted how different aspects of both the failure and the network yield to different levels of vulnerability.

Two main paths of future work can be identified. First, we should consider the case in which $\mathcal{X}$ is not given by a discrete set of points, but rather as continuous area. Second, at the light of the large computational effort needed to solve some of the instance considered here, we think it is important to design and implement more sophisticated algorithmic techniques such as decomposition approach in order to be able to consider larger and more complex instances.

Chapter 9

The Maximum Weight Connected Subgraph Problem

9.1. Introduction

The *Maximum (Node-) Weight Connected Subgraph Problem* (MWCS) is the problem of finding a connected subgraph with maximum total weight in a node-weighted (di)graph. It belongs to the class of network design problems and has applications in various different areas such as forestry, wildlife preservation planning, systems biology, computer vision, and communication network design.

Lee and Dooly [Lee and Dooly, 1998] introduced a cardinality-constrained version of the problem for building a designed fiber-optic communication network over time, where the given node weights reflect their degree of importance. They defined the *maximum-weight connected graph problem* for an undirected graph with given node weights, in which they search the connected subgraph of maximum weight consisting of exactly a predescribed number of nodes. The same problem version was considered already in [Hochbaum and Pathria, 1994] (the authors called it *Connected k-Subgraph Problem*) for a Norwegian off-shore oil-drilling application.

Another application arises in the area of system biology [see Dittrich et al., 2008, Yamamoto et al., 2009, Backes et al., 2011]. In [Yamamoto et al., 2009], the authors suggest the cardinality-constrained MWCS in order to detect core source components in gene networks, which seem to be responsible for the difference between normal cells and mutant cells. The input graphs are constructed from gene regulation networks combined with gene expression data provided as node weights. Maximum weight connected subgraphs are considered to be good candidates for these core source components. A directed version of the MWCS has been considered in [Backes et al., 2011], where the most deregulated connected subnetwork in regulatory pathways with the highest sum of node scores (arising from expression data) is searched. In their model, they call a

subgraph connected if all the nodes are reachable from one node, also called the *root* in the subgraph. The detected roots are likely to be the molecular *key-players* of the observed deregulation.

A budgeted version arises in conservation planning, where the task is to select land parcels for conservation to ensure species viability, also called *corridor design* [Dilkina and Gomes, 2010]. Here, the nodes of the graph do not only have node weights associated with the habitat suitability but also some costs, and the task is to design wildlife corridors that maximize the suitability with a given limited budget. Also in forest planning, the MWCS arises as a subproblem, e.g., for designing a contiguous site for a natural reserve or for preserving large contiguous patches of mature forest [Carvajal et al., 2013].

A surprising application of the MWCS arises in activity detection in video sequences. Here, a 3D graph is constructed from a video in which the nodes correspond to local video subregions and the edges to their proximity in time and space. The node weights correspond to the degree of activity of interest, and so the maximum weight connected subgraph corresponds to the portion of the video that maximizes a classifier's score [Chen and Grauman, 2012].

All the above mentioned applications have in common that the MWCS arises with node weights only. In many papers, the MWCS has been solved by transforming the given instance to the *Prize-Collecting Steiner Tree Problem*. Here, the given graph has non-negative node weights and negative edge costs, and the task is to find a maximum weight subtree, where the weight is computed as the sum of the node and edge weights in the subtree. The Prize-Collecting Steiner Tree Problem has been studied intensively in the literature [see Johnson et al., 2000, Ljubić et al., 2006], and the publicly available branch-and-cut (B&C) code of [Ljubić et al., 2006] is used in many recent applications to solve the underlying problems to optimality.

However, in their recent work, [Backes et al., 2011] attack the MWCS directly, which has the advantage to avoid variables for the arcs. The authors suggest a new integer linear programming formulation which is based on node variables only. The intention of our research was to study the MWCS straightly, and to suggest tight MIP formulations that improve the MIP models from the literature in theory and practice.

Our Contribution: We propose a new MIP model for the MWCS based on the concept of node separators in digraphs. We provide a theoretical and computational comparison of the new model with other models recently used in the literature. We show that the new model has the advantage of using only node variables while preserving the tight LP bounds of the Prize-Collecting Steiner Tree (PCStT) model. Furthermore, we study the connected subgraph polytope and show under which conditions the newly introduced inequalities are facet defining. In an extensive computational study, we compare different MIP models on a set of benchmark instances used in systems

biology and on an additional set of network design instances. The obtained results indicate that the new formulation outperforms the previous ones in terms of the running time and in terms of the stability with respect to variations of node weights.

The paper is organized as follows. Section 9.2 contains a formal definition of the MWCS and some complexity results. The following Sections provide four different MIP formulations and polyhedral studies. Our B&C algorithm and the practical experiments are discussed in Section 9.5.

9.2. The Maximum Weight Connected Subgraph Problem

In this section we formally introduce the MWCS for directed graphs and discuss some complexity results.

Definition 1. (The Maximum Weight Connected Subgraph Problem, MWCS) Given a digraph $G = (V, A)$, $|V| = n$, with node weights $p : V \to \mathbb{Q}$, the MWCS is the problem of finding a connected subgraph $T = (V_T, A_T)$ of G, that maximizes the score $p(T) = \sum_{v \in V_T} p_v$ and such that there exists a node $i \in V_T$ (called *root* or *key player*) such that every other node $j \in V_T$ can be reached from i by a directed path in T.

The MWCS in undirected graphs is to find a connected subgraph T that maximizes the score $p(T)$. However, if $G = (V, E)$ is an undirected graph, without loss of generality we will consider its bidirected counterpart (V, A) where A is obtained by replacing each edge by two oppositely directed arcs. Hence, it is sufficient to present results that hold for digraphs (which are more general), and the corresponding results for undirected graphs can be easily derived from them. We assume that in our MWCS instances always positive and negative node weights are present, otherwise, the solution would be trivial. Observe that any feasible solution of the MWCS contains a tree with the same solution value. Hence it is equivalent to search a maximum node-weighted tree in the given graph.

Furthermore, it can be distinguished between the *rooted* and *unrooted* MWCS, i.e., a root node r can be pre-specified or not. In this work we will concentrate on the unrooted MWCS, or simply the MWCS in the rest of the paper.

Regarding the complexity of the MWCS, it has been shown that the problem is NP-hard (in the supplementary documentation of the paper by [Ideker et al., 2002], the authors provide an NP-hardness proof sketched by R. Karp). Since it is possible to translate the problem to the Prize-Collecting Steiner tree problem, all its polynomially solvable cases carry over to the MWCS. E.g., the PCStT is solvable in polynomial time for the graph class of bounded treewidth [Bateni et al., 2011].

Furthermore, one can show that the following result holds even when the MWCS is defined on undirected graphs:

Proposition 1. It is NP-hard to approximate the optimum of the MWCS within any constant factor $0 < \epsilon < 1$.

Proof. For a given MWCS instance, let APP be the objective function value of an approximate solution, and let OPT be the optimal solution value. Recall that for a given constant $0 < \epsilon < 1$, a given problem can be approximated within factor ϵ if and only if $APP/OPT \geq \epsilon$, for any problem instance. To prove this result for the MWCS it is sufficient to make a reduction from the SAT problem that works similarly to the one given in [see Theorem 4.1 Feigenbaum et al., 2001]. By doing so, we can show that for a given formula ϕ for SAT, we can build an instance $G = (V, E)$ of the MWCS in polytime, such that: (i) if ϕ is a yes-instance, then the optimal MWCS solution on G has value $\epsilon(1 + \epsilon^3)$, and (ii) if ϕ is a no-instance, then the optimal MWCS solution on G has value ϵ^2. □

Some applications consider the *cardinality-constrained MWCS*, where the task is to find a connected subgraph with K nodes. Hochbaum and Pathria in [Hochbaum and Pathria, 1994] have shown that this problem version is NP-hard even if all node weights are 0 or 1 and the graph is either bipartite or planar. For trees and for complete layered DAGs, it is solvable in polynomial time via dynamic programming [Hochbaum and Pathria, 1994, Lee and Dooly, 1998]. Observe that for this problem version, the node weights can be assumed to be all positive, and the maximization variant and the minimization variant are equivalent. Goldschmidt [O. and Hochbaum, 1997] noted that no approximation algorithm is known with a factor better than $O(K)$, and such an algorithm is almost trivial to find. The cardinality-constrained MWCS (and also the MWCS) can be solved by translating it into the edge-weighted version, which has been studied as the *k-Minimum Spanning Tree Problem* (k-MST) or *k-Cardinality Tree Problem* in the literature [see, e.g., Fischetti et al., 1994, Chimani et al., 2009].

9.3. MIP Formulations for the MWCS

In this section we revise three MIP models for the MWCS recently presented in the literature, and propose a novel approach based on the concept of node separators in digraphs.

The MIP formulations considered in this chapter are based on the observation that if there is a path between i and any other node in $T = (V_T, A_T)$, then we will search for a subgraph which is an arborescence rooted at $i \in V_T$. In our models, two types of binary variables will be used to describe a feasible MWCS solution $T = (V_T, A_T)$: binary variables y_i associated to nodes $i \in V$ will be set to one iff $i \in V_T$, and additional

binary variables x_i will be set to one iff the node $i \in V$ is the key player, i.e., if it is used as the root of the arborescence.

Notation and Preliminaries: A set of vertices $S \subset V$ ($S \neq \emptyset$) and its complement $\overline{S} = V \setminus R$ induce two directed cuts: $(S, \overline{S}) = \delta^+(S) = \{(i,j) \in A \mid i \in S, j \in \overline{S}\}$ and $(\overline{S}, S) = \delta^-(S) = \{(i,j) \in A \mid i \in \overline{S}, j \in S\}$. When there is an ambiguity regarding the graph in which the directed cut is considered, we will sometimes write δ_G instead of only δ to specify that the cut is considered w.r.t. graph G. For a set $C \subset V$, let $D^-(C)$ denote the set of nodes outside of C that have ingoing arcs into C, i.e., $D^-(C) = \{i \in V \setminus C \mid \exists (i,v) \in A, v \in C\}$.

A digraph G is called strongly connected (or simply, *strong*) if for any two distinct nodes k and ℓ from V, there exists a (k,ℓ) path in G. A node i is a cut point in a strong digraph G if there exists a pair of distinct nodes k and ℓ from V such that there is no (k,ℓ) path in $G - i$.

For two distinct nodes k and ℓ from V, a subset of nodes $N \subseteq V \setminus \{k,\ell\}$ is called (k,ℓ) *node separator* if and only if after eliminating N from V there is no (k,ℓ) path in G. A separator N is *minimal* if $N \setminus \{i\}$ is not a (k,ℓ) separator, for any $i \in N$. Let $\mathcal{N}(k,\ell)$ denote the family of all (k,ℓ) separators. Obviously, if $\exists (k,\ell) \in A$ or if ℓ is not reachable from k, we have $\mathcal{N}(k,\ell) = \emptyset$. Let $\mathcal{N}_\ell = \cup_{k \neq \ell} \mathcal{N}(k,\ell)$ be the family of all node separators with respect to $\ell \in V$ that we will refer to as ℓ-separators.

For binary variables $\mathbf{a} \in \{0,1\}^{|F|}$, we denote by $a(F')$ the sum $\sum_{i \in F'} a_i$ for any subset $F' \subseteq F$.

9.3.1 The Prize-Collecting Steiner Tree Model

In [Dittrich et al., 2008] the authors observed that the MWCS on undirected graphs is equivalent to the Prize-Collecting Steiner Tree Problem (PCStT), in the sense that there exists a transformation from the MWCS into the PCStT such that each optimal solution of the PCStT on the transformed graph corresponds to an optimal MWCS solution from the original graph. Recall that, given an undirected graph $H = (V_H, E_H)$ with non-negative node weights $\tilde{p}_v$ and non-negative edge costs $\tilde{c}_e$, the PCStT is the problem of finding a subtree T_H of H that maximizes the function $\sum_{v \in T_H} \tilde{p}_v - \sum_{e \in T_H} \tilde{c}_e$, i.e., the difference between the collected node prizes and edge costs. The transformation from the MWCS into the PCStT is given as follows: Given an input graph G of the MWCS we set $H := G$ and $w = \min_{v \in V} p_v$ (note, that $w < 0$). In order to get non-negative node weights, we set $\tilde{p}_v := p_v - w \ \forall v \in V$ and $\tilde{c}_e = -w$, for all $e \in E$. This transformation also works for digraphs, i.e., if H is a digraph, the PCStT consists of finding a subarborescence of H (rooted at some node $i \in V$) that maximizes the given objective function. The transformation is correct, since any feasible solution is

an arborescence, which has indegree 1 for every node, and the weight transformations neutralize each other.

We now present the MIP model proposed in [Ljubić et al., 2006] for the PCStT that is used for solving the MWCS after transforming it into the PCStT [see Dittrich et al., 2008]. Consider a transformation from a (directed or undirected) PCStT instance into a rooted digraph $G_d = (V_d, A_d)$ that works as follows: If the input graph $G = (V, E)$ is undirected, then we create the arc set A by bidirecting each edge. In any case we now have a directed graph $G = (V, A)$. The vertex set $V_d = V \cup \{r\}$ contains the nodes of the input graph G *and an artificial root* vertex r. We add new arcs from the root r to nodes v whose out-degree is non-empty in order to get the arc set A_d i.e., $A_d = A \cup \{(r, v) \mid v \in V \text{ and } \delta^+(v) \neq \emptyset\}$. All arc weights are set to the weights of their undirected counterparts, and the weight of an arc $(r, v) \in A_d$ is set to w.

In the graph G_d, a subgraph $T_d = (V_{T_d}, A_{T_d})$ that forms a directed tree rooted at r is called a *rooted Steiner arborescence*. It is a feasible solution of the PCStT if the out-degree of the root is equal to one. To model feasible Steiner arborescences in G_d, we will use two types of binary variables: (a) binary variables y_i introduced above associated to all nodes $i \in V$, and (b) binary variables z_{ij}, such that $z_{ij} = 1$ if arc (i, j) belongs to a feasible Steiner arborescence T_d and $z_{ij} = 0$ otherwise, for all $(i, j) \in A_d$.

The set of constraints that characterizes the set of feasible solutions of the unrooted PCStT is given by:

$$z(\delta^-(i)) = y_i, \quad \forall i \in V \setminus \{r\} \tag{9.1}$$

$$z(\delta^-(S)) \geq y_k, \quad \forall S \subseteq V \setminus \{r\},\ k \in S \tag{9.2}$$

$$z(\delta^+(r)) = 1. \tag{9.3}$$

The *in-degree* constraints (9.1) guarantee that the in-degree of each vertex of the tree is equal to one. The directed cut constraints (9.2) ensure that there is a directed path from the root r to each costumer k such that $y_k = 1$. The equality (9.3) makes sure that the artificial root is connected to exactly one of the nodes. Thus, the MWCS can be formulated using the following model that we will denote by $(PCStT)$:

$$\max \left\{ \sum_{v \in V} (p_v - w) y_v + \sum_{(i,j) \in A_d} w z_{ij} \mid (\mathbf{y}, \mathbf{z}) \text{ satisfies (9.1)-(9.3)},\ (\mathbf{y}, \mathbf{z}) \in \{0,1\}^{n+|A_d|} \right\}.$$

The $(PCStT)$ model uses node and arc variables ($\mathbf{y}$ and $\mathbf{z}$) given that it relies on an equivalence with the PCStT. However, considering Definition 1 it seems more natural to find a formulation based only in the space of $\mathbf{y}$ variables since no arc costs are involved. In the next section we will discuss several models that enable elimination of arc variables in the MIP models.

9.3.2 Model of [Backes et al., 2011]

Recently, in [Backes et al., 2011] a new MIP model for the MWCS is introduced which avoids the explicit use of arc variables. Let $\mathcal{C}$ denote the family of all directed cycles in G. The new model, that we will denote by $(CYCLE)$, reads as follows:

$$x(V) = 1 \tag{9.4}$$

$$x_i \leq y_i, \quad \forall i \in V \tag{9.5}$$

$$y(D^-(i)) \geq y_i - x_i, \quad \forall i \in V \tag{9.6}$$

$$y(C) - x(C) - y(D^-(C)) \leq |C| - 1, \quad \forall C \in \mathcal{C} \tag{9.7}$$

$$(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n}. \tag{9.8}$$

Inequalities (9.4) make sure that one node is selected as a root, and inequalities (9.5) state that if the node is chosen as a root, it has to belong to the solution. Constraints (9.6) are the *in-degree constraints* – they ensure that for each node which is not the root, at least one of the incoming neighbors needs to be taken into the solution. In a directed acyclic graph, in-degree constraints are sufficient to guarantee connectivity, but in general, imposing only the in-degree constraints may allow solutions that consist of several disconnected components. To avoid this, cycle constraints (9.7) are added to guarantee connectivity. These constraints make sure that whenever all nodes from a cycle are taken in a solution, and none of them is set as the root, at least one of the neighboring nodes from $D^-(C)$ has to be taken as well.

Observation 1. Constraints (9.7) are redundant for those $C \in \mathcal{C}$ such that $C \cup D^-(C) = V$.

To see this, observe that using the root constraint (9.4), the cycle constraints (9.7) can be rewritten as follows:

$$y(C) \leq y(D^-(C)) + |C| - 1 + x(C) = y(D^-(C)) + |C| - x(D^-(C)),$$

which is always satisfied by the model due to constraints (9.5) and $y_i \leq 1$, for all $i \in V$.

In this model an artificial root node r is not explicitly introduced. However, it is not difficult to see that for any feasible MWCS solution there is a one-to-one mapping between variables z_{ri} introduced above and the variables x_i, for all $i \in V$.

The following result shows that the $(CYCLE)$ model provides very weak upper bounds, in general.

Lemma 1. Given an instance of the MWCS, let OPT be the value of the optimal solution, and let UB be the upper bound obtained by solving the LP relaxation of the $(CYCLE)$ model. Then, there exist MWCS instances for which $UB/OPT \in O(n)$.

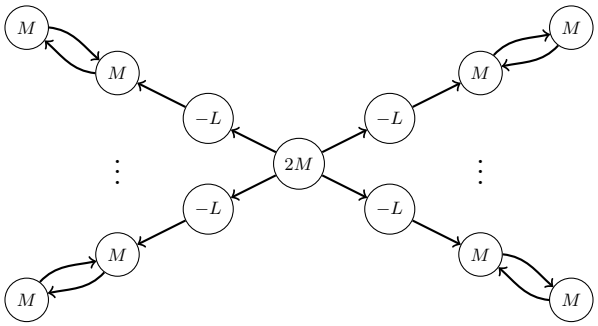


FIGURE 9.1: An example showing that the LP bounds of the $(CYCLE)$ model can be as bad as $O(n)$. The labels of nodes represent their weights: $M > 0$ and $L >> M$.

Proof. Consider an example given in Figure 9.1. The variables of the LP relaxation of the $(CYCLE)$ model are set as follows: $y_i = x_i = 0$ for the nodes i with negative weights; $y_i = 1/2$ and $x_i = 0$ for the nodes i in the 2-cycles, and $x_i = y_i = 1$ for the node in the center. There are $K_n = (n-1)/3 \in O(n)$ branches in this graph. We have $UB = K_n M + 2M$ and $OPT = 2M$, which concludes the proof. □

9.3.3 A Model Based on (k, ℓ) Node Separators

We now present an alternative approach to model the MWCS in the space of $(\mathbf{x}, \mathbf{y})$ variables that relies on the constraints that have been recently used by [Fügenschuh and Fügenschuh, 2008] and [Carvajal et al., 2013] to model connectivity in the context of sheet metal design and forest planning, resp. Notice that for an arbitrary pair of distinct nodes (k, ℓ) in G, if ℓ is taken into the solution and k is chosen as root, then either (i) there is a direct arc from k to ℓ, or (ii) at least one node from any (k, ℓ) separator $N \in \mathcal{N}(k, \ell)$ has to be taken into the solution. The latter fact can be stated using the following inequalities that we will refer to as *node-separator constraints*:

$$y(N) - x(N) \geq y_\ell + x_k - 1, \quad \forall k, \ell \in V,\ \ell \neq k,\ N \in \mathcal{N}(k, \ell). \tag{9.9}$$

If the nodes k and ℓ are connected by an arc, then $\mathcal{N}(k, \ell) = \emptyset$, in which case we need to consider the in-degree inequalities (9.6) to make sure k is connected to ℓ. Thus, we can formulate the unrooted MWCS as

$$(CUT)_{k,\ell} \qquad \max \left\{ \sum_{v \in V} p_v y_v \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies (9.4)-(9.6), (9.9) and } (\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \right\}.$$

Inequalities (9.9) can be separated in polynomial time in a support graph that splits nodes into arcs. Given a fractional solution $(\tilde{x}, \tilde{y})$, for each pair of nodes (k, ℓ) such that $\tilde{y}_\ell + \tilde{x}_k - 1 > 0$ we generate a graph $G_{k\ell}$ in which all nodes $i \neq k, \ell$ are replaced by arcs. Arc capacities are then set to 1, except for the arcs associated to nodes, whose

capacities are set to $\tilde{y}_i - \tilde{x}_i$. If the maximum flow that can be sent from k to ℓ in $G_{k\ell}$ is less than $\tilde{y}_\ell + \tilde{x}_k - 1 > 0$, we have detected a violated inequality of type (9.9).

Using the root constraint (9.4), inequalities (9.9) can also be reformulated as follows:

$$y(N) \geq y_\ell + x(N \cup \{k\}) - 1 \;\Rightarrow y(N) + x(V \setminus (N \cup \{k, l\})) \geq y_\ell - x_\ell,$$

which can be interpreted as follows: If node ℓ is in the solution and it is not the root, then for each $k \in V$ such that $\mathcal{N}(k, \ell) \neq \emptyset$ and each $N \in \mathcal{N}(k, \ell)$, either one of the nodes from N is part of the solution, or none of the nodes from $N \cup \{k\}$ is chosen as the root node.

Inequalities (9.9) are quite intuitive, however they are not facet defining. In the next section we will show how the (k, ℓ) node separator constraints can be lifted to obtain facet defining inequalities.

9.3.4 A Model Based on Generalized Node Separator Inequalities

Observe that the inequality (9.9) can be lifted as follows: Assume that $N \in \mathcal{N}(k, \ell)$ also separates another node $k' \neq k$ from ℓ. Since at most one node can be set as a root, the right-hand side of (9.9) can be increased as follows: $y(N) - x(N) \geq y_\ell + x_k + x_{k'} - 1$. In fact, this motivates us to introduce a generalized family of node separator inequalities, that can be obtained by a parallel lifting of (9.9).

Generalized Node-Separator Inequalities Let ℓ be an arbitrary node in V and let $N \in \mathcal{N}_\ell$ be an arbitrary ℓ-separator. Let $W_{N,\ell}$ be the set of nodes i such that there is a directed (i, ℓ)-path in $G - N$. More formally:

$$W_{N,\ell} = \{i \in V \setminus N \mid \exists (i, \ell) \text{ path } P \text{ in } G - N\} \cup \{\ell\}.$$

Then, for any feasible MWCS solution, the following has to be satisfied: if node ℓ is part of a solution, then either the root of the solution is in $W_{N,\ell}$, or, otherwise, at least one of the nodes from N has to be taken. Hence, the following inequalities, that we will refer to as *generalized node-separator inequalities*, are valid for the MWCS:

$$y(N) + x(W_{N,\ell}) \geq y_\ell, \quad \forall \ell \in V,\ N \in \mathcal{N}_\ell \tag{gNSep}$$

Notice that the in-degree inequalities (9.6) are a subfamily of (gNSep): The in-degree inequality can be rewritten as $\sum_{j \in D^-(\ell)} y_j + x_\ell \geq y_\ell$, i.e., they are a special case of the generalized node-separator cuts for $N = D^-(\ell)$ in which case $W_{N,\ell} = \{\ell\}$. In order to see that (gNSep) are lifted inequalities (9.9), notice that (gNSep) can be rewritten as follows:

$$y(N) - x(N) \geq y_\ell + x(V \setminus (N \cup W_{N,\ell})) - 1, \quad \forall \ell \in V,\ N \in \mathcal{N}_\ell.$$

Together with this observation this proves that the following model is a valid MIP formulation for the MWCS:

$$(CUT) \qquad \max \left\{ \sum_{v \in V} p_v y_v \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies (9.4)-(9.5), (gNSep) and } (\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \right\}.$$

Proposition 2. Generalized node-separator inequalities can be separated in polynomial time.

Proof. Consider an auxiliary support graph in which the nodes are splitted as follows: each node $i \in V$ is replaced by an arc (i_1, i_2). All ingoing arcs into i are now connected to i_1, all outgoing arcs from node i are now connected to i_2. In other words, we create a graph $G' = (V', A')$ such that $V' = \{i_1 \mid i \in V\} \cup \{i_2 \mid i \in V\} \cup \{r\}$ (r is an artificial root), $A' = \{(i_2, j_1) \mid (i,j) \in A\} \cup \{(i_1, i_2) \mid i \in V\} \cup \{(r, i_1) \mid i \in V\}$. For a given fractional solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ arc capacities in G' are defined as:

$$cap_{uv} = \begin{cases} \tilde{y}_i, & \text{if } u = i_1, v = i_2, i \in V, \\ \tilde{x}_i, & \text{if } u = r, v = i_1, i \in V, \\ 1, & \text{otherwise.} \end{cases} \tag{9.10}$$

We calculate the maximum flow on G' between r and (ℓ_1, ℓ_2) in G' for a node ℓ such that $\tilde{y}_\ell > 0$. To check whether there are violated inequalities of type (gNSep), it only remains to show that (i) every minimum cut $(\overline{S}, S)$ in G' such that the corresponding flow is less than $\tilde{y}_\ell$ corresponds to a (gNSep) inequality for the given $\ell \in V$ and some $N \in \mathcal{N}_\ell$, or (ii) that a corresponding violated (gNSep) cut can be generated from $(\overline{S}, S)$ in polynomial time. Observe that any minimum cut $(\overline{S}, S)$ in G' which is smaller than $\tilde{y}_\ell$ can be represented as union of arcs adjacent to the root, plus union of arcs of type (i_1, i_2). Hence, each $(\overline{S}, S)$ cut implies the following inequalities:

$$\sum_{(r,j) \in \delta^-(S)} x_j + \sum_{(i_1,i_2) \in \delta^-(S)} y_i \geq y_\ell. \tag{9.11}$$

We can now define a partitioning (U, N, W) of the node set V such that:

$$W = \{i \in V \mid i_1, i_2 \in S\}, \quad N = \{i \in V \mid i_1 \notin S, i_2 \in S\}, \quad U = V \setminus (W \cup N).$$

Rewriting the inequality (9.11), we obtain: $x(W) + y(N) \geq y_\ell$. Observe that $U \neq \emptyset$. Indeed, if $U = \emptyset$ then $N \cup W = V$, but then we have $x(N) + y(W) \geq x(V) = 1 \geq \tilde{y}_\ell$, i.e., such cuts will never be violated. Hence, given the proper partition (U, N, W), the set N is obviously a (k, ℓ) separator for any $k \in U$ (after removing (r, i_1) arcs from G', the arcs $(i_1, i_2) \in \delta^-(S)$ are arc-separators that separate U from the rest of the graph). If W contains only nodes that can reach ℓ in $G - N$, then inequality (9.11) belongs to the (gNSep) family. Otherwise we reverse all arcs in $G - N$ and perform a breadth-first

search from ℓ. All nodes that can be reached from ℓ (notice that they cannot belong to U), by definition, determine the set $W_{N,\ell}$. If the original cut (9.11) was violated, the new one with the left-hand side equal to $y(N) + x(W_{N,\ell})$ will be violated as well. □

9.3.5 Some More Useful Constraints

In this section we present additional constraints that are useful for practically solving MWCS instances.

Connected Component Inequalities In some applications of the MWCS, a K-cardinality constraint is imposed: $\sum_{i \in V} y_i = K$. For a given node $k \in V$, let P_k contain all the nodes that are further than $K-1$ hops away from k. In that case, the following inequalities are valid for the MWCS:

$$x_k + y_\ell \leq 1, \quad \forall \ell \in P_k. \tag{9.12}$$

Rewriting the connected component cuts, we obtain:

$$\sum_{j \neq k} x_j \geq y_\ell, \quad \forall \ell \in P_k,$$

these constraints can be further strengthened by down lifting the coefficients of the left-hand side. Whenever node ℓ is in the solution, then either ℓ is the root, or the root cannot be more than $K-1$ hops away from ℓ. Let W_ℓ be the set of such potential root nodes including ℓ. We have

$$x(W_\ell) \geq y_\ell, \quad \forall \ell \in V.$$

Out-Degree Inequalities: The following set of inequalities state that whenever a node i such that $p_i \leq 0$ is taken into a solution, this is because it leads us to another node with positive weights:

$$y(D^+(i)) \geq y_i, \quad \forall i \in V \text{ s.t. } p_i \leq 0. \tag{9.13}$$

Observe that these constraints are not valid if K-cardinality constraints are imposed.

Symmetry-Breaking Inequalities: In case the input graph is undirected, there exist many equivalent optimal solutions with different orientations. In order to break those symmetries, we can impose the following constraint that chooses the node with the smallest index to be the root of the subgraph:

$$x_j + y_i \leq 1, \quad \forall i < j. \tag{9.14}$$

9.4. Polyhedral Study

Let $\mathcal{P}$ denote the connected subgraph (CS) polytope in the space of $(\mathbf{x}, \mathbf{y})$ variables:

$$\mathcal{P} = \text{conv}\{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \mid (\mathbf{x}, \mathbf{y}) \text{ satisfies } (9.4), (9.5), (\text{gNSep})\}.$$

In this section we compare the proposed MIP formulations with respect to their quality of LP bounds and we show that, under certain conditions, the newly introduced generalized node-separator inequalities are facet defining for the CS polytope.

9.4.1 Theoretical Comparison of MIP Models

Let $\mathcal{P}_{\text{LP}}(.)$ denote the polytope of the LP relaxations of the MIP models presented above obtained by replacing integrality conditions by $0 \leq x_i, y_i \leq 1$, for all $i \in V$, and let $v_{LP}(.)$ be the optimal LP values of the associated MIP relaxations. For the $\mathcal{P}_{\text{LP}}(PCStT)$ polytope, we set $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \mid x_i = z_{ri} \text{ and } (y, z) \in \mathcal{P}_{\text{LP}}(PCStT)\}$. We can show that:

Proposition 3. We have:

1. $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) = \mathcal{P}_{\text{LP}}(CUT) \subsetneq \mathcal{P}_{\text{LP}}(CUT_{k\ell})$ and $\mathcal{P}_{\text{LP}}(CUT) \subsetneq \mathcal{P}_{\text{LP}}(CYCLE)$.
2. Moreover, there exist MWCS instances such that $v_{LP}(CYCLE)/v_{LP}(CUT) \in O(n)$.
3. The polytopes $\mathcal{P}_{\text{LP}}(CYCLE)$ and $\mathcal{P}_{\text{LP}}(CUT_{k\ell})$ are not comparable.

Proof. 1. Assume that $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) = \mathcal{P}_{\text{LP}}(CUT)$: We first show that $Proj_{(x,y)}(\mathcal{P}_{\text{LP}}(PCStT)) \subseteq \mathcal{P}_{\text{LP}}(CUT)$. Let $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ be a feasible solution for the relaxation of the $PCStT$ model, we will show that the solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ such that $\hat{x}_i = \hat{z}_{ri}$ belongs to $\mathcal{P}_{\text{LP}}(CUT)$. Let $\ell \in V$ be an arbitrary node such that $\hat{y}_\ell > 0$, choose some $N \in \mathcal{N}_\ell$ and consider the associated $W_{N,\ell} \subset V$. Let G_d be the corresponding directed instance of the PCStT with the root r (Section 9.3.1). Consider now a cut $(\overline{W}_d, W_d)$ in G_d where $W_d = N \cup W_{N,\ell}$. We have: $\delta^-_{G_d}(W_d) = \{(r,i) \in A_d \mid i \in W_{N,\ell}\} \cup Rest$, where $Rest = \{(j,i) \in A_d \mid j \in \overline{W}_d, i \in N\}$. Observe that $Rest \subseteq \delta^-_{G_d}(N) \subseteq \cup_{i \in N} \delta^-_{G_d}(i)$. Therefore, we have:

$$\hat{\mathbf{y}}(N) = \sum_{i \in N} \hat{\mathbf{z}}(\delta^-_{G_d}(i)) \geq \hat{\mathbf{z}}(\delta^-_{G_d}(N)) \geq \hat{\mathbf{z}}(Rest). \tag{9.15}$$

Since $(\overline{W}_d, W_d)$ is a Steiner cut in G_d, it holds that $\hat{\mathbf{z}}(\delta^-_{G_d}(W_d)) \geq \hat{y}_\ell$. This, together with (9.15) implies:

$$\hat{\mathbf{y}}(N) + \hat{\mathbf{x}}(W_{N,\ell}) \geq \hat{\mathbf{z}}(Rest) + \hat{\mathbf{x}}(W_{N,\ell}) = \hat{\mathbf{z}}(\delta^-_{G_d}(W_d)) \geq \hat{y}_\ell.$$

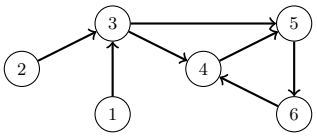


FIGURE 9.2: An example showing that $\mathcal{P}_{\text{LP}}(CUT_{k\ell}) \not\subseteq \mathcal{P}_{\text{LP}}(CYCLE)$. The LP solution $y_4 = y_5 = y_6 = 1$, $y_1 = y_2 = y_3 = x_1 = x_2 = 1/2$ is feasible for the $(CUT_{k\ell})$ model and infeasible for $(CYCLE)$.

To show that $\mathcal{P}_{\text{LP}}(CUT) \subseteq Proj_y(\mathcal{P}_{\text{LP}}(PCStT))$ consider an LP solution $(\check{\mathbf{y}}, \check{\mathbf{x}}) \in \mathcal{P}_{\text{LP}}(CUT)$. We will construct a solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \in \mathcal{P}_{\text{LP}}(PCStT)$ such that $\check{\mathbf{y}} = \hat{\mathbf{y}}$ and $\hat{z}_{rj} = \check{x}_j$, $\forall j \in V$. On the graph G' (see Proof of Proposition 2) with arc capacities of (i_1, i_2) set to $\check{y}_i$ for each $i \in V$, arc capacities of (r, j_1) set to $\check{x}_j$, and capacities set to 1 for the remaining arcs, we are able to send $\check{y}_\ell$ units of flow from the root r to every $\ell_1 \in V'$ such that $\check{y}_\ell > 0$. Let f_{ij}^k denote the amount of flow of commodity k, associated with $k_1 \in V'$, sent along an arc $(i, j) \in A'$. Let $\mathbf{f}$ be the minimal feasible multi-commodity flow on G' (i.e., the effective capacities on G' used to route the flow cannot be reduced without violating the feasibility of this flow). We now define the values of $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ as follows: $\hat{z}_{rj} = \check{x}_j$, $\forall j \in V$ and

$$\hat{z}_{ij} = \begin{cases} \max_{k \in V} f^k_{i_2 j_1}, & i, j \in V \\ \max_{k \in V} f^k_{r j_1}, & i = r, j \in V \end{cases} , \forall (i,j) \in A; \quad \hat{y}_i = \hat{\mathbf{z}}(\delta^-(i)) \, , \forall i \in V.$$

Obviously, the constructed solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ is feasible for the $(PCStT)$ model and, due to the assumption that $\mathbf{f}$ is minimal feasible, it follows that $\check{\mathbf{y}} = \hat{\mathbf{y}}$ and $\check{\mathbf{x}}$ is equivalent to $\hat{\mathbf{z}}$, which concludes the proof.

$\mathcal{P}_{\text{LP}}(CUT) \subsetneq \mathcal{P}_{\text{LP}}(CYCLE)$: Let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be an arbitrary point from $\mathcal{P}_{\text{LP}}(CUT)$. In order to prove that $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \mathcal{P}_{\text{LP}}(CYCLE)$ we only need to show that constraints (9.7) are satisfied (recall that in-degree inequalities (9.6) are contained in (gNSep)). Given the Observation 1, it is sufficient to consider cycles C such that $C \cup D^-(C) \subset V$. Since for any such cycle C the set $D^-(C)$ defines a separator for any node $\ell \in C$, from constraints (gNSep) we have that $\hat{y}(D^-(C)) + \hat{x}(C) \geq \hat{y}_\ell$. For the remaining nodes $j \in C$, $j \neq k$, we apply the bounds $1 \geq \hat{y}_j$. Summing up together these $|C|$ inequalities, we obtain (9.7).

2. Consider the example given in Figure 9.1 for which the (CUT) model finds the optimal solution.

3. The example given in Figure 9.1 shows an instance for which the LP solution is feasible for the $(CYCLE)$ and infeasible for the $(CUT_{k\ell})$ model. The example given in Figure 9.2 shows an instance for which the LP solution is feasible for the $(CUT_{k\ell})$ and infeasible for the $(CYCLE)$ model.

□

9.4.2 Facets of the CS Polytope

In this section we establish under which conditions some of the presented inequalities are facet defining for the CS polytope.

Lemma 2. If G is a strong digraph, then the dimension of the polytope $\mathcal{P}$ is $\dim(\mathcal{P}) = 2n - 1$.

Proof. We will construct the set of $2n$ feasible, affinely independent solutions as follows: Since G is strong, we can find n spanning arborescences by choosing each $i \in V$ as a root. That way, we build n affinely independent solutions. In addition, consider n single node solutions (for each $i \in V$), in which we have $x_i = y_i = 1$ and all remaining $x_j = y_j = 0$, for all $j \neq i$. The matrix obtained by merging the characteristic vectors of these solutions has full rank, $2n$. □

Lemma 3. Trivial inequalities $x_i \geq 0$ are facet defining if G is strong and i is not a cut point in G.

Proof. Consider a family $\mathcal{T}$ of spanning arborescences on the set $V \setminus \{i\}$ in which each $j \neq i$ is taken once as a root. This is possible because $G - i$ remains a strong digraph. There are $n-1$ such solutions, and they are affinely independent. Add now to $\mathcal{T}$ single node solutions, for each $j \in V \setminus \{i\}$. Finally, add to $\mathcal{T}$ a spanning arborescence in G with a root $j \neq i$. The matrix associated to incidence vectors from $\mathcal{T}$ has full rank, $2n - 1$. □

Lemma 4. Trivial inequalities $y_i \leq 1$ are facet defining if G is strong.

Proof. Consider a spanning arborescence T rooted at i. We will then apply a *pruning technique* in order to generate n affine independent feasible MWCS solutions. We start with T in which case $\mathbf{y}$ consists of all ones. We iteratively remove one by one leaves from T, until we end up with a single root node i. Thereby, we generate a family $\mathcal{T}$ of n affinely independent solutions. We then add to $\mathcal{T}$ $n-1$ solutions obtained by choosing a spanning arborescence rooted at j, for all $j \neq i$. The matrix associated to incidence vectors from $\mathcal{T}$, has full rank, $2n - 1$. □

Notice that $y_i \geq 0$ are not facet defining inequalities because $y_i = 0$ implies $x_i = 0$. Similarly, $x_i \leq 1$ do not define facets of $\mathcal{P}$ because they are dominated by $x_i \leq y_i$.

Lemma 5. Coupling inequalities $y_i \geq x_i$ are facet defining if G is strong and i is not a cut point in G.

Proof. Construct a family $\mathcal{T}$ of n affinely independent solutions by applying pruning to a spanning arborescence rooted at i. Add then to $\mathcal{T}$ additional $n-1$ arborescences on the set $V \setminus \{i\}$ in which each $j \neq i$ is taken once as a root (this is possible because

$G - i$ remains strong). The matrix associated to incidence vectors from $\mathcal{T}$, has full rank, $2n - 1$. □

Proposition 4. Given $\ell \in V$ and $N \in \mathcal{N}_\ell$, the associated (gNSep) inequality is facet defining if G is strong, N is a minimal ℓ-node separator and the subgraph induced by $W_{N,\ell}$ ($|W_{N,\ell}| \geq 2$) is strong.

Proof. We prove the result by the indirect method. Let $F(\ell, N) = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{2n} \mid y(N) + x(W_{N,\ell}) = y_\ell\}$. Consider a facet defining inequality of the form $\mathbf{ax} + \mathbf{by} \geq a_0$. We will show that if all points in $F(\ell, N)$ satisfy

$$\mathbf{ax} + \mathbf{by} = a_0, \tag{9.16}$$

then (9.16) is a positive multiple of (gNSep). Consider $\ell' \in W$, $\ell' \neq \ell$. A path from ℓ to ℓ', completely contained in $W_{N,\ell}$ and rooted at ℓ exists in G ($W_{N,\ell}$ is strong) and it is a feasible MWCS solution that belongs to $F(\ell, N)$. Let $(\mathbf{x}^1, \mathbf{y}^1)$ be the characteristic vector of this path. A subpath obtained after removing ℓ' from this path, also rooted at ℓ, is another feasible solution from $F(\ell, N)$, and let $(\mathbf{x}^2, \mathbf{y}^2)$ be the corresponding characteristic vector. We have: $\mathbf{ax}^1 + \mathbf{by}^1 - \mathbf{ax}^2 - \mathbf{by}^2 = 0$. Therefore we have $b'_\ell = 0$, for all $\ell' \in W$, $\ell' \neq \ell$. Consider now a node $k \in U = V \setminus (N \cup W_{N,\ell})$. To show that $b_k = 0$, for all $k \in U$, we distinguish the following cases:
(1) If $D^-(k) \cap U \neq \emptyset$, then there exists an arc (k', k), $k' \in U$ that builds a feasible MWCS solution B from $F(\ell, N)$. Also, the single node solution $B' = \{k'\}$ belongs to $F(\ell, N)$. After subtracting the equations (9.16) with the substituted characteristic vectors of B and B', we obtain $b_k = 0$.
(2) If there exists an arc $(i, k) \in A$ for some $i \in N$, then, consider a path P from i to ℓ that does not cross $N \cup U$ (such P exists because N is minimal) and a path $P' = P \cup \{(i, k)\}$, in both of them we set i as root. Both P and P' belong to $F(\ell, N)$. After subtracting the equations (9.16) with the substituted characteristic vectors of P and P', we obtain $b_k = 0$.
(3) Finally, if there exists an arc $(j, k) \in A$ for some $j \in W_{N,\ell}$, we consider a path Q from ℓ to j in $W_{N,\ell}$ (such path exists because $W_{N,\ell}$ is strong) and a path $Q' = Q \cup \{(j, k)\}$. Both Q and Q' belong to $F(\ell, N)$. After subtracting the equation (9.16) with the substituted characteristic vectors of Q and Q', we obtain $b_k = 0$. Hence, the equation (9.16) can be rewritten as $\mathbf{ax} + \sum_{i \in N \cup \{\ell\}} b_i x_i = a_0$. Notice that a single node solution $\{k\}$ belongs to $F(\ell, N)$, for each $k \in U$. By plugging the associated vector into (9.16), it follows that $a_k = a_0$, for all $k \in U$. Consider now two spanning arborescences in $W_{N,\ell}$, one rooted at ℓ, the other rooted at arbitrary $\ell' \neq \ell$ (this is possible, because $W_{N,\ell}$ is strong). After subtracting the equation (9.16) with the substituted characteristic vectors of those two arborescences, we obtain $a_{\ell'} = a_\ell = \alpha$, for all $\ell' \in W_{N,\ell}$. Since $N \in \mathcal{N}_\ell$ and it is minimal, for each $i \in N$ there exist $k \in U$ such that there exist a path P_k from k to ℓ that crosses N exactly at the node i. Let P'_k be

a subpath of P_k from i to ℓ. Both paths belong to $F(\ell, N)$ and after subtracting the associated equations (9.16), it follows that $a_i = a_k$, and hence $a_i = a_0$, for all $i \in N$.

So far, (9.16) can be rewritten as $a_0 x(\overline{W}_{N,\ell}) + \alpha x(W_{N,\ell}) + \sum_{i \in N \cup \{k\}} b_i y_i = a_0$. After plugging in the characteristic vector of P'_k into this equation, it follows that $a_0 + b_i + b_\ell = a_0$, and therefore we have $b_i = -b_\ell = \beta$, for all $i \in N$. Equation (9.16) becomes now $a_0 x(\overline{W}_{N,\ell}) + \alpha x(W_{N,\ell}) + \beta y(N) - \beta y_\ell = a_0$. Notice that solution $\{\ell\}$ also belongs to $F(\ell, N)$, which implies that $\alpha - \beta = a_0$. Finally, substituting a_0 in the previous equation, and using the equation (9.4), $x(V) = 1$, we end up with the following form of (9.16):

$$\beta[-x(\overline{W}_{N,\ell}) + y(N) - y_\ell = -1],$$

which together with equation (9.4) concludes the proof. □

9.5. Computational Results

For testing the computational performance of the presented formulations we have considered both directed and undirected MWCS instances. The $(CYCLE)$ model of [Backes et al., 2011] has been developed for directed graphs (regulatory networks) with K-cardinality constraints, i.e., any feasible solution has to be comprised by exactly K nodes (for a given $K > 1$). Executables of this implementation are available online [see GeneTrail]. For the $(PCStT)$ and (CUT) models we have developed our own B&C implementations that work with and without cardinality constraints. The real-world instances used in [Backes et al., 2011] require K-cardinality constraints. Therefore, in the part of our computational study conducted on digraphs, we impose cardinality constraints for all three models, $(PCStT)$, (CUT) and $(CYCLE)$. For the other set of instances we take the size of the unconstrained optimal solution (obtained by the (CUT) model) and provide the corresponding value of K as input to the $(CYCLE)$ model.

In the following, we describe (i) components of the designed B&C algorithms and some implementation details, (ii) a testbed used for the experiments, and (iii) an extensive analysis of the obtained results.

9.5.1 Branch-and-Cut Algorithms

Separation of Inequalities For the $(PCStT)$ model, connectivity inequalities (9.2) are separated within the B&C framework by means of the maximum flow algorithm given by [Cherkassky and Goldberg, 1995]. The separation problem is solved on a support graph whose arc capacities are given by the current LP value of $\mathbf{z}$ variables. We randomly select a terminal $v \in V$ such that $p_v > 0$ and $y_v > 0$, and calculate

the maximum flow between the artificial root and v, and insert the corresponding constraint (9.2), if violated.

For the (CUT) formulation, the separation of (gNSep) is performed by solving the maximum flow problems as described in the proof of Proposition 2, with arc capacities given by (9.10).

In all cases, instead of adding a single violated cut per iteration, we use *nested*, *back-flow* and *minimum cardinality* cuts [see Koch and Martin, 1998, Ljubić et al., 2006] to add as many violated cuts as possible. We restrict the number of inserted cuts within each separation callback to 25.

Primal Heuristic Our primal heuristic finds feasible solutions using the information available from the current LP solution in a given node of the branch-and-bound tree. Although we develop two different B&C algorithms, derived from two MIP models, the embedded primal heuristics are based on the same idea. We select a subset of potential "key-players" (nodes with a positive outgoing degree and with sufficiently large $\mathbf{y}$ values) and run a restricted breadth-first search (BFS) from each of them. Out of the constructed connected components, i.e., feasible solutions of the MWCS, we select the one with the largest total weight.

MIP Initialization We initialize the $(PCStT)$ model with the root out-degree constraints (9.3). For the undirected MWCS, we also add symmetry-breaking constraints (similar to (9.14)) and inequalities $z_{ji} + z_{ij} \leq y_i$, for all $e : \{i, j\} \in E$ since they avoid too frequent calls of the maximum flow procedure. For the variants where no cardinality constraint is defined, we also include the flow-balance constraints: $z(\delta^-(i)) \leq z(\delta^+(i))$, for all $i \in V$ such that $p_i \leq 0$. These constraints ensure that a node with non-positive weight can not be a leaf in an optimal PCStT solution.

We initialize the (CUT) model with the constraints (9.4), (9.5), (9.6). For the cases where no cardinality constraint is imposed, the out-degree constraints (9.13) are also included. Finally, the symmetry-breaking constraints (9.14) are added for the undirected case.

Implementation The proposed approaches were implemented using CPLEX$^{\text{TM}}$12.3 and Concert Technology. All CPLEX parameters were set to their default values, except the following ones: (i) CPLEX cuts were turned off, (ii) CPLEX heuristics were turned off, (iii) CPLEX preprocessing was turned off, (iv) the time limit was set to 1800 seconds [except for the instances from Backes et al., 2011], and (v) higher branching priorities were given to $\mathbf{y}$ variables, in the case of the $(PCStT)$ models, and to $\mathbf{x}$ variables, in the case of the (CUT) model. All the experiments were performed on a Intel Core2 Quad 2.33 GHz machine with 3.25 GB RAM, where each run was performed on a single processor.

9.5.2 Benchmark Instances

We have considered two sets of benchmark instances arising from applications in systems biology and from network design.

System Biology Instances We have considered instances used in [Dittrich et al., 2008] and [Backes et al., 2011]. In [Dittrich et al., 2008], only a single protein-protein interaction network is considered. The instance is presented as an undirected graph comprised by 2034 nodes (proteins) and 8399 edges (interactions). The considered protein-protein interaction network corresponds to a well studied human one and the protein scores come from a lymphoma microarray dataset (LYMPH). The instance is available at [PlanetLisa].

In [Backes et al., 2011], six instances of regulatory networks, i.e., directed graphs, were considered. These instances have the same underlying network (KEGG human regulatory network of protein complexes), which is a graph comprised by 3917 nodes and 133 310 arcs. The differences between the six benchmark instances of this set are the scores associated to the proteins (or protein complexes) which depend on the pathogenic process under consideration. All the instances are available online [see GeneTrail]. For providing a valid comparison with the method proposed in [Backes et al., 2011], it is necessary to impose cardinality constraints to the solutions. Values $K \in \{10, 11, \ldots, 25\}$ are considered. This leads to 16 different instances for each of the six different score settings.

Network Design Instances These are Euclidean random instances which are generated as proposed by Johnson, Minkoff, and Phillips in their paper on the Prize-Collecting Steiner Tree Problem [Johnson et al., 2000]. The topology of these instances is similar to street networks. First, n nodes are randomly located in a unit Euclidean square. A link between two nodes i and j is established if the Euclidean distance d_{ij} between them is no more than $\alpha/\sqrt{n}$, for a fixed $\alpha > 0$.

To generate node weights, we performed the following procedure: $\delta\%$ of the nodes are randomly selected to be associated with non-zero weights. Out of them, $\epsilon\%$ are associated with a weight taken uniformly randomly from $[-10, 0]$ and the remaining ones are associated with a weight taken uniformly randomly from $[0, 10]$.

When generating these instances we do not impose whether links are directed or not. When reading the input files we define if the link between i and j corresponds to an edge $e : \{i, j\}$ or to an arc $a : (i, j)$. This allows us to use the same set of instances for both, the directed and the undirected case.

For the computational experiments we considered $n \in \{500, 750, 1000, 1500\}$, $\alpha \in \{0.6, 1.0\}$, $\delta \in \{0.25, 0.50, 0.75\}$, $\epsilon \in \{0.25, 0.50, 0.75\}$. This leads to 18 instances for each fixed value of n.

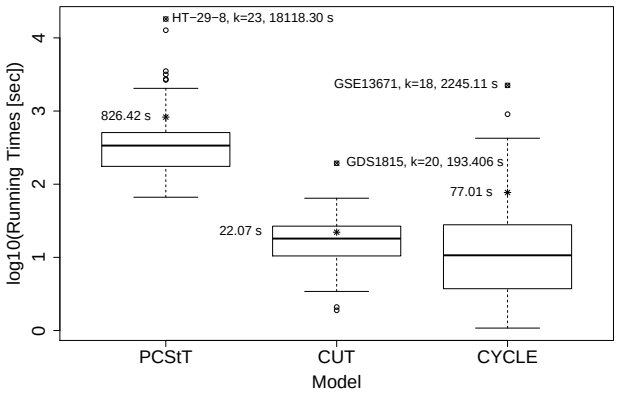


FIGURE 9.3: Box plots of $\log_{10}$-values of the running times [sec] (instances from [Backes et al., 2011], $K \in \{10, \ldots, 25\}$).

9.5.3 Algorithmic Performance

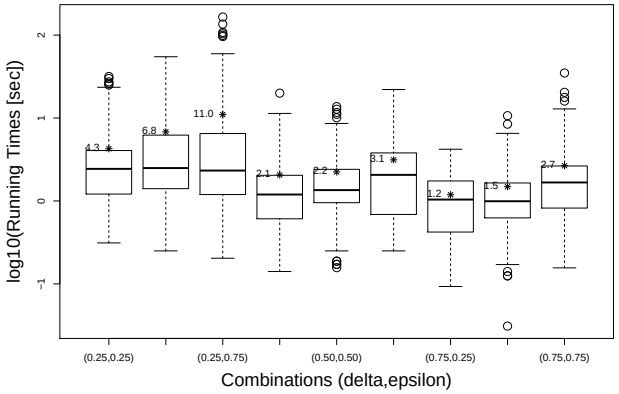
MWCS on Digraphs For this study, we consider the instances GSE13671, GDS1815, HT-29-8, HT-29-24, HT-116-8 and HT-116-24 from [Backes et al., 2011], and our randomly generated instances.

In Figure 9.3, using the box plots we show the $\log_{10}$-values of the running times for the three approaches considering all instances of [Backes et al., 2011] and all values of K. There are $16 \times 6 = 96$ problems in total for each approach. The values marked with an asterisk correspond to the $\log_{10}$-values of the mean running time (shown as the label next to the asterisk). The values marked with symbol $\times$ correspond to the $\log_{10}$-values of the maximum running times (the label next to it shows the name of the instance, K, and the running time). The obtained results indicate that, for this group of instances, $(PCStT)$ is the approach with the worst performance since most of the running times are at least one order of magnitude larger than the ones of the other two approaches. When comparing (CUT) and $(CYCLE)$, one can observe that the distribution of the running times of the $(CYCLE)$ model has a larger dispersion (the *box* is wider) and its outliers are almost one order of magnitude larger than the maximum running times of the (CUT) model. In a few cases however the $(CYCLE)$ model solves some instances faster than the (CUT) model (which can be seen from the minimum values and the values in the first-quartile). Overall, the mean value of the running times of the (CUT) model is 22 sec which is almost three times smaller than the mean running time of the $(CYCLE)$ model (77 sec). The value of the maximum running time of the (CUT) model is 193 sec which is more than 10 times smaller than the maximum running time of the $(CYCLE)$ model (2245 sec, reached for $K = 18$ for the instance GSE13671, see Figure 9.3). The fact that the box of the (CUT) model is considerably narrower than the box of the $(CYCLE)$ model, indicates that the (CUT) approach is more robust regarding the variation of the scores of protein complexes and the value of K.

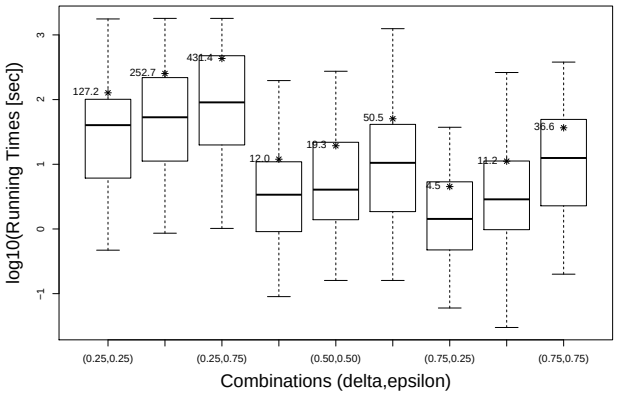
In Table 9.1 we report for each instance from [Backes et al., 2011] the average values (over all $K \in \{10, \ldots, 25\}$) of the running times and the average number of cuts added for each of the $(PCStT)$, (CUT) and $(CYCLE)$ models (columns Time(sec), #(9.2), #(gNSep) and #(9.7), respectively). In column δ we show the fraction of nodes with a score different than 0 and in column ϵ the fraction of them with a negative score. The results indicate that the performance of the $(CYCLE)$ model strongly depends on the instances under consideration (the average running times of GSE13671 are two orders of magnitude larger that the ones of HT-116-8), which also explains the dispersion shown in Figure 9.3. Likewise, for the $(PCStT)$ model, the average running time for the instance HT-29-8 is an order of magnitude larger than for the instance GSE13671. In contrast to the unstable performance of $(PCStT)$ and $(CYCLE)$ models, the (CUT) model seems to be more independent on the type of considered instances. From the same table we may conclude that the number of cuts needed to prove the optimality is one order of magnitude smaller for the (CUT) model than for the other two models. This means that the (gNSep) cuts are more effective in closing the gap than the (9.7) and (9.2) cuts. Regarding δ and ϵ, it seems that the (CUT) model is not sensitive to their values, while the $(CYCLE)$ model performs better when ϵ is smaller.

For the set of Euclidean network instances, running times of the (CUT) and $(CYCLE)$ model are given in Figure 9.4(a) and 9.4(b), respectively (for many instances we reached the time-limit for the $(PCStT)$ model, so we do not consider it here). This time we group instances according to different combinations of (δ, ϵ) values. Each box contains $16 \times 8 = 128$ values obtained for the settings: $K \in \{10, \ldots, 25\}$, $n \in \{500, 750, 1000, 1500\}$ and $\alpha \in \{0.6, 1.0\}$. Comparing Figure 9.4(a) and 9.4(b) we observe that although the average running times (marked with asterisk) of the (CUT) model are in general one order of magnitude smaller than those of the $(CYCLE)$ model, both of them present a similar pattern: (i) For a given δ, the increase of ϵ from 0.25 to 0.75 produces a worsening of the algorithmic performance. This worsening is visible not only in the increase of the running times, but also in their higher dispersion (wider boxes and more outliers). Increasing ϵ (for a fixed δ), means that a larger proportion of nodes has a negative weight; since our goal is to find a connected component of exactly K nodes the more nodes with negative weight, the more difficult is the task of reaching the "attractive" nodes that lead to a better solution. (ii) On the other hand, increasing δ from 0.25 to 0.75 produces an improvement of the algorithmic performance, i.e., the more nodes with non-zero weights, the easier the problems. One possible reason for this could be the symmetries induced by a large portion of nodes with zero weight (as it is the case for $\delta = 0.25$). Hence, by decreasing this portion (i.e., increasing δ) the cutting-planes that are added through the separation become more effective, and the primal heuristic is able to find more diverse, and eventually better, incumbent solutions.

MWCS on Undirected Graphs For this computational comparison we do not impose cardinality constraints. In order to be able to perform a comparison with the



(a) Influence of δ and ϵ on the performance of the (CUT) model (random instances, $K \in \{10, \ldots, 25\}$).



(b) Influence of δ and ϵ on the performance of the ($CYCLE$) model (random instances, $K \in \{10, \ldots, 25\}$).

FIGURE 9.4: Dependance of the running times on the (δ, ϵ) settings.

($CYCLE$) model that requires a digraph G and K as its input, we run the ($CYCLE$) model with (i) G transformed into a digraph, and (ii) with the value of K set to be the size of the optimal unconstrained MWCS solution (obtained by, e.g., the (CUT) model). For these graphs we impose a time limit of 1800 seconds. Figure 9.5 shows the performance profile of the three approaches regarding the total running time. Figure 9.6 shows the performance profile of the achieved gaps within this time limit. We observe that also in the case of undirected graphs, the (CUT) approach significantly outperforms the ($CYCLE$) and the ($PCStT$) approach: While the (CUT) approach produces solutions of less than 1% of gap in almost 100% of the instances, the ($PCStT$) approach produces solutions with more than 15% of gap in more than 40% of the instances. The ($CYCLE$) approach solves about 50% of instances to optimality, with most of the gaps of the unsolved instances being below 15%.

In Table 9.2 we provide more details on these results. Each row corresponds to a fixed value of n, with 18 different instances obtained by varying δ, ϵ and α. Column #NOpt indicates how many out of those 18 instances were not solved to optimality

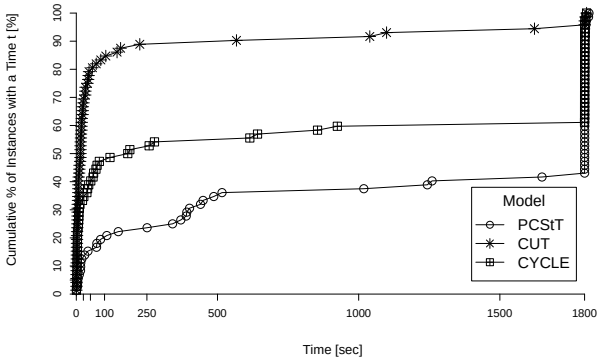


FIGURE 9.5: Performance profile of running times on random undirected instances.

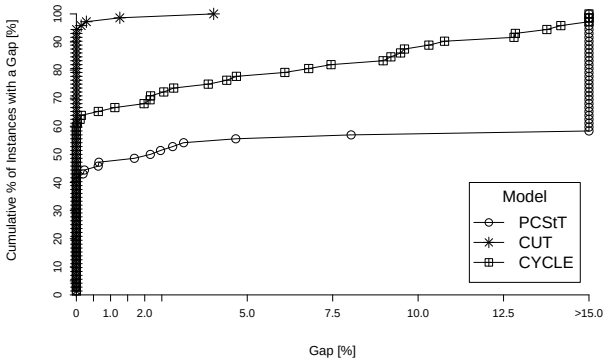


FIGURE 9.6: Performance profile of final gaps (%) on random undirected instances.

within the imposed time limit of 1800 seconds. For a given n, and for each of the three approaches we additionally report on the following values: the average running time (column Time(sec)); the average gap of those instances that were not solved to optimality (column Gap(%)), and the average number of inserted cutting planes (columns #(9.2), #(gNSep) and #(9.7), respectively). These results show that the (CUT) model is by far more effective than the $(CYCLE)$ model for this group of instances. The average running times of the (CUT) model are one order of magnitude smaller than those of the $(PCStT)$ and $(CYCLE)$ model. All but four instances can be solved by the (CUT) model to optimality, while in the case of the $(CYCLE)$ and $(PCStT)$ model, 29 and 42 instances remain unsolved, respectively. The number of cutting planes of type (gNSep) needed to close the gap is one order magnitude smaller than the number of cuts of type (9.7) or (9.2).

So far, it seems clear that for the considered instances the (CUT) model significantly outperforms the $(PCStT)$ approach. However for the LYMPH instance studied in [Dittrich et al., 2008], for which $\delta = 1.0$ and $\epsilon = 0.97$, the $(PCStT)$ model takes only 3.19 seconds to find the optimal solution while the $(CYCLE)$ model takes 15.56 seconds, and the (CUT) model 50.70 seconds. The optimal solution, whose objective value is

			(*PCStT*)		(*CUT*)		(*CYCLE*)	
Instance	δ	ϵ	Time(sec)	#(9.2)	Time(sec)	#(gNSep)	Time(sec)	#(9.7)
GSE13671	0.89	0.73	176.11	1206	17.85	97	341.95	3754
GDS1815	0.92	0.64	878.63	3565	46.09	225	37.95	1264
HT-29-8	0.92	0.66	2846.36	5400	22.03	182	14.17	178
HT-29-24	0.92	0.61	196.56	1292	11.40	61	60.59	1330
HT-116-8	0.92	0.54	623.10	2214	15.26	108	3.21	129
HT-116-24	0.92	0.55	237.78	1149	19.82	93	4.19	130
Average			826.42	2471	22.07	128	77.01	1131

TABLE 9.1: Average values for instances from [Backes et al., 2011] ($K \in \{10, \ldots, 25\}$).

70.2, is comprised by 37 nodes with positive weight and 9 with negative weight. It is not easy to derive a concrete answer of why, for this particular instance, the (*PCStT*) model is faster than the (*CUT*) model. The following two factors could be responsible for this behavior: (i) the sparsity of the graph (the number of edges is approximately four times the number of nodes, while in random instances this ratio is almost 10) which means that the number of $\mathbf{z}$ variables is not too large, and (ii) there are significantly less symmetries due to the fact that there are no nodes with zero weight. These factors might explain why, in this particular case, it becomes easier to solve the problem with the prize-collecting Steiner tree reformulation, rather than directly looking for a connected component that maximizes the objective function.

		(*PCStT*)				(*CUT*)				(*CYCLE*)			
#nodes	#arcs	Time	Gap(%)	#(9.2)	#NOpt	Time	Gap(%)	#(gNSep)	#NOpt	Time	Gap(%)	#(9.7)	#NOpt
500	4558	677.24	>15.00	1055	5	15.30	–	69	0	615.36	5.50	4289	6
750	7021	1243.57	>15.00	1552	11	108.78	1.27	99	1	471.68	2.64	1721	4
1000	9108	1304.76	>15.00	1955	12	150.03	0.29	201	1	990.84	6.76	3176	9
1500	14095	1526.41	>15.00	2021	14	453.82	2.08	373	2	1086.19	10.55	2139	10

TABLE 9.2: Average values for different values of n (random instances, $\alpha \in \{0.6, 1.0\}$, $\delta, \epsilon \in \{0.25, 0.50, 0.75\}$, 18 problems per each n).

9.6. Conclusion

Our work was motivated by the wide range of applications of the MWCS and a recent work in [Backes et al., 2011] who were the first ones to propose a MIP model for the MWCS derived on the set of node variables only. In this work we were able to provide a tight MIP model that outperforms the model from [Backes et al., 2011] both theoretically and computationally. The new model also works on the space of node variables and is valid for all previously studied variants of the MWCS (cardinality constrained, budget constrained and undirected/directed one). We have studied the CS polytope and we have shown that the newly introduced family of generalized node-separator inequalities is facet defining. Our computational study has shown that the new approach outperforms the previously proposed ones, in particular if the inputs are digraphs with non-empty subsets of zero-weight nodes.

Chapter 10

The Rooted Maximum Node-Weight Connected Subgraph Problem

10.1. Introduction

In this work we study a variant of the *connected subgraph problem* in which we are given a graph with a pre-specified root node (and possibly an additional set of terminals). Nodes of the graph are associated with (not necessarily positive) weights. The goal is to find a connected subgraph containing the root and the terminals that maximizes the sum of node-weights. In addition, a budget constraint may be imposed as well: in this case, each node is additionally associated with a non-negative cost, and the cost of connecting the nodes is not allowed to exceed the given budget. Both problem variants are NP-hard, unless all node weights are non-negative and no budget is imposed, in which case the problem is trivial. The problem is called the *Rooted Maximum Node-Weight Connected Subgraph Problem* (RMWCS), or the RMWCS with *Budget Constraint* (B-RMWCS), respectively.

The problem has been introduced by [Lee and Dooly, 1998] in the context of the design of fiber-optic communication networks over time, where the authors refer to the problem as the *constrained maximum weight connected graph problem*. The authors impose K-cardinality constraints, i.e., they search for a connected subgraph containing K nodes (including a predetermined root) that maximizes the collected node-weights. Obviously, K-cardinality constraints are a special form of the budget constraints in which every node is associated a cost equal to one, and the budget is equal to K.

A budgeted version arises in the wildlife conservation planning, where the task is to select land parcels for conservation to ensure species viability, also called *corridor*

design [see, e.g., Conrad et al., 2012, Dilkina and Gomes, 2010]. Here, the nodes correspond to land parcels, their weights are associated with the habitat suitability, and node costs are associated with land value. The task is to design wildlife corridors that maximize the suitability with a given limited budget. Also in forest planning, the connected subgraph arises as subproblem, e.g., for designing a contiguous site for a natural reserve or for preserving large contiguous patches of mature forest [Carvajal et al., 2013]. [Moss and Rabani, 2007] have proposed an $O(\log n)$ approximation algorithm for the B-RMWCS with non-negative node-weights, where n is the number of nodes in the graph. For more details on the problems related on the RMWCS, see e.g., the literature review given in [Dilkina and Gomes, 2010].

In this chapter we will address the RMWCS in digraphs as well. This is motivated by some applications in systems biology where regulatory networks are represented using (not necessarily bidirected) digraphs and with node weights that can also be negative. The goal is to find a rooted subgraph in which there is a directed path from the root to any other node that maximizes the sum of node weights. In systems biology, the roots are frequently referred to as "seed genes" as they are assumed to be involved in a particular disease. In [Backes et al., 2011], for example, the authors search for the connected subgraph in a digraph without a prespecified root node (i.e., determination of the seed gene, also called the key player, is part of the optimization process). To solve the problem of [Backes et al., 2011] one can, for example, iterate over a set of potential key players, solve the corresponding RMWCS and choose the best solution.

Our Contribution. Previously studied mixed integer programming (MIP) formulations for the (B-)RMWCS use arc and possibly flow variables to model the problem [see Dilkina and Gomes, 2010]. In this work we propose three new MIP models for the (B-)RMWCS derived in the natural space of node variables. We first provide a theoretical comparison of the quality of lower bounds of these models. We also show that one of our models which is based on the concept of *node separators*, preserves the tight LP bounds of the previously proposed *cut set* model of [Dilkina and Gomes, 2010]. In the second part of the paper we study the rooted connected subgraph polytope (in the natural space of node variables) and show under which conditions the node separator inequalities are facet-defining. In an extensive computational study, we compare the node-separator and the cut-set model on a set of benchmark instances for the wildlife corridor design problem used in [Dilkina and Gomes, 2010] and on a set of network design instances.

Outline of the Paper. Three new MIP models for the (B-)RMWCS are proposed in Section 10.2. A comparison of the MIP models and results regarding the facets of the rooted connected subgraph polytope are given in Section 10.3 and computational results are presented in Section 10.4.

10.2. MIP Formulations for the RMWCS

In this section we present three new MIP models for the RMWCS and its budget-constrained variant. Before that, we first review the model recently proposed by [Dilkina and Gomes, 2010] which is based on the reformulation of the problem into the (budget-constrained) Steiner arborescence problem. The latter model is derived on the space of arc variables, while the remaining ones are defined in the natural space of node variables.

Since every RMWCS on undirected graphs can be considered as the same problem on digraphs (by replacing every edge with two oppositely directed arcs), in the remainder of this chapter we will present the more general results for digraphs. The corresponding results for undirected graphs can be easily derived from them.

Definitions and Notation.

Formally, we define the RMWCS as follows: Given a digraph $G = (V \cup \{r\}, A)$, with a root r, a set of terminals $R \subset V$, and node weights $p : V \to \mathbb{R}$, the RMWCS is the problem of finding a connected subgraph $T = (V_T, A_T)$, that spans the nodes from $\{r\} \cup R$ and such that every node $j \in V_T$ can be reached from r by a directed path in T, and that maximizes the sum of node weights $p(T) = \sum_{v \in V_T} p_v$. Additionally, in the B-RMWCS, node costs $c : V \to \mathbb{R}^+$ and a budget limit $B > 0$ are given. The goal is to find a connected subgraph T that maximizes $p(T)$ and such that its cost does not exceed the given budget, i.e., $c(T) = \sum_{v \in V_T} c_v \leq B$.

A set of vertices $S \subset V$ $(S \neq \emptyset)$ and its complement $\overline{S} = V \setminus R$, induce two directed cuts: $(S, \overline{S}) = \delta^+(S) = \{(i,j) \in A \mid i \in S, j \in \overline{S}\}$ and $(\overline{S}, S) = \delta^-(S) = \{(i,j) \in A \mid i \in \overline{S}, j \in S\}$. For a set $C \subset V$, let $D^-(C)$ denote the set of nodes outside of C that have ingoing arcs into C, i.e., $D^-(C) = \{i \in V \setminus C \mid \exists (i,v) \in A, v \in C\}$.

A digraph G is called strongly connected (or simply, *strong*) if for any two distinct nodes k and ℓ from V, there exists a (k,ℓ) path in G. A node i is a *cut point* in a strong digraph G if there exists a pair of distinct nodes k and ℓ from V such that there is no (k,ℓ) path in $G - i$. A node i is a *cut point with respect to* r if there exists a node $k \neq i, r$ such that there is no (r,k) path in $G - i$. For two distinct nodes k and ℓ from V, a subset of nodes $N \subseteq V \setminus \{k,\ell\}$ is called (k,ℓ) *(node) separator* if there exists a (k,ℓ) path in G and after eliminating N from V there is no (k,ℓ) path in G. A (k,ℓ) separator N is *minimal* if $N \setminus \{i\}$ is not a (k,ℓ) separator, for any $i \in N$. Let $\mathcal{N}(k,\ell)$ denote the family of all (k,ℓ) separators. Obviously, if $\exists (k,\ell) \in A$ or if ℓ is not reachable from k, we have $\mathcal{N}(k,\ell) = \emptyset$.

For variables $\mathbf{a}$ defined on a finite set F, we denote by $a(F')$ the sum $\sum_{i \in F'} a_i$ for any subset $F' \subseteq F$. Throughout the paper, let the graph $G = (V \cup \{r\}, A)$, $n = |V|$, and $m = |A|$.

10.2.1 Directed Steiner Tree Model of [Dilkina and Gomes, 2010]

[Dilkina and Gomes, 2010] propose to solve the B-RMWCS as a budget-constrained directed Steiner tree problem rooted at r. Their models are based on the observation that it is sufficient to search for a subtree (subarborescence) since no costs are associated to arcs in G, hence every solution containing cycles can be reduced without changing the weight. It is sufficient to use arc variables to model the problem since in a directed tree, the in-degree of every node is equal to one, so that the objective function can be expressed as $\max \sum_{i \in V} p_i z(\delta^-(i))$, where z are binary variables associated with the arcs of A that encode the subarborescence. [Dilkina and Gomes, 2010] proposed three MIP models for the B-RMWCS. Two of them are flow based formulations (a single-commodity flow and a multi-commodity flow based one). The authors showed that the flow-based formulations are computationally outperformed by the cut-set model which is presented below.

We further use a set of auxiliary binary variables y for the vertex set V, where y_i will be equal to one if node i is part of the subtree, and zero, otherwise. In other words, we basically perform the substitution $y_i = z(\delta^-(i))$. The set of feasible B-RMWCS solutions can be described using inequalities (10.1)-(10.4). Constraints (10.1) and (10.2) ensure that the solution is a Steiner arborescence rooted at r, equations (10.3) make sure that all terminals are connected and (10.4) is the budget constraint:

$$z(\delta^-(i)) = y_i \qquad \forall i \in V \setminus \{r\} \tag{10.1}$$

$$z(\delta^-(S)) \geq y_k \qquad \forall k \in S,\ \forall S \subseteq V \setminus \{r\},\ S \neq \emptyset \tag{10.2}$$

$$y_i = 1 \qquad \forall i \in R \tag{10.3}$$

$$\mathbf{c}^T y \leq B \tag{10.4}$$

Constraints (10.2), also known as *cut* or *connectivity inequalities* ensure that there is a directed path from the root r to each node k such that $y_k = 1$. *In-degree* constraints (10.1) guarantee that the in-degree of each vertex of the arborescence is equal to one. Thus, the rooted Steiner arborescence model for the B-RMWCS (denoted by (SA_r)) is given as

$$(SA)_r \quad \max \left\{ p^T y \mid (\mathbf{y}, \mathbf{z}) \text{ satisfies (10.1)-(10.4)},\ (\mathbf{y}, \mathbf{z}) \in \{0,1\}^{n+m} \right\}.$$

We notice that in [Ljubić et al., 2006] these sets of constraints and the transformation into the directed Steiner tree were used for solving the Prize-Collecting Steiner Tree problem (PCStT). A connection between the PCStT and the unrooted MWCS has been observed by [Dittrich et al., 2008]: the authors showed that the unrooted MWCS can be transformed into the PCStT and used the branch-and-cut approach from [Ljubić et al., 2006] to solve the MWCS on a large protein-protein interaction network. Consequently, the same relation holds for the rooted MWCS as well.

The previous model uses node and arc variables ($\mathbf{y}$ and $\mathbf{z}$) given that it relies on a transformation into the Steiner arborescence problem. However it seems more natural to find a formulation based only in the space of $\mathbf{y}$ variables since no arc costs are involved in the objective function. In the next section we will discuss several models that enable elimination of arc variables in the MIP models.

10.2.2 Node-Based Formulations for the RMWCS

We now propose three MIP models that are derived in the natural space of y variables defined as above. We search for an arborescence rooted at r, but this time, we avoid explicit use of arc variables.

Model Based on Subtour Elimination Constraints. This model is an adaptation of the model by [Backes et al., 2011] that was recently proposed for the unrooted MWCS on directed graphs. The following inequalities will be called the *in-degree constraints*:

$$y(D^-(i)) \geq y_i, \quad \forall i \in V \setminus (\{r\} \cup D^+(r)) \tag{10.5}$$

They ensure that, whenever a node i is taken into a solution, at least one of its incoming neighbors has to be in the solution as well (notice that we do not need to impose this constraint for the outgoing neighbors of the root node). Constraints (10.5) however do not guarantee that the obtained solution is connected to the root. Let $\mathcal{C}$ denote the family of all directed cycles in G that do not contain the root node and are not "neighbors" of the root, i.e.:

$$\mathcal{C} = \{C \mid C \text{ is a cycle in } G, \text{ s.t. } r \notin C, \text{ and } r \notin D^-(C)\}.$$

In order to ensure connectivity of the solution, [Backes et al., 2011] add the following constraints, that we will refer to as the *subtour elimination constraints*:

$$y(C) - y(D^-(C)) \leq |C| - 1, \quad \forall C \in \mathcal{C}. \tag{10.6}$$

These constraints state that for each cycle $C \in \mathcal{C}$ whose node set is contained in the solution, at least one of the neighboring nodes outside of that cycle needs to belong to the solution as well. The model, that we will denote by $CYCLE_r$ reads as follows:

$$(CYCLE_r) \quad \max \left\{ p^T y \mid \mathbf{y} \text{ satisfies (10.3)-(10.6)}, \ \mathbf{y} \in \{0,1\}^n \right\}.$$

A Flow-Based Model. Alternatively to the previous model, to ensure connectivity, we can use multi-commodity flows where the available arc capacities are defined as the minimum node capacities at each end of an arc. Finding a feasible solution now means allocating node capacities that will enable to send one unit of flow from the root to each

of the nodes taken into the subnetwork. In this context, constraints (10.5) and (10.6) can be replaced by the following set of constraints that ensure that there is enough capacity on the nodes so that a unit of flow can be sent from the root to any other node $i \in V \setminus \{r\}$ with $y_i = 1$. These constraints state that (i) whenever an arc is part of a feasible solution of the RMWCS, both of its end nodes are included into the solution and (ii) the induced subgraph is connected:

$$\sum_{(i,j)\in\delta^-(S)} \min\{y_i, y_j\} \geq y_k, \quad \forall k \notin \{r\} \cup D^+(r),\ \forall S \subseteq V \setminus \{r\},\ k \in S. \tag{10.7}$$

Constraints (10.7) represent just a compact way of writing $2^{|\delta^-(S)|}$ inequalities;see also [Chen et al., 2013] where these constraints have been proposed for a problem arising in the design of telecommunication networks. They can be separated in polynomial time by solving a maximum-flow problem in an auxiliary support graph. Observe finally that indegree constraints (10.5) are also implied by these constraints: For each node $i \notin r \cup D^+(r)$, we have $y(D^-(i)) \geq \sum_{(j,i)\in\delta^-(i)} \min\{y_j, y_i\} \geq y_i$. We can now define the B-RMWCS as

$$(CUT_m) \qquad \max\left\{p^T y \mid \mathbf{y} \text{ satisfies (10.3),(10.4),(10.7) and } \mathbf{y} \in \{0,1\}^n\right\}.$$

Formulation Based on Node Separators. The other way of modeling the connectivity of a solution using only node variables is to consider *node separators*. This idea has been recently used in [Fügenschuh and Fügenschuh, 2008, Carvajal et al., 2013] and [Chen et al., 2013] to model connectivity in the context of sheet metal design, forest planning, and telecommunication network design, respectively. The following inequalities will be called *node-separator constraints*:

$$y(N) \geq y_k, \quad \forall k \notin \{r\} \cup D^+(r),\ N \in \mathcal{N}(r,k). \tag{10.8}$$

These constraints ensure that for each node k taken into the solution, either k is a direct neighbor of r, or there has to be a path from r to k such that for each node i on this path, $y_i = 1$. Notice that whenever $\mathcal{N}(k,\ell) \neq \emptyset$, $D^-(k) \in \mathcal{N}(k,\ell)$ and in this case the in-degree inequalities (10.5) are contained in (10.8). Thus, we can formulate the B-RMWCS as

$$(CUT_r) \qquad \max\left\{p^T y \mid \mathbf{y} \text{ satisfies (10.3),(10.4),(10.8), } \mathbf{y} \in \{0,1\}^n\right\}.$$

10.2.3 Some More Useful Constraints

In case that the budget constraint (10.4) is imposed, the following family of cover inequalities can be used to cut off infeasible solutions. **Cover Inequalities.** We say that a subset of nodes $V_C \subset V$ is a cover if the sum of node costs in V_C is greater

than the allowed budget B. In that case, at least one node from V_C has to be left out in any feasible solution. A cover V_C is minimal if $C \setminus \{i\}$ for any $i \in V_C$ is not a cover anymore. Let $\mathcal{V}_C$ be a family of all minimal covers with respect to B. Then, the following *cover inequalities* are valid for the B-RMWCS:

$$\sum_{i \in V_C} y_i \leq |V_C| - 1, \quad \forall V_C \in \mathcal{V}_C \tag{10.9}$$

For further details on cover inequalities, [see, e.g., Kaparis and Letchford, 2010].

10.3. Polyhedral Results

In this section we compare the proposed MIP formulations with respect to their quality of LP bounds and we show that, under certain conditions, the newly introduced node-separator inequalities are facets of the rooted connected subgraph polytope.

10.3.1 Theoretical Comparison of MIP Models

Let $\mathcal{P}_{\text{LP}}(.)$ denote the polytope of the LP-relaxations of the MIP models presented above and $v_{LP}(.)$ their optimal LP-values. We can show that:

Proposition 1. We have $\mathcal{P}_{\text{LP}}(CUT_r) \subsetneq \mathcal{P}_{\text{LP}}(CUT_m) \subsetneq \mathcal{P}_{\text{LP}}(CYCLE_r)$, and there exist instances for which the strict inequality holds.

Proof. $\mathcal{P}_{\text{LP}}(CUT_m) \subsetneq \mathcal{P}_{\text{LP}}(CYCLE_r)$: Consider a feasible solution $\hat{\mathbf{y}}$ of the LP relaxation of model CUT_m. We will show that each such solution is feasible for the model $CYCLE_r$. Let C be an arbitrary cycle from $\mathcal{C}$. Then, obviously, for any node $k \in C$, we have $\hat{y}_i(D^-(C)) \geq \sum_{(i,j)\in\delta^-(C)} \min\{\hat{y}_i, \hat{y}_j\} \geq \hat{y}_k$. Adding up this inequality with inequalities $1 \geq \hat{y}_i$, for each $i \in C \setminus \{k\}$, we obtain: $\hat{y}(D^-(C)) + |C| - 1 \geq \hat{y}(C)$ which is exactly the subtour elimination inequality associated to C. To see that the strict inequality holds, consider the directed graph shown in Figure 10.1(a).
$\mathcal{P}_{\text{LP}}(CUT_r) \subsetneq \mathcal{P}_{\text{LP}}(CUT_m)$: Consider a feasible solution $\hat{\mathbf{y}}$ of the LP relaxation of the CUT_r model. Let $k \in V \setminus (\{r\} \cup D^+(r))$ be an arbitrary node such that $\hat{y}_k > 0$ and let $S \subset V \setminus \{r\}$ be a set such that $k \in S$. Then, we will show that $\sum_{(ij)\in\delta^-(S)} \{\hat{y}_i, \hat{y}_j\} \geq \hat{y}_k$, i.e., $\hat{\mathbf{y}}$ satisfies (10.7). Let $N_1 = \{i \mid (i,j) \in \delta^-(S)\}$. Observe that $r \notin N_1$ and by definition, N_1 is a node separator for k, i.e., $N_1 \in \mathcal{N}(r,k)$. Let $N_2 = \{j \mid (i,j) \in \delta^-(S)\}$: (i) If $k \notin N_2$, then N_2 is a node separator for k ($N_2 \in \mathcal{N}(r,k)$). Consider the bipartite graph defined by $\delta^-(S)$. Each possible vertex cover $N' \subset N_1 \cup N_2$ on this graph, induces a node separator for k, i.e., $N' \in \mathcal{N}(r,k)$. There are $2^{|\delta^-(S)|}$ vertex covers in total, and constraints (10.8) associated to them imply constraint (10.7); (ii) if $k \in N_2$, then all vertex covers involving k trivially satisfy $\hat{y}(N') \geq \hat{y}_k$ for $k \in N'$. Together

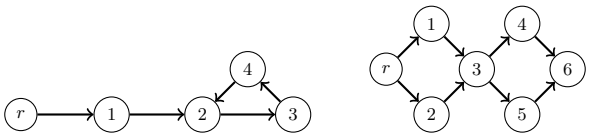


FIGURE 10.1: Examples that prove the strength of the new formulations. (a) The LP-solution of $CYCLE_r$ sets $y_2 = y_3 = y_4 = 2/3$ and $y_1 = 0$, and this solution is infeasible for the model CUT_m. (b) The LP-solution of CUT_m satisfies $y_1 = \cdots = y_5 = 1/2$ and $y_6 = 1$. This solution is infeasible for CUT_r.

with the remaining vertex covers, inequality (10.7) is implied. An example shown in Figure 10.1(b) shows an instance for which the strict inequality holds.

□

Proposition 2. The (SA_r) model and the (CUT_r) model are equally strong, i.e., $v_{LP}(SA_r) = v_{LP}(CUT_r)$.

Proof. We first show that $v_{LP}(SA_r) \geq v_{LP}(CUT_r)$: Let $(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ be a feasible solution for the relaxation of the SA_r model. Let $k \in V \backslash \{r\}$ be a node such that $\hat{y}_k > 0$ and let $N \in \mathcal{N}(r, k)$. Because of in-degree constraints of the SA_r model, we have that $\sum_{i \in N} \hat{y}_i = \sum_{i \in N} \hat{\mathbf{z}}(\delta^-(i))$. If N is removed from G, k cannot be reached from r. Let $S_r \subseteq V$, $r \in S_r$, be all the nodes i that can be reached from r after removing N, and let $S_k = V \setminus (N \cup S_r)$, $k \in S_k$. Because of inequalities (10.2), it holds that $\hat{\mathbf{z}}(\delta^+(S_r)) \geq \hat{y}_k$. Moreover, observe that for each $(i, j) \in \delta^+(S_r)$ we have that $i \in S_r$ and $j \in N$, which means that $\sum_{i \in N} \hat{\mathbf{z}}(\delta^-(i)) \geq \hat{\mathbf{z}}(\delta^+(S_r))$. Therefore, $\sum_{i \in N} \hat{y}_i \geq \hat{y}_k$, which proves that any LP solution of the SA_r model can be projected into a feasible solution of the CUT_r with the same objective value.

To show that $v_{LP}(CUT_r) \geq v_{LP}(SA_r)$ consider a solution $\check{\mathbf{y}} \in \mathcal{P}_{\text{LP}}(CUT_r)$. We will construct a solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \in \mathcal{P}_{\text{LP}}(SA_r)$ such that $\check{\mathbf{y}} = \hat{\mathbf{y}}$. On the graph G' (see Section 10.4.1, separation of separator inequalities) with arc capacities of (i_1, i_2) set to $\check{y}_i$ for each $i \in V \setminus \{r\}$ and to 1 otherwise, we are able to send $\check{y}_k$ units of flow from the root r to every (k_1, k_2) such that $\check{y}_k > 0$. Let f_{ij}^k denote the amount of flow of commodity k, sent along an arc $(i, j) \in A'$. Let $\mathbf{f}$ be the minimal feasible multi-commodity flow on G' (i.e., the effective capacities on G' used to route the flow cannot be reduced without violating the feasibility of this flow). We now define the values of $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ as follows:

$$\hat{z}_{ij} = \begin{cases} \max_{k \in V \setminus \{r\}} f^k_{i_2 j_1}, & i, j \in V \setminus \{r\} \\ \max_{k \in V \setminus \{r\}} f^k_{i, j_1}, & i = r, j \in V \setminus \{r\} \end{cases}, \forall (i, j) \in A, \quad \text{and}$$

let $\hat{y}_i = \hat{\mathbf{z}}(\delta^-(i))$, for all $i \in V \setminus \{r\}$. Obviously, the constructed solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ is feasible for the (SA_r) model, and, due to the assumption that $\mathbf{f}$ is minimal feasible, it follows that $\check{\mathbf{y}} = \hat{\mathbf{y}}$, which concludes the proof.

□

Finally, regarding the strength of the three MIP models studied by [Dilkina and Gomes, 2010], we notice that their single-commodity flow model is weaker than the multi-commodity model, which is equally strong as the cut-set model (SA_r) [see, e.g., Ljubić, 2004].

10.3.2 Facets of the RCS Polytope

In this section we consider the RMWCS with $R = \emptyset$, and let $\mathcal{P}$ denote the *rooted connected subgraph (RCS) polytope* defined in the natural space of $\mathbf{y}$ variables:

$$\mathcal{P} = \text{conv}\{\mathbf{y} \in \{0,1\}^n \mid \mathbf{y} \text{ satisfies (10.8)}\}.$$

In this section we establish under which conditions some of the presented inequalities are facets of the RCS polytope.

Lemma 1. The RCS polytope is full-dimensional (i.e., $\dim(\mathcal{P}) = n$) if and only if there exists a directed path between r and any $i \in V$.

Proof. We first generate a spanning arborescence T in G rooted at r. We will then apply a *tree pruning technique* in order to generate $n+1$ affine independent feasible RMWCS solutions. We start with the arborescence T in which case $\mathbf{y}$ consists of all ones. We iteratively remove one by one leaf from T, until we end up with a single root node (in which case $\mathbf{y}$ is a zero vector). Thereby, we generate a set of $n+1$ affinely independent solutions. Conversely, if $\mathcal{P}$ is full dimensional, then in order to create a feasible solution containing an arbitrary node $i \in V$, there has to be a directed path between r and i in G.

□

Lemma 2. Inequality $y_i \geq 0$ for $i \in V$ is facet defining if and only if in the graph $G - i$, any node $j \in V \setminus \{i\}$ can be reached from r.

Proof. Assume that in $G - i$ every node can be reached from r. Then, in $G - i$ we can build an arborescence T spanning all the nodes from $V \setminus \{i\}$. By applying the tree pruning technique from above, we generate n affinely independent RCS solutions such that $y_i = 0$. Conversely, assume that $y_i \geq 0$ is a facet for some $i \in V$ for which there exist some $j \neq i$ such that j cannot be reached from r in $G - i$. Therefore, by fixing $y_i = 0$ we also have $y_j = 0$, which is a contradiction.

□

Lemma 3. Inequality $y_i \leq 1$ for $i \in V$ is facet defining if and only if every node in V can be reached from r and there either exists $(r, i) \in A$, or there exist two node disjoint paths between r and i in G.

Proof. Assume that every node from V can be reached from r and $(r, i) \in A$. We build a spanning arborescence T using this arc, and apply the tree pruning until we end up with this single arc. Thereby, we generate n affinely independent solutions. Alternatively, assume that $(r, i) \notin A$ but there are two node-disjoint paths P and P' between r and i. Let $\ell_P \geq 1$ be the number of internal nodes on the path P. We build a family $\mathcal{T}$ of n affinely independent solutions as follows: Let T be a spanning arborescence T such that P is fully contained in T. We apply the tree pruning until we end up with the path P and insert all those solutions in $\mathcal{T}$. Thereby, we will generate $n - \ell_P$ affinely independent solutions. We now consider another spanning subtree T' such that P' is fully contained in T'. We apply pruning on T' until only P' remains. Subtrees T_i obtained by the pruning procedure in which $i \in P$ was a leaf which has been just deleted, are inserted in $\mathcal{T}$. That way, we create additional ℓ_P solutions such that $y_i = 1$, that are affinely independent with the previously generated ones.

To prove that the conditions are sufficient, assume that $y_i = 1$ is a facet of $\mathcal{P}$ and there is a single path P (with $\ell_P \geq 1$) connecting r with i in G or the paths are not node-disjoint, i.e., there exist $j \neq i$ such that every r-i path passes through j. Then, obviously, each RMWCS feasible solutions such that $y_i = 1$ also satisfies $y_j = 1$, which is a contradiction.

□

Given some $k \in V$ and $N \in \mathcal{N}(r, k)$, let us now consider the corresponding node separator inequalities: $y(N) \geq y_k$. Let $S_r \subset V$ denote the subset of nodes that can be reached from r in $G - N$, and let S_k be the remaining nodes, i.e., $S_k = V \setminus (N \cup S_r)$. Then, we have:

Proposition 3. Given some $k \in V$ and $N \in \mathcal{N}(r, k)$, the associated node separator inequality $y(N) \geq y_k$ is facet defining if N is minimal, every node in V can be reached from r and every node in S_k can be reached from k.

Proof. For a given $k \in V$ and $N \in \mathcal{N}(r, k)$, that satisfy the above properties we prove the statement using the indirect method. Let $F(k, N) = \{y \in \{0, 1\}^n \mid \sum_{i \in N} y_i = y_k\}$. Consider a facet defining inequality of the form $\mathbf{a}^t\mathbf{y} \geq a_0$. We will show that if all points in $F(k, N)$ satisfy

$$\mathbf{a}^t\mathbf{y} = a_0 \tag{10.10}$$

then $\mathbf{a}^t\mathbf{y} \geq a_0$ is a positive multiple of (10.8). Observe first that the zero vector belongs to $F(k, N)$. By plugging it into (10.10), we get $a_0 = 0$. Consider now an arbitrary node $\ell \in S_r$. Consider a path P from r to ℓ in S_r, and its subpath Q obtained by deleting ℓ. Characteristic vectors of both of them belong to $F(k, N)$, and by subtracting them, we obtain $a_\ell = 0$, for all $\ell \in S_r$. Consider now an arbitrary $\ell \in S_k$. Let P be a path from r to ℓ that passes through exactly one node $i \in N$ and through k. We can find such a path for the following reasons: (i) A path from r to k over a single node $i \in N$ exists because N is minimal. (ii) A path from k to ℓ fully contained in S_k also exists by our assumption. Let Q be a subpath of P obtained by deleting ℓ. Characteristic vectors of P and Q belong to $F(k, N)$, and by subtracting them, we obtain $a_\ell = 0$, for all $\ell \in S_k$. Finally, consider an arbitrary $i \in N$ and a path P' from r to k passing through i and no other nodes from N. Characteristic vector of P' belongs to $F(k, n)$ and after plugging it into (10.10), we obtain $a_i + a_k = 0$, for all $i \in N$. Therefore, we have $a_i = -a_k = \alpha$, and (10.10) can be written as $\alpha(y(N) - y_k) = 0$, which concludes the proof.

□

10.4. Computational Results

In this section, we study the computational performance of Branch-and-Cut (B&C) algorithms for the models (SA_r) and (CUT_r) for both the RMWCS and the B-RMWCS.

10.4.1 Branch-and-Cut Algorithms

Constraint Separation At each node of the search tree, constraints (10.2) of the (SA_r) formulation are separated by solving a max-flow problem [see for further details Ljubić et al., 2006]. For the (CUT_r) model, inequalities (10.8) can be separated in polynomial time on an auxiliary support graph G' that splits all nodes except the root into arcs so that each $i \in V$ is replaced by an arc (i_1, i_2). All ingoing arcs into i are now connected to i_1, and all outgoing arcs from i are now connected from i_2. For a given node fractional solution $\tilde{y}$ and $k \in V \setminus (\{r\} \cup D^+(r))$ such that $\tilde{y}_k > 0$, to check whether there are violated inequalities of type (10.8) we calculate the maximum flow between r and (k_1, k_2) in G' whose arc capacities are defined as $\tilde{y}_i$ for splitted arcs and to zero, otherwise. For both cases, we also use *nested*, *back-flow* and *minimum cardinality* cuts in order to insert as many violated cuts as possible [see Koch and Martin, 1998, Ljubić et al., 2006]. At each separation callback, we limit the number of inserted cuts to 25.

For the B-RMWCS, the cover inequalities (10.9) are separated by solving a knapsack problem (which is weakly NP-hard) for each fractional solution $\tilde{y}$:

$$(P_{CI}) \qquad \min\{\sum_{i\in V}(1-\tilde{y}_i)a_i \mid \sum_{i\in V} c_i a_i > B, a_i \in \{0,1\}^n\};$$

if the optimal value of (P_{CI}) is less than one, the nodes $i \in V$ such that $a_i = 1$ are the nodes of a cover V_C for which the corresponding inequality (10.9) is violated. Finally, once the violated cover inequality is detected, we insert the following *extended cover inequality* in the MIP:

$$\sum_{i\in V_C\cup V^*(C)} y_i \leq |V_C| - 1, \quad \forall V_C \in \mathcal{V}_C \tag{10.11}$$

where $V^*(C) = \{i \in V \setminus V_C \mid c_i \geq \max_{j\in V_C} c_j\}$. We solve the knapsack problem P_{CI} within the B&C using CPLEX. Only at the root node of the branch-and-bound tree the problem P_{CI} is solved to optimality; in the remaining nodes it is solved until reaching a 0.01% gap. **Primal Heuristic.** At a given node of the branch-and-bound tree, we use the information of the current LP solution $\tilde{\mathbf{y}}$ in order to construct feasible primal solutions for the (B-)RMWCS. The procedure, which is equivalent for both (SA_r) and (CUT_r), consists of a (restricted) breadth-first search (BFS) that starts from the root node r and constructs a connected component. A node is incorporated into this component if its weight $\tilde{p}_v := p_v\tilde{y}_v$ is non-negative and its cost c_v added to the cost of the current component does not violate the budget B. **MIP Initialization.** As described in §10.4.2, part of our benchmark set consists of 4-grid graphs. In this case, all 4-cycles are easily enumerated by embedding the grid into the plane and iterating over all faces except for the outer face. Let $\mathcal{C}_4$ be the set of all 4-cycles C such that $r \notin C \cup D^-(C)$ and let $A[C]$ be the set of arcs associated to it. Therefore, in case of 4-grids, the (SA_r) model is initialized with the following *4-cycle inequalities*:

$$z(A[C]) \leq y(C \setminus i), \quad \forall i \in C,\ \forall C \in \mathcal{C}_4. \tag{10.12}$$

The corresponding 4-cycle inequalities for the (CUT_r) model are:

$$y(D^-(C)) \geq y_i, \quad \forall i \in C,\ \forall C \in \mathcal{C}_4. \tag{10.13}$$

Additionally, indegree constraints (10.1) (or (10.5)) and $z_{ij} + z_{ji} \leq y_i\ \forall e : \{i_{\neq r}, j\} \in E$ are added to the MIP. **Implementation.** The B&C algorithms were implemented using CPLEX$^{\text{TM}}$12.3 and Concert Technology. All CPLEX parameters were set to their default values, except that: (i) CPLEX cuts, CPLEX heuristics, and CPLEX preprocessing were turned off, and (ii) higher branching priorities were given to $\mathbf{y}$ variables in the case of the (SA_r) model. All the experiments were performed on a Intel Core2 Quad 2.33 GHz machine with 3.25 GB RAM, where each run was performed on a single processor. We denote as "Basic" the B&C implementation for which neither

the separation of CI nor the addition of 4-cycle inequalities, (10.12) or (10.13), is considered.

10.4.2 Benchmark Instances

Wildlife Corridor Design Instances. We have considered three real instances provided in [Dilkina and Gomes, 2010] that are instances of the corridor design problem for grizzly bears in the Rocky mountains, labeled as `CD-40×40-sq` (242 nodes, 469 edges), `CD-10×10-sq` (3299 nodes, 6509 edges) and `CD-25-hex` (12889 nodes, 38065 edges). In all of them, three reserves are given and the root is chosen as one of them.

We have also considered 4-grid instances generated using the generator of [Dilkina and Gomes, 2010]. The description of the parameters used for setting up the instances and the generator itself are available online at [Dilkina and Gomes, 2012]. These instances are labeled as `CD-O-C-T` [see for further details Dilkina and Gomes, 2012]. In our experiments we have generated instances with $n + 1 = \texttt{O}^2$, where $\texttt{O} \in \{10, 15, 20\}$. We also generated both, correlated and uncorrelated instances ($\texttt{C} = \{\texttt{U},\texttt{W}\}$). Weights and costs are independently and uniformly taken from $\{1, \ldots, 10\}$. We also considered $\texttt{T} = \{\texttt{2fR},\texttt{R}\}$ and, in addition to the root, we consider two more terminals. For each combination of these parameters we have generated 20 instances.

These instances were used for both the RMWCS and the B-RMWCS. For the B-RMWCS, for a given instance `I` with set of terminals R, let $\hat{C}_{min}$ be the cost of the minimum Steiner Tree on R with arc costs $\hat{c}_{ij} = c_j$. Values of the available budget B are defined using slacks over $\hat{C}_{min}$ [see also Dilkina and Gomes, 2010]. For example, a 10% of budget slack corresponds to $B = 1.10 \times \hat{C}_{min}$. For the RMWCS, we redefine weights as $w'_v = p_v - c_v$, which can be done because p_v and c_v have comparable units. That way, w'_v somehow represents the net-profit of including node v into the solution. For the RMWCS we set $R = \emptyset$ and we take as root node the reserve node with the smallest index.

Network Design Instances. These Euclidean instances with a topology similar to street networks are generated as proposed in [Johnson et al., 2000]: First, n nodes are randomly located in a unit Euclidean square. A link between two nodes i and j is established if the Euclidean distance d_{ij} between them is no more than $\alpha/\sqrt{n}$, for a fixed $\alpha > 0$. For a given n and a given α, weights and costs are independently and uniformly taken from $\{1, \ldots, 10\}$.

We generated instances using $n = \{500, 750, 1000\}$ and $\alpha = \{0.6, 1.0\}$; in case that for a given distribution of n nodes in the plane the value of α is not enough for defining a connected graph, it is increased by 0.01 until connecting all components. For each combination of n and α, 20 instances are generated. We take as root the node with

	SA_r					CUT_r				
Instance	T_{av}(s)	T_{med}(s)	Gap	#(10.2)	#NOpt	T_{av}(s)	T_{med}(s)	Gap	#(10.8)	#NOpt
CD-40×40-sq	5.28	4.45	0.00	388	0	4.28	3.27	0.00	90	0
CD-10×10-sq	619.58	332.40	0.07	1262	10	1389.07	1441.68	1.39	871	14
CD-25-hex	–	–	5.17	11524	18	–	–	4.81	2958	18
CD-10-U-2fR	1.67	1.12	–	527	0	2.71	1.82	–	360	0
CD-10-W-2fR	1.80	1.00	–	535	0	2.22	1.50	–	389	0
CD-10-U-3R	0.91	0.71	–	362	0	0.63	0.38	–	157	0
CD-10-W-3R	3.08	0.50	–	389	0	0.82	0.42	–	190	0
CD-15-U-2fR	12.47	7.71	–	1085	0	26.33	13.78	–	883	0
CD-15-W-2fR	12.40	8.08	–	1222	0	26.61	10.98	–	1071	0
CD-15-U-3R	4.56	2.98	–	814	0	7.84	2.81	–	513	0
CD-15-W-3R	4.86	2.88	–	809	0	7.34	3.24	–	539	0

TABLE 10.1: Computational performance on B-RMWCS (+C4+CI) instances from [Dilkina and Gomes, 2010].

index 0 and when considering a set of terminals, these correponds to those nodes with labels 1 and 2.

10.4.3 Analyzing the Computational Performance

Results for the B-RMWCS. Table 10.1 shows a comparison of (SA_r) and (CUT_r) models (including 4-cycle and CI) on the set of corridor design instances. The first three rows correspond to the real instances provided by [Dilkina and Gomes, 2010], so for each of them we report statistics over a set of 18 problems (obtained for different budget slacks taken from $\{10, 15, \ldots, 95\}$). For the remaining rows, since we create 20 instances for each parameter setting, the reported values correspond to statistics over $18 \times 20 = 360$ instances. In columns T_{av}(s) and T_{med}(s) we report the average and median running times (in seconds), respectively, of those instances solved to optimality, in columns Gap we show the gaps (as percentages) of those instances that were not solved to optimality within 1800 seconds. Columns #(10.2) and #(10.8) show the number of connectivity cuts of the (SA_r) and (CUT_r) model, respectively. Column #NOpt shows the number of instances that are not solved to optimality within 1800 seconds. We observe that for all 4-grid instances, except for the CD-10×10-sq graph for which a more detailed analysis is given below, both approaches are able to solve all instances in more or less reasonable times, although the (SA_r) model is slightly better than the (CUT_r) model. On the other hand, the number of inserted violated cuts of the (CUT_r) model is in all of the cases significantly smaller than the corresponding number for the (SA_r) model. The efficacy of the (SA_r) model can be explained by the sparsity of 4-grid graphs. On the contrary, for the only more dense instance of this group, namely CD-25-hex, which is a 6-grid with 12889 nodes and 38065 edges, the (CUT_r) model performs better than the (SA_r) model. More precisely, the avg. gap and its standard deviation for the (SA_r) model are 5.17% and 1.11%, resp., while for the (CUT_r) model these values are 4.81% and 0.81%, resp.'

To analyze the effects of special inequalities, namely 4-cycle and CI, we compare three approaches: Basic, Basic plus 4-cycle inequalities (denoted by "+C4") and Basic plus 4-cycle and CI (denoted by "+C4+CI"). In Figure 10.2 we present the box-plots of the gaps attained within 1800 seconds when solving real instance `CD-10×10-sq` for budget slacks taken from $\{10, 15, \ldots, 95\}$. The values marked with an asterisk and $\times$ correspond to the mean and maximum running time, respectively. Below the bottom of each box the number of instances solved to optimality is indicated, and next to "#Cuts:" we report the average number of detected cuts of type (10.2) and (10.8), respectively.

The box-plots indicate that for the Basic setting the (CUT_r) model significantly outperforms the (SA_r) model on this instance, in terms of the quality of the solutions (smaller gaps), the stability of the approach (smaller dispersion), and the number of instances solved to optimality. This is mainly due to the fact that in the (CUT_r) model there are less variables, so the optimization becomes easier and more stable. However, when including 4-cycle inequalities, although both approaches perform better, (SA_r) now outperforms (CUT_r). The average number of inserted cuts of type (10.2) decreases from 5989 to 1264 when 4-cycle inequalities are added, while for the (CUT_r) model this reduction is more attenuated (only 18%). This means that for this instance constraints (10.12) are empirically more effective than (10.13) in reducing too frequent calls of the maximum flow procedure. When adding the separation of CI ("+CI") we observe that these constraints are more beneficial for the (SA_r) model than for the (CUT_r) model - the latter one even slows down with addition of these cuts. This can be explained by some numerical instability that can appear when dealing with the separation of CI. We conclude that the advantage of the (CUT_r) model of having less variables vanishes when more sophisticated ideas are considered.

For the Network Design instances (whose complete results are not reported due to space limitation), the graph density plays a role in the performance of the two models. For instance, for $n \in \{500, 750\}$ and $\alpha = 0.6$, the (SA_r) model solves 536 instances out of 760 within the time limit, while the (CUT_r) model solves 443. However, when $\alpha = 1.0$, the (SA_r) approach solves 483 while the (CUT_r) approach solves 502. In both cases, the average running times of the (CUT_r) model needed to prove optimality are smaller than those of the (SA_r) model.

Results for the RMWCS. For the RMWCS we have considered the same corridor design instances and, in addition, the network design instances with a weight transformation as described in § 10.4.2. In Table 10.2, equivalent to Table 10.1, we report the results obtained for the corridor design instances. In this case, time limit is set to 3600 seconds. We observe that the (CUT_r) model outperforms the (SA_r) model on real instances, and on random lattices it is the other way around, although the differences are less visible.

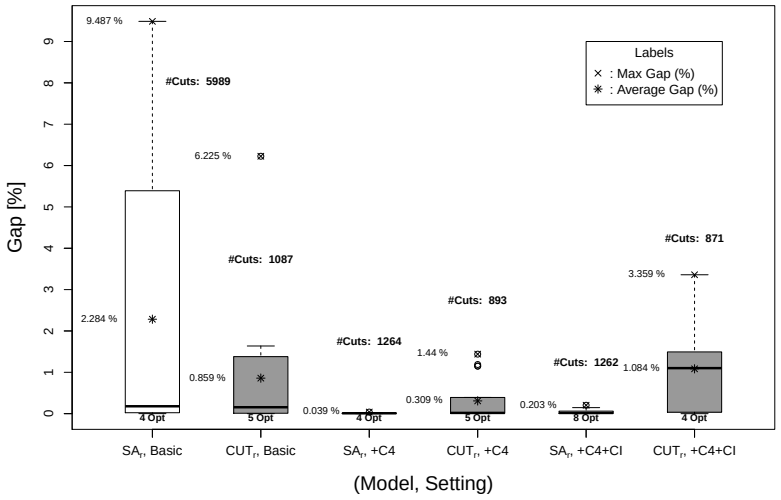


FIGURE 10.2: Box-plots of the gaps [%] reached within 1800 sec for the `CD-10×10-sq` instance considering (SA_r) and (CUT_r) and three different settings of the B&C (Budget slack [%] taken from $\{10, 15, \ldots, 95\}$).

	SA_r				CUT_r			
Instance	Time(sec)	Gap(%)	#(10.2)	#NOpt	Time(sec)	Gap(%)	#(10.8)	#NOpt
`CD-40×40-sq`	0.70	–	254	0	0.16	–	10	0
`CD-10×10-sq`	316.11	–	3998	0	88.70	–	60	0
`CD-25-hex`	3600.00	1.99	20304	1	2611.13	–	14756	0
`CD-10-U-2fR`	0.15	–	231	0	0.14	–	34	0
`CD-10-W-2fR`	0.14	–	239	0	0.18	–	40	0
`CD-10-U-3R`	0.13	–	226	0	0.13	–	28	0
`CD-10-W-3R`	0.15	–	241	0	0.12	–	26	0
`CD-15-U-2fR`	1.28	–	720	0	11.59	–	99	0
`CD-15-W-2fR`	1.35	–	755	0	3.66	–	94	0
`CD-15-U-3R`	1.24	–	763	0	2.02	–	73	0
`CD-15-W-3R`	1.45	–	809	0	2.26	–	78	0
`CD-20-U-2fR`	7.67	–	1618	0	166.32	–	223	0
`CD-20-W-2fR`	7.41	–	1615	0	74.46	–	234	0
`CD-20-U-3R`	7.57	–	1667	0	16.90	–	133	0
`CD-20-W-3R`	8.39	–	1765	0	86.18	–	195	0

TABLE 10.2: Computational performance on instances from [Dilkina and Gomes, 2010] when solving the RMWCS.

The results on the network design instances are reported in Table 10.3. For a given n and α equal to 0.6 and 1.0, respectively, column #nodes shows $n+1$ and column #edges shows the average number of edges for a set of 20 instances created using this setting. All instances of this group were solved to optimality, therefore in Table 10.3 we only report the average running times and the average number of detected connectivity cuts. For these instances, the (CUT_r) approach clearly outperforms the (SA_r) approach; for these instances, the ratio between the number of edges and the number of nodes is, depending on the value of α, around 5 or 13, in contrast to the corridor design instances, where this ratio is close to two. This characteristic implies a practical difficulty for the

		SA_r		CUT_r	
#nodes	#edges	Time(sec)	#(10.2)	Time(sec)	#(10.8)
500	2535	11.42	1218	2.29	22.8
500	6484	3.50	211	0.84	<10
750	3845	57.07	2541	5.67	25.8
750	9944	7.69	287	1.71	<10
1000	5180	97.41	3188	15.59	36.3
1000	13397	10.16	302	2.77	<10

TABLE 10.3: Computational performance on the RMWCS network design instances.

(SA_r) model due to the increase of the number of variables. Besides, for this group of instances, 4-cycle constraints and CI cannot be used in the initialization.

10.4.4 Conclusion.

The obtained computational results let us conclude that both models (CUT_r) and (SA_r) perform very well in practice, and that their performance is complementary. Using the (CUT_r) model (i.e., having less variables) pays off for denser graphs with many zero-weight nodes for both, B-RMWCS and RMWCS.

Chapter 11

Final Remarks

In this thesis we have addressed a collection of Network Design problems which are strongly motivated by applications from Telecommunications, Logistics and Bioinformatics. In most cases we have justified the need of taking into account uncertainty in some of the problem parameters, and different Robust Optimization models have been used to hedge against it. Mixed integer linear programming formulations along with sophisticated algorithmic frameworks have been designed, implemented and rigorously assessed for the majority of the studied problems. The obtained results let us draw the following general conclusions: (i) relevant real problems can be effectively represented as (discrete) optimization problems within the framework of network design; (ii) uncertainty can be appropriately incorporated into the decision process if a suitable robust optimization model is considered; (iii) optimal, or nearly optimal, solutions can be obtained for large instances if a tailored algorithm, that exploits the structure of the problem, is designed; (iv) a systematic and rigorous experimental analysis allows to understand both, the characteristics of the obtained (robust) solutions and the behavior of the proposed algorithm.

Most of the models and algorithmic tools developed in this thesis can complement each other and can be extended to other related network design problems. For instance, the considered applications in Bioinformatics (see Chapters 9) suggest that the Minmax Regret criterion is appropriate for incorporating uncertainty in this context. Therefore, we can use the models and the algorithms proposed for the Maximum Weight Connected Subgraph problem and combine them with the algorithmic framework designed for the Minmax Regret Spanning Tree (see Chapter 6). Likewise, one might be interested in studying the Connected Facility Location problem under uncertainty. Hence, the approach developed in Chapter 4 for the Robust Uncapacitated Facility Location problem can be extended to this variant by borrowing some additional modeling and algorithmic techniques from the methodology presented in Chapter 3 for the Robust Two-Level Network Design.

Overall, this thesis intends to contribute, mainly from a methodological point of view, to the fields of Network Design and Robust Optimization.

Bibliography

D. Adjiashvili. *Structural Robustness in Combinatorial Optimization*. PhD thesis, ETH ZURICH, 2012.

R. K. Ahuja, T. L. Magnanti, and J.B. Orlin, editors. *Network Flows*. Prentice-Hall, 1st edition, 1993.

H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of some combinatorial optimization problems: a survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

M. Albareda-Sambola, E. Fernández, and F. Saldanha-da Gama. The Facility Location problem with Bernoulli demands. *Omega*, 39(3):335–345, 2011.

M. Albareda-Sambola, A. Alonso-Ayuso, L. Escudero, E. Fernández, and C. Pizarro. Fix-and-relax-coordination for a multi-period location-allocation problem under uncertainty. *Computers & OR*, 40(12):2878–2892, 2013.

A. Altin, E. Amaldi, P. Belotti, and M. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1):100–115, 2007.

S. Alumur, S. Nickel, and F. Saldanha-da Gama. Hub location under uncertainty. *Transportation Research Part B: Methodological*, 46(4):529–543, 2012.

E. Álvarez-Miranda, A. Candia, X. Chen, B. Li, and X. Hu. Efficient algorithms for the PCStT with interval data. In B. Chen, editor, *Proceedings of AAIM 2010*, volume 6124 of *LNCS*, pages 13–24. Springer, 2010.

E. Álvarez-Miranda, I. Ljubić, and P. Toth. Exact solutions for the robust prize-collecting steiner tree problem. In *Proceedings of the IEEE International Congress on Modern Telecommunications and Control Systems 2011*, pages 1–7, 2011.

E. Álvarez-Miranda, V. Cacchiani, T. Dorneth, M. Jünger, F. Liers, A. Lodi, T. Parriani, and D. Schmidt. Models and algorithms for robust network design with several traffic scenarios. In A. Mahjoub, V. Markakis, I. Milis, and V. Paschos, editors, *Proceedings of ISCO 2012*, volume 7422 of *LNCS*, pages 261–272. Springer, 2012.

E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The maximum weight connected subgraph problem. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization*, pages 245–270. Springer, 2013a.

E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The rooted maximum node-weight connected subgraph problem. In C. Gomes and M. Sellmann, editors, *Proceedings of CPAIOR 2013*, volume 7874 of *LNCS*, pages 300–315. Springer, 2013b.

E. Álvarez-Miranda, I. Ljubić, S. Raghavan, and P. Toth. The recoverable robust two-level network design problem. *Submitted to INFORMS Journal on Computing*, 2013c.

E. Álvarez-Miranda, I. Ljubić, and P. Toth. A note on the Bertsimas & Sim algorithm for robust combinatorial optimization problems. *4OR*, 11(4):349–360, 2013d.

E. Álvarez-Miranda, I. Ljubić, and P. Toth. Exact approaches for solving robust prize-collecting steiner tree problems. *European Journal of Operational Research*, 229(3): 599–612, 2013e.

G. Andreello, A. Caprara, and M. Fischetti. Embedding $\{0, 1/2\}$-cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19: 229–238, 2007.

A. Archer, M. Bateni, M. Hajiaghayi, and H. Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. *SIAM Journal on Computing*, 40 (2):309–332, 2011.

I. Aron and P. Van Hentenryck. A constraint satisfaction approach to the robust spanning tree problem with interval data. In A. Darwiche and N. Friedman, editors, *Proceedings of UAI 2002*, pages 18–25. Morgan Kaufmann, 2002.

I. Aron and P. Van Hentenryck. On the complexity of the robust spanning tree problem with interval data. *Operations Research Letters*, 32(1):36–40, 2004.

N. Assimakopoulos. A network interdiction model for hospital infection control. *Computers in Biology and Medicine*, 17(6):413 – 422, 1987.

A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, Series B(92):425–437, 2000.

A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

I. Averbakh. The minmax relative regret median problem on networks. *INFORMS Journal on Computing*, 17(4):451–461, 2005.

I. Averbakh and V. Lebedev. Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138(3):289–301, 2004.

C. Backes, A. Rurainski, G. Klau, O. Müller, D. Stöckel, A. Gerasch, J. Küntzer, D. Maisel, N. Ludwig, M. Hein, A. Keller, H. Burtscher, M. Kaufmann, E. Meese, and H. Lenhof. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, 1:1–13, 2011.

M. Bailly-Bechet, A. Braunstein, and R. Zecchina. A prize-collecting steiner tree approach for transduction network inference. In P. Degano and R. Gorrieri, editors, *Computational Methods in Systems Biology*, volume 5688 of *LNCS*, pages 83–95. Springer, 2009.

M. Bailly-Bechet, C. Borgs, A. Braunstein, J. Chayes, A. Dagkessamanskaia, J. Francois, and R. Zecchina. Finding undetected protein associations in cell signaling by belief propagation. *Proceedings of the National Academy of Sciences*, pages 1–6, 2010.

A. Balakrishnan, T. Magnanti, and P. Mirchandani. A dual-based algorithm for multi-level network design. *Management Science*, 40(5):567–581, 1994a.

A. Balakrishnan, T. Magnanti, and P. Mirchandani. Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Science*, 40(7):846–867, 1994b.

E. Balas. The prize-collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286: 509–512, 1999.

M. Bateni, C. Chekuri, A. Ene, M. Hajiaghayi, N. Korula, and D. Marx. Prize-collecting Steiner problems on planar graphs. In D. Randall, editor, *Proceedings of the 22nd Symposium on Discrete Algorithms*, pages 1028–1049. ACM/SIAM, 2011.

J.E. Beasly. Collection of test data sets for a variety of operations research (OR) problems, 1990. URL http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and Engineering*, 6(3):283–313, 2005.

A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, Series B(88):411–421, 2000.

A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, Series A(99):351–376, 2004.

A. Ben-Tal, L. El-Ghaoui, and A. Nemirovski, editors. *Robust Optimization*. Series in Applied Mathematics. Princeton, 1st edition, 2010.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, Series B(98):49–71, 2003.

D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

D. Bienstock, M. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Progamming*, 59(1-3):413–420, 1993.

D. Bienstock, S. Chopra, O. Günlük, and C. Tsai. Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81(2):177–199, 1998.

J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Series in Operations Research and Financial Engineering. Springer, 2nd edition, 2011.

C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In J. Pahl, T. Reiners, and S. Voß, editors, *Proceedings of INOC 2011*, volume 6701 of *LNCS*, pages 7–17. Springer, 2011.

C. Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

C. Büsing, A. Koster, and M. Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optimization Letters*, 5(3):379–392, 2011.

V. Cacchiani, A. Caprara, L. Galli, L. Kroon, and G. Maróti. Recoverable robustness for railway rolling stock planning. In M. Fischetti and P. Widmayer, editors, *Proceedings of ATMOS 2008*, volume 9 of *OpenAccess Series in Informatics*. Schloss Dagstuhl, 2008.

A. Candia-Véjar, E. Álvarez-Miranda, and N. Maculan. Minmax regret combinatorial optimization: An algorithm perspective. *RAIRO-OR*, 45(2):101–129, 2011.

S. Canuto, M. Resende, and C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38(1):50–58, 2001.

A. Caprara and M. Fischetti. $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. *Mathematical Programming*, 74:221–235, 1996.

R. Carvajal, M. Constantino, M. Goycoolea, J.P. Vielma, and A. Weintraub. Imposing connectivity constraints in forest planning models. *Operations Research*, 61(4):824–836, 2013.

S. Chamberland. On the design problem of two-level IP networks. In *Proceedings of the IEEE 14th Symposium on International Telecommunications Network Strategy and Planning*, pages 1–6, 2010.

C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38:106–128, 2007.

C. Chen and K. Grauman. Efficient activity detection with max-subgraph search. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2012*, pages 1274–1281. IEEE, 2012.

S. Chen, I. Ljubić, and S. Raghavan. The generalized regenerator location problem, 2013.

X. Chen, J. Hu, and X. Hu. A new model for path planning with interval data. *Computers & OR*, 36(6):1893–1899, 2009.

B. Cherkassky and A. Goldberg. On implementing push-relabel method for the maximum flow problem. In E. Balas and J. Clausen, editors, *Proceedings of IPCO IV*, volume 920 of *LNCS*, pages 157–171. Springer, 1995.

R. Chien. Synthesis of a communication net. *IBM Journal of Research and Development*, 4(3):311–320, 1960.

M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Obtaining optimal k-cardinality trees fast. *ACM Journal of Experimental Algorithmics*, 14, 2009.

S. Chopra and C. Tsai. A branch-and-cut approach for minimum cost multi-level network design. *Discrete Mathematics*, 242(1-3):65–92, 2002.

R. Church, M. Scaparra, and R. Middleton. Identifying critical infrastructure: The median and covering facility interdiction problems. *Annals of the Association of American Geographers*, 94(3):491–502, 2004.

S. Cicerone, G. Di Stefano, M. Schachtebeck, and A. Schöbel. Multi-stage recovery robustness for optimization problems: A new concept for planning under disturbances. *Information Sciences*, 190:107–126, 2012.

J. Conrad, C. Gomes, J. van Hoeve, A. Sabharwal, and J. Suter. Wildlife corridors as a connected subgraph problem. *Journal of Environmental Economics and Management*, 63(1):1–18, 2012.

K. Cormican, D. Morton, and R. Wood. Stochastic network interdiction. *Operations Research*, 46(2):184–197, 1998.

G. Cornuejols, G. Nemhauser, and L. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*. Wiley-Interscience, 1990.

A. Costa, P. França, and C. Lyra. Two-level network design with intermediate facilities: An application to electrical distribution systems. *Omega*, 39(1):3–13, 2011.

T. Cui, Y. Ouyang, and Z. Shen. Reliable facility location design under the risk of disruptions. *Operations Research*, 58(4):998–1011, 2010.

J. Current. Design of a hierarchical transportation network with transshipment facilities. *Transportation Science*, 22(4):270–277, 1988.

J. Current, C. ReVelle, and J. Cohon. The hierarchical network design problem. *European Journal of Operational Research*, 27(1):57 – 66, 1986.

G. D'Angelo, G. Di Stefano, A. Navarra, and C. Pinotti. Recoverable robust timetables: An algorithmic approach on trees. *IEEE Transactions on Computers*, 60(3):433–446, 2011.

G. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3/4): 197–206, 1955.

M. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, 2nd edition, 2013.

DDM. Department of Disaster Management of Bangladesh, Ministry of Disaster Management and Relief, Government of the Republic Bangladesh, 2014. URL http://http://www.ddm.gov.bd/flood.php/.

B. Dilkina and C. Gomes. Solving connected subgraph problems in wildlife conservation. In A. Lodi, M. Milano, and P. Toth, editors, *Proceedings of CPAIOR 2010*, volume 6140 of *LNCS*, pages 102–116. Springer, 2010.

B. Dilkina and C. Gomes. Synthetic corridor problem generator, 2012. URL http://www.cs.cornell.edu/~bistra/connectedsubgraph.htm.

M. Dittrich, G. Klau, A. Rosenwald, T. Dandekar, and T. Muller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of SIGCOMM '99*, pages 95–108. ACM, 1999.

C. Duin and A. Volgenant. Reducing the hierarchical network design problem. *European Journal of Operational Research*, 39(3):332–344, 1989.

C. Duin and T. Volgenant. The multi-weighted steiner tree problem. *Annals of Operations Research*, 33(6):451–469, 1991.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22(4):425–460, 2000.

H. A. Eiselt and V. Marianov, editors. *Foundations of Location Analysis*, volume 155 of *International Series in Operations Research & Management Science*. Springer, 2011.

L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1–3):23–47, 2003.

M. Fischetti and M. Monaci. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation*, 4(3):239–273, 2012.

M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. Technical report, DEI, University of Padova, Italy, 2013.

M. Fischetti, K. Hamacher, H.and Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994.

M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of benders' cuts. *Mathematical Programming*, Series B(124):175–182, 2010.

L. Ford and D. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.

A. Fügenschuh and M. Fügenschuh. Integer linear programming models for topology optimization in sheet metal design. *Mathematical Methods of Operations Research*, 68(2):313–331, 2008.

D. Fulkerson and G. Harding. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming*, 13(1):116–118, 1977.

V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471 – 483, 2014.

Y. Gao. Uncertain models for single facility location problems on networks. *Applied Mathematical Modelling*, 36(6):2592 – 2599, 2012.

N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

GeneTrail, September 10th 2012. url = http://genetrail.bioinf.uni-sb.de/ilp/.

N. Gïlpinar, D. Pachamanova, and E. Canakoglu. Robust strategies for facility location under uncertainty. *European Journal of Operational Research*, 225(1):21–35, 2013.

M. Goemans and D. Williamson. The Primal-dual method for approximation algorithms and its application to network design problems. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 144–191. PWS Publishing Company, 1997.

A. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

B. Golden. A problem in network interdiction. *Naval Research Logistics Quarterly*, 25 (4):711–713, 1978.

S. Gollowitzer, L. Gouveia, and I. Ljubić. Enhanced formulations and branch-and-cut for the two level network design problem with transition facilities. *European Journal of Operational Research*, 225(2):211–222, 2013.

R. Gomory and T. Hu. Multi-terminal network flow. *SIAM Journal on Applied Mathematics*, 9:551–570, 1961.

R. Gomory and T. Hu. An application of generalized linear programming to network flows. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):260–283, 1962.

Google. Google Maps, 2013. URL http://maps.google.com/.

Google. Google Earth, 2014. URL http://www.google.com/intl/en/earth/.

L. Gouveia and J. Telhada. The multi-weighted steiner tree problem: A reformulation by intersection. *Computers & OR*, 35(11):3599–3611, 2008.

M. Grötschel, C. Monma, and M. Stoer. Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM Journal on Optimization*, 2(3):474–504, 1992.

S. Gupta and J. Rosenhead. Robustness in sequential investment decisions. *Management Science*, 15(2):B18–B29, 1968.

M. Haouari and J. Siala. A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Computers & OR*, 33(5):1274–1288, 2006.

M. Haouari, S. Layeb, and H. Sherali. The prize collecting Steiner tree problem: models and lagrangian dual optimization approaches. *Computational Optimization and Applications*, 40(1):13–39, 2008.

M. Haouari, S. Layeb, and H. Sherali. Algorithmic expedients for the prize collecting Steiner tree problem. *Discrete Optimization*, 7(1-2):32–47, 2010.

R. Hassin, R. Ravi, and F. Salman. Facility location on a network with unreliable links. In *Proceedings of INOC 2009*. University of Pisa, 2009.

R. Hemmecke, R. Schültz, and D. Woodruff. Interdicting stochastic networks with binary interdiction effort. In D. Woodruff, editor, *Network Interdiction and Stochastic Integer Programming*, volume 22 of *Operations Research/Computer Science Interfaces Series*, pages 69–84. Springer US, 2003.

D. S. Hochbaum and A. Pathria. Node-optimal connected k-subgraphs, 1994. manuscript, UC Berkeley.

S. Huang. *A constraint optimization framework for discovery of cellular signaling and regulatory networks*. PhD thesis, Massachusetts Institute of Technology, June 2011.

S. Huang and E. Fraenkel. Integration of proteomic, transcriptional, and interactome data reveals hidden signaling components. *Science Signaling*, 2(81):ra40, 2009.

T. Ideker, O. Ozier, B. Schwikowski, and A. Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(Supplement 1):S233–S240, 2002.

The `igraph` Project. The `igraph` library for complex network research, 2012. URL http://igraph.sourceforge.net/.

E. Israeli and R. Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

D. Johnson, M. Minkoff, and S. Phillips. The prize collecting steiner tree problem: theory and practice. In D. Shmoys, editor, *Proceedings of the 11th Symposium on Discrete Algorithms*, pages 760–769. ACM/SIAM, 2000.

K. Kaparis and A. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming*, Series B(124):69–91, 2010.

O. Karasan, M. Pinar, and H. Yaman. The robust shortest path problem with interval data. *Technical Report Bilkent University*, 2004.

A. Kasperski. *Discrete Optimization with Interval Data: Minmax Regret and Fuzzy Approach*. Springer, studies in fuzzines and soft computing edition, 2008.

A. Kasperski and P. Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5): 177–180, 2006.

A. Kasperski, M. Makuchowski, and P. Zieliński. A tabu search algorithm for the minmax regret minimum spanning tree problem with interval data. *Journal of Heuristics*, 14(4):1391–402, 2012.

H. Kerivin and A. Mahjoub. Design of survivable networks: A survey. *Networks*, 46 (1):1–21, 2005.

L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008. ISSN 1432-4350.

G. Klau, I. Ljubić, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting steiner tree problem. In K. Deb, editor, *Proceedings of GECCO 2004*, volume 3102 of *LNCS*, pages 1304–1315. Springer, 2004.

O. Klopfenstein and D. Nace. Cover inequalities for robust knapsack sets - application to the robust bandwidth packing problem. *Networks*, 51(1):59–72, 2012.

T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

A. Koster, M. Kutschka, and C. Raack. Robust network design: Formulations, valid inequalities, and computations. *Networks*, 61(2):128–149, 2013.

P. Kouvelis and G. Yu, editors. *Robust discrete optimization and its applications*. Nonconvex Optimization and its Applications. Kluwer Academic Publishers, 1st edition, 1997.

A. Kuehn and M. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

M. Labbé, R. Séguin, P. Soriano, and C. Wynants. Network synthesis with non-simultaneous multicommodity flow requirements: Bounds and heuristics, 1999.

G. Laporte and F. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

C. Lasher, C. Poirel, and T. Murali. Cellular response networks. In L. Heath and N. Ramakrishnan, editors, *Problem Solving Handbook in Computational Biology and Bioinformatics*, pages 233–252. Springer, 2011.

C. Lee, K. Lee, K. Park, and S. Park. Branch-and-price-and-cut approach to the robust network design problem without flow bifurcations. *Operations Research*, 60 (3):604–610, 2012.

H. Lee and D. Dooly. Decomposition algorithms for the maximum-weight connected graph problem. *Naval Research Logistics*, 45:817–837, 1998.

A. Letchford and S. Miller. Fast bounding procedures for large instances of the simple plant location problem. *Computers & Operations Research*, 39(5):985–990, 2012.

Q. Li, B. Zeng, and A. Savachkin. Reliable facility location design under disruptions. *Computers & OR*, 40(4):901–909, 2013.

C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. Recoverable robustness. *Technical Report ARRIVAL-TR-0066, ARRIVAL Project*, 2007.

C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. Ahuja, R. Möhring, and C. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 1–/27. Springer, 2009.

I. Ljubić. *Exact and Memetic Algorithms for Two Network Design Problems.* PhD thesis, Vienna University of Technology, 2004.

I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Progamming*, Series B(105):427–449, 2006.

I. Ljubić, P. Mutzel, and B. Zey. Stochastic survivable network design problems. *Electronic Notes in Discrete Mathematics*, 41:245,Äì–252, 2013.

A. Lucena and M. G. Resende. Strong lower bounds for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1-3):277–294, 2004.

T. Magnanti and R. Wong. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, 29(3):464–484, 1981.

S. Mattia. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54(3):619–643, 2013.

k. Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

M. Minoux. Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. *Annals of Discrete Mathematics (11) Studies on Graphs and Discrete Programming*, 59:269–277, 1981.

P. Mirchandani. The multi-tier tree problem. *INFORMS Journal on Computing*, 8(3): 202–218, 1996.

R. Montemmani. The robust shortest path problem with interval data via benders decomposition. *4OR*, 3(4):315–328, 2005.

R. Montemmani. A Benders descomposition approach for the robust spanning tree problem with interval data. *European Journal of Operational Research*, 174(3):1479–1490, 2006.

R. Montemmani and L. Gambardella. A branch and bound algorithm for the robust spanning tree with interval data. *European Journal of Operational Research*, 161(3): 771–779, 2005.

R. Montemmani, J. Barta, M. Mastrolilli, and L. Gambardella. The robust traveling salesman problem with interval data. *Transportation Science*, 41(3):366–381, 2007.

A. Moss and Y. Rabani. Approximation algorithms for constrained node weighted Steiner tree problems. *SIAM Journal on Computing*, 37(2):460–481, 2007.

J. Mulvey, R. Vanderbei, and S. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2):264–281, 1995.

A. Murray. An overview of network vulnerability modeling approaches. *GeoJournal*, 78(2):209–221, 2013.

Y. Nikulin. Simulated annealing algorithm for the robust spanning tree. *Journal of Heuristics*, 18(4):593–625, 2008.

Goldschmidt. O. and D. Hochbaum. k-edge subgraph problems. *Discrete Applied Mathematics*, 74(2):159–169, 1997.

C. Obreque, M. Donoso, G. Gutiérrez, and V. Marianov. A branch and cut algorithm for the hierarchical network design problem. *European Journal of Operational Research*, 200(1):28 – 35, 2010.

F. Ortega and L. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.

PAGASA. Philippine Atmospheric, Geophysical and Astronomical Services Administration, Department of Science and Technology, Republic of the Philippines, 2014. URL http://web.pagasa.dost.gov.ph/.

J. Pereira and I. Averbakh. Exact and heuristic algorithms for the interval data robust assignment problem. *Computers & Operations Research*, 38(8):1153–1163, 2011a.

J. Pereira and I. Averbakh. The robust set covering problem with interval data. *Annals of Operations Research*, 207(1):1–19, 2011b.

F. Pérez-Galarce, E. Álvarez-Miranda, A. Candia, and P. Toth. On exact solutions for the minmax regret spanning tree problem. *Accepted for publication in Computers & OR*, 2014.

C. Phillips. The network inhibition problem. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 776–785. ACM, 1993.

H. Pirkul, J. Current, and V. Nagarajan. The hierarchical network design problem: A new formulation and solution procedures. *Transportation Science*, 25(3):175–182, 1991.

PlanetLisa, September 10th 2012. url = http://www.planet-lisa.net/.

A. Prodon, S. De Negre, and T. Liebling. Locating leak detecting sensors in a water distribution network by solving prize-collecting Steiner arborescence problems. *Mathematical Programming*, Series B(124):119–141, 2010.

R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006.

W. Rei, J-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345, 2009.

J. Rosenhead, M. Elton, and S. Gupta. Robustness and optimality as criteria for strategic decisions. *Operational Research Quarterly*, 23(4):413–431, 1972.

E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007.

A. Ruszczyński and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, 1st edition, 2003.

Y. Saad and M. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7):867–872, 1988.

A. Salles da Cunha, A. Lucena, N. Maculan, and M. Resende. A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 157(6):1198–1217, 2009.

J. Salmeron, K. Wood, and R. Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems*, 24(1):96–104, 2009.

N. Sancho. A suboptimal solution to a hierarchial network design problem using dynamic programming. *European Journal of Operational Research*, 83(1):237–244, 1995.

L. Sanità. *Robust Network Design*. PhD thesis, Università La Sapienza, Roma, 2009.

L. Sanità. Private communication, 2013.

A. Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

Z. Shen, R. Zhan, and J. Zhang. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. *INFORMS Journal on Computing*, 23(3): 470–482, 2011.

L. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38(7): 547–564, 2006.

L. Snyder and M. Daskin. Reliability models for facility location: The expected failure cost case. *Transportation Science*, 39(3):400–416, 2005.

L. Snyder and M. Daskin. Reliability models for facility location: The expected failure cost case. *IIE Transactions*, 38(11):971–985, 2006.

O. Solyali, J. Cordeau, and G. Laporte. Robust inventory routing under demand uncertainty. *Transportation Science*, 46(3):327–340, 2012.

A. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

A. Thiele, T. Terry, and M. Epelman. Robust linear optimization with recourse. *Tech. Report TR09-01, University of Michigan*, 2009.

E. Uchoa. Reduction tests for the prize-collecting Steiner problem. *Operations Research Letters*, 34(4):437–444, 2007.

United Nations Statistics Division. UNSD Statistical Databases, 2013. URL http://millenniumindicators.un.org/unsd/databases.htm.

S. Uryasev and P. Pardalos, editors. *Stochastic Optimization: Algorithms and Applications*, volume 54 of *Series on Applied Optimization*. Springer, 1st edition, 2001.

R. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17(4): 638–663, 1967.

V. Verter. Uncapacitated and capacitated facility location problems. In H. A. Eiselt and V. Marianov, editors, *Foundations of Location Analysis*, volume 155 of *International Series in Operations Research & Management Science*, pages 25–37. Springer, 2011.

S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2011*, pages 1401–1408. IEEE, 2011.

A. Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 46(2):265–280, 1945.

R. Wollmer. Removing arcs from a network. *Operations Research*, 12(6):934–940, 1964.

T. Yamamoto, H. Bannai, M. Nagasaki, and S. Miyano. Better decomposition heuristics for the maximum-weight connected graph problem using betweenness centrality. In J. Gama, V. Costa, A. Jorge, and P. Brazdil, editors, *Discovery Science*, volume 5808 of *LNCS*, pages 465–472. Springer, 2009.

H. Yaman, O. Karasan, and M. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

yWorks. yEd Graph Editor, 2012. URL http://www.yworks.com/.

L. Zhao and B. Zeng. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *Tech. Report - University of South Florida*, 2012.