FORMULAÇÕES E ALGORITMOS BASEADOS EM PROGRAMAÇÃO LINEAR INTEIRA PARA O PROBLEMA QUADRÁTICO DA ÁRVORE GERADORA MÍNIMA

DILSON LUCAS PEREIRA

FORMULAÇÕES E ALGORITMOS BASEADOS EM PROGRAMAÇÃO LINEAR INTEIRA PARA O PROBLEMA QUADRÁTICO DA ÁRVORE GERADORA MÍNIMA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Alexandre Salles da Cunha

Belo Horizonte, Minas Gerais

Março de 2014

DILSON LUCAS PEREIRA

FORMULATIONS AND ALGORITHMS BASED ON LINEAR INTEGER PROGRAMMING FOR THE QUADRATIC MINIMUM SPANNING TREE PROBLEM

Thesis presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Alexandre Salles da Cunha

Belo Horizonte, Minas Gerais
March 2014

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

FOLHA DE APROVAÇÃO

Formulações e algoritmos baseados em programação linear inteira para o problema quadrático da árvore geradora mínima

DILSON LUCAS PEREIRA

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

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

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

[illegible]
PROF. HENRIQUE PACCA LOUREIRO LUNA
Instituto de Computação - UFAL

[illegible]
PROF. LUIDI GELABERT SIMONETTI
Centro Tecnológico - UFF

[illegible]
PROF. MAURÍCIO CARDOSO DE SOUZA
Departamento de Engenharia de Produção - UFMG

Belo Horizonte, 26 de março de 2014.

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Pereira, Dilson Lucas

P436f Formulations and Algorithms Based on Linear Integer Programming for the Quadratic Minimum Spanning Tree Problem / Dilson Lucas Pereira — Belo Horizonte, 2014
xviii, 97 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Alexandre Salles da Cunha

1. Computação - Teses. 2. Programação (Matemática) - Teses. 3. Programação inteira - Teses.
3. Lagrange, funções de - Teses. I. Orientador.
II. Título

519.6*61(043)

Agradecimentos

O desenvolvimento deste trabalho só foi possível graças à contribuição de diversas pessoas:
Minha esposa, que sempre esteve ao meu lado, com seu amor, carinho e dedicação.
Meus pais e meu irmão, que sempre me deram todo o suporte e incentivo.
O Professor Alexandre, que me orientou com muito empenho, sempre se preocupou com que eu pudesse atingir uma boa formação, e me forneceu inúmeras lições, nos mais diferentes aspectos.
Os responsáveis pelo PPGCC, que me deram a oportunidade, e que se dedicam em manter o alto nível do programa.
As secretárias do DCC, que sempre me atenderam com prestatividade e paciência.
O CNPq e a CAPES, que forneceram o suporte financeiro para que eu pudesse realizar este trabalho.
A todos estes, meus sinceros agradecimentos.

Acknowledgements

Part of the research reported in this thesis was conducted while I was under the supervision of Michel Gendreau, at CIRRELT, Montreal - Canada. I stayed in Montreal for one year as a visiting student from the sandwich doctorate program of the Brazilian agency Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). I would like to thank Michel for his hospitality and for taking his time so that we could have many valuable meetings.

Abstract

The quadratic minimum spanning tree problem (QMSTP) is an NP-Hard problem that consists of finding a spanning tree of quadratic minimum cost of a graph. In this problem, besides a cost for each edge, the objective function also involves interaction costs for each pair of edges in the tree. In a particular case of QMSTP, the adjacent-only QMSTP (AQMSTP), interaction costs are defined only for adjacent edges. Despite having a particular cost structure, AQMSTP is as hard as QMSTP, in theory and in practice. We address both problems in this thesis.

For the general case, we first propose a linear integer programming (IP) formulation based on the reformulation-linearization technique (RLT). This formulation is by itself stronger than previous ones in the literature. Additionally, we also introduce a novel type of formulation. It stems from the idea of partitioning spanning trees into forests of a given fixed size. This idea yields a hierarchy of formulations of increasing strength. The larger the forests, the stronger the formulation. At the first hierarchy level, where the forests have the minimum possible size, i.e., they involve only one edge, we have precisely the RLT formulation. At the opposite end, where the forests have the maximum possible size, i.e., they are spanning trees, we have a formulation whose linear programming (LP) relaxation bound matches the optimal QMSTP solution value. Many possible relaxations of the hierarchy are studied. We give results with respect to the strength of their LP bounds and the difficulty of computing them. On the computational side, three Lagrangian relaxation (LR) procedures and two parallel branch-and-bound (BB) algorithms are developed. For the first time, several instances in the literature are solved to optimality, including some with 50 vertices.

Although successful for some types of problem instances, this general approach fails to appropriately address AQMSTP. This fact motivates an AQMSTP formulation that exploits its particular cost structure: We use the stars of the graph to formulate the problem as an IP that involves exponentially many rows and columns. We also consider a reformulation that arises by projecting the stars out of this formulation. The reformulation, although defined on a compact variable space, has exponentially many additional constraints. One of its main advantages lies on the fact that most optimization packages are capable of directly handling models with a small number of variables and large number of constraints, through separation callbacks. This facilitates the development of a BB algorithm.

In order to solve the LP relaxation of our AQMSTP formulation, we propose a row-and-column generation (RCG) algorithm. A cutting plane (CP) algorithm is proposed to solve the LP relaxation of its projection. We then develop a branch-and-cut-and-price (BCP) algorithm, based on RCG, and a branch-and-cut algorithm (BC), based on CP. Computational results show that the LP relaxation bounds provided by the AQMSTP formulation dominate the RLT based bounds we developed for the general QMSTP. As a consequence, instances with as many as 50 vertices are solved to proven optimality.

Resumo

O problema quadrático da árvore geradora mínima (QMSTP) é um problema NP-Difícil que consiste em encontrar uma árvore geradora de custo quadrático mínimo em um grafo. Neste problema, além de um custo para cada aresta, a função objetivo também involve custos de interação para cada par de arestas na árvore. Em um caso particular do QMSTP, o QMSTP só com custos de adjacência (AQMSTP), custos de interação são definidos somente para arestas adjacentes. Apesar desta estrutura de custo particular, o AQMSTP é tão difícil quanto o QMSTP, na teoria e na prática. Ambos os problemas são abordados nesta tese.

Para o caso geral, primeiramente propomos uma formulação de programação linear inteira (IP) baseada na técnica de reformulação-linearização (RLT). Esta formulação é por si só mais forte que formulações anteriores na literatura. Adicionalmente, também introduzimos um novo tipo de formulação, que parte da idéia de particionar árvores geradoras em florestas de um dado tamanho fixo. Esta idéia leva à uma hierarquia de formulações cada vez mais fortes. Quanto maiores as florestas, mais forte a formulação. No primeiro nível da hierarquia, onde as florestas têm o menor tamanho possível, i.e., envolvem apenas uma aresta, temos precisamente a formulação RLT. No lado oposto, onde as florestas têm o maior tamanho possível, i.e., são árvores geradoras, temos uma formulação cujo limite da relaxação de programação linear (LP) tem o mesmo valor que a solução ótima do QMSTP. Diversas possíveis relaxações da hierarquia são estudadas. Apresentamos resultados com respeito à força de seus limites de LP e à dificuldade de computá-los. No lado computacional, três procedimentos baseados em relaxação lagrangeana (LR) e dois algoritmos branch-and-bound (BB) paralelos são desenvolvidos. Pela primeira vez, diversas instâncias na literatura são resolvidas na otimalidade, incluindo algumas com 50 vértices.

Embora a abordagem geral obtenha sucesso em alguns tipos de instâncias, ela não se demonstra apropriada para tratar o AQMSTP. Este fato motiva uma formulação para o AQMSTP que explora sua estrutura de custo particular: Usamos as estrelas do grafo para formular o problema como um IP que envolve exponencialmente muitas linhas e colunas. Também consideramos a reformulação que surge ao projetarmos as estrelas para fora desta formulação. A reformulação, embora definida em um espaço de variáveis compacto, tem um número exponencial de restrições adicionais. Uma de suas principais vantagens está no fato de que a maioria dos pacotes de otimização são capazes de lidar diretamente com modelos com um número pequeno de variáveis e um número grande de restrições, por meio de callbacks de separação. Isto facilita o desenvolvimento de um algoritmo BB.

Para resolver a relaxação de LP de nossa formulação do AQMSTP, propomos um algoritmo de geração de linhas e colunas (RCG). Um algoritmo de planos de corte (CP) é proposto para resolver a relaxação de LP de sua projeção. Após isto, desenvolvemos um algoritmo branch-and-cut-and-price (BCP), baseado em RCG, e um algoritmo branch-and-cut (BC), baseado em CP. Resultados computacionais mostram que os limites de LP fornecidos pela formulação do AQMSTP dominam os limites baseados em RLT que desenvolvemos para o caso geral. Como consequência, instâncias com até 50 vértices são resolvidas na otimalidade.

Lista de Siglas

AQMSTP	adjacent-only quadratic minimum spanning tree problem
BB	branch-and-bound
BC	branch-and-cut
BCP	branch-and-cut-and-price
BQFP	boolean quadric forest polytope
BQP	boolean quadric polytope
CP	cutting plane
GMSTP	generalized minimum spanning tree problem
IP	linear integer programming
LP	linear programming
LR	Lagrangian relaxation
MSTP	minimum spanning tree problem
pDP	p-dispersion problem
QAP	quadratic assignment problem
QMSTP	quadratic minimum spanning tree problem
RCG	row-and-column generation
RLT	reformulation-linearization technique
SEC	subtour elimination constraint

Contents

Chapter 1

Introduction

Assume we are given a connected and undirected graph $G = (V, E)$, with $n = |V|$ vertices and $m = |E|$ edges, and a matrix $Q = (q_{ij})_{i,j \in E} \geq \mathbf{0}$ of *interaction costs* between the edges of G. The quadratic minimum spanning tree problem (QMSTP) is a quadratic 0-1 programming problem that consists of finding a spanning tree of G whose incidence vector $\mathbf{x} \in \mathbb{B}^m$ minimizes the function

$$\sum_{i,j \in E} q_{ij} x_i x_j .$$

QMSTP is NP-Hard, as proven by Assad and Xu [7] by a reduction from the quadratic assignment problem (QAP) [26, 10]. An interesting particular case of QMSTP is that in which interaction costs $q_{ij} = 0$ if edges i and j do not share an endpoint. This case is referred to as the adjacent-only QMSTP (AQMSTP). AQMSTP was also proven NP-Hard by Assad and Xu [7], by means of a reduction from the Hamiltonian path problem. Another important particular case is that in which Q is diagonal. In this case, the objective function becomes linear, and QMSTP reduces to the minimum spanning tree problem (MSTP), for which several polynomial time algorithms are known [42, 27].

1.1 Outline of the Thesis and Main Contributions

The thesis is divided in 5 main chapters and 4 appendices. Below we summarize each main section and its contributions.

- *Chapter 1: Introduction.* We began the present chapter with the definition of the two problems addressed in the thesis, QMSTP and AQMSTP. In Section 1.2 we give motivation for studying these two problems. The QMSTP and AQMSTP literature is reviewed in Section 1.3. Some notation used throughout the thesis is presented in Section 1.4.

- *Chapter 2: The Quadratic Minimum Spanning Tree Problem.* We start this chapter by studying for the first time a formulation derived from the reformulation linearization technique (RLT), in Section 2.1.1. We show that such a formulation dominates other formulations in the QMSTP literature and introduce a Lagrangian relaxation lower bounding scheme to evaluate its linear programming (LP) relaxation bounds.

  In Section 2.1.2, we present a novel type of formulation for QMSTP, based on the idea of decomposing spanning trees into forests. That formulation gives a hierarchy of lower bounds of increasing strength, it has the first formulation as its weakest case. With the purpose of developing computationally practical lower bounding procedures based on it, we study possible relaxations. We show that some of the relaxations are theoretically hard to solve, but manage to give a polynomial time algorithm for a particular case, which we use to develop a Lagrangian relaxation based bounding algorithm, in Section 2.1.2.1. A different approach is taken in Section 2.1.2.2, we provide a reformulation of the hierarchy and show that an easy to solve relaxation can used in a Lagrangian framework to evaluate LP lower bounds in our hierarchy. A Lagrangian relaxation algorithm, with heuristic adjustment of multipliers, is presented.

  Based on two of the lower bounding schemes developed in the preceding sections, we propose two branch-and-bound (BB) algorithms in Section 2.2. We provide computational results in Section 2.3. These results show that our lower bounds are much stronger than the bounds in the literature. As a consequence, our exact algorithms manage to solve problems with up to 50 vertices, with the help of parallel programming.

- *Chapter 3: The Adjacent-Only Minimum Spanning Tree Problem.* In our study of the general QMSTP, we identified that the existing approaches in the literature fail to adequately address the AQMSTP case. We discuss reasons for that in Section 3.1. In Section 3.2 we present an AQMSTP formulation that takes its particular cost structure into account, i.e., it uses stars of the graph as modeling entities. As a result, the formulation is naturally integer. Since the formulation is not compact, in Section 3.3 we develop a row and column generation algorithm for computing its LP bounds. That algorithm is then used as the basis for an

AQMSTP branch-and-price algorithm. We provide computational results that validate our approach: lower bounds from the star formulation are much stronger the previous ones in the literature, including those presented in Chapter 2. As a consequence, the branch-and-price algorithm solves instances with up to 40 vertices, while the algorithms developed in Chapter 2 manage to solve instances with at most 20 vertices.

In Section 3.4 we study an unusual approach for obtaining lower bounds from extended formulations with exponentially many columns. We project the variables associated to stars out of our AQMSTP formulation. That results in an AQMSTP reformulation on a compact variable space, with exponentially many constraints. To evaluate its LP bounds, we propose a cutting plane algorithm, which is also used as the basis for an AQMSTP branch-and-cut algorithm. We discuss the benefits of using the projection approach, such as easier to solve pricing problems and easier to implement exact algorithms. Surprisingly, the branch-and-cut algorithm manages to solve instances with up to 50 vertices.

- *Chapter 4: Future Work - The Quadratic Assignment Problem.* In this chapter we discuss some lines of research we intend to investigate in the future. In particular, we show how the hierarchy presented in Chapter 2 can be extended for QAP and discuss some possible approaches for obtaining bounds.

- *Chapter 5: Conclusion.* Concluding remarks are presented in this chapter.

- *Appendix A Separation Algorithms.* We provide polynomial time, maximum flow based, separation algorithms for some inequalities used in our RLT based formulation of Chapter 2.

- *Appendix B: Proofs.* Detailed proofs for some results in the thesis are presented in this appendix.

- *Appendix C: Considerations about the Work of Öncan and Punnen [35].* Some results presented in Öncan and Punnen [35] conflict with some results obtained by ourselves. This led us to identify some mistakes in that work. This issue, along with evidence for the correctness of the results in this thesis, are discussed in this appendix.

- *Appendix D: Detailed Branch-and-Bound Results.* Detailed BB results, for each instance in our test bed, are provided in this chapter.

1.2 Motivation

There are three main motivations for studying an NP-Hard combinatorial optimization problem. First, when studying such a problem, one might develop solution techniques and theoretical results that might be generalized or adapted for other problems, and thus become contributions for the general field of combinatorial optimization. One well known example is the case of the traveling salesman problem [14, 6].

The second reason involves real-world applications of the problem. QMSTP and AQMSTP applications can be found in the context of telecommunication, transportation, and hydraulic networks [7]. Below, we describe two examples of QMSTP and AQMSTP applications in more detail.

A QMSTP application can be found in the context of wireless sensor networks. In this context, the communication between sensor nodes occurs by means of radio transmission. Assuming that the radio frequency assigned for each possible communication link in the network has been defined beforehand, one might wish to find a communication spanning tree that minimizes the radio interference between pairs of links. By letting the off-diagonal entries of Q be some measure of the interference between pairs of links, and letting the diagonal entries be a zero-valued vector, the problem can be solved as a QMSTP.

For the AQMSTP case, consider the situation in which a company C_0 wants to create a telecommunications network whose topology is a tree spanning a set V that represents customers, computational devices, or other entities. Instead of building all of the communication links, the company might rent part of the infrastructure from other companies $C_1, C_2, \ldots, C_T$. Let E_0 denote the set of links that might possibly be built by C_0. Assume that each company a, $1 \leq a \leq T$, owns a set of links E_a, and let $E = \cup_{a=0}^{T} E_a = E$. Without loss of generality we can also assume $E_a \cap E_b = \emptyset$ for any pair of companies $0 \leq a < b \leq T$. For each communication link i, there is a cost q_{ii} for constructing the link, in case $i \in E_0$, or for renting the link from company a, if $i \in E_a$, $1 \leq a \leq T$. If we decide to use links $i = \{u, v\}$ and $j = \{v, w\}$, and $i \in E_a$ and $j \in E_b$, $b \neq a$, then, in addition to the construction or renting costs q_{ii} and q_{jj}, *interface* costs q_{ij} and q_{ji} could also arise. The nature of these costs could be related to the acquisition of additional infrastructure to connect links that belong to different companies, or to switch from different communication technologies used by the companies, for example.

The third reason comes from problems that, being closely related to QMSTP and AQMSTP, may eventually benefit from theoretical and algorithmic developments for the latter. More generally, since QMSTP and AQMSTP are NP-Hard, any problem in the NP computational complexity class can be polynomially reduced to them.

Thus, good algorithms for QMSTP and AQMSTP might imply in good algorithms for other hard problems. Assad and Xu [7] gave two examples: the polynomial reductions from QAP to QMSTP and from the Hamiltonian path problem to AQMSTP. We give another, the generalized minimum spanning tree problem (GMSTP) [34].

Consider a graph $\overline{G} = (\overline{V}, \overline{E})$, weights $(w_i)_{i \in \overline{E}}$, and a partition $(\overline{V}_1, \ldots, \overline{V}_K)$ of $\overline{V}$. The generalized minimum spanning tree problem asks for a minimum weight tree of $\overline{G}$ such that exactly one vertex from each cluster V_k is in the tree. To reduce this problem to a QMSTP, let $G = (V, E)$, where $V = \overline{V}$ and $E = \overline{E} \cup \{\{i, j\} : i \neq j \in V_k, 1 \leq k \leq K\}$. Let $Q = (q_{ij})_{i,j \in E}$, where, for $i \in E$, $q_{ii} = w_i$ if $i \in \overline{E}$, and $q_{ii} = 0$ otherwise. For each V_k, $1 \leq k \leq K$, and for each pair of edges $i = \{u_i, v_i\}$ and $j = \{u_j, v_j\}$ such that $u_i \in V_k$, $u_j \in V_k$, $u_i \neq u_j$, $v_i \notin V_k$, and $v_j \notin V_k$, let $q_{ij} = M$, where M is a sufficiently large value. Then, any solution $\overline{T} = (V_{\overline{T}}, E_{\overline{T}})$ for GMSTP implies a solution $T = (V_T, E_T)$, with $V_T = \overline{V}$ and $E_T = E_{\overline{T}} \cup_{k=1}^{K} T_k$, where T_k is a tree of G spanning V_k, for QMSTP, both with the same objective value. Conversely, given a solution $T = (V_T, E_T)$, *with objective less than* M, $\overline{T} = (V_{\overline{T}}, E_{\overline{T}})$, where $E_{\overline{T}} = E_T \cap \overline{E}$ and $V_{\overline{T}}$ is the vertex set induced by $E_{\overline{T}}$, is a solution for GMSTP with the same objective value. Since GMSTP is NP-Hard, this transformation gives an alternative proof for the following well known result.

Proposition 1.1. *QMSTP is NP-Hard.* □

Although we are not aware of its computational complexity or real-world applications, we describe another interesting problem, that is an AQMSTP sub-case. Assume we are given a graph $G = (V, E)$, with costs q_{ii} for each $i \in E$, and we are asked to find a minimum cost connected subgraph of G such that, if edges $i = \{u, v\}$ and $j = \{v, w\}$ are in the solution, then $k = \{u, w\}$ must also be in the solution. A solution for this problem can be obtained by letting $q_{ij} = q_{kk}$ and solving the problem as an AQMSTP.

1.3 Literature Review

QMSTP and AQMSTP were introduced by Assad and Xu [7]. They proved that these problems are in the NP-Hard class by means of a polynomial reduction from the Hamiltonian path problem to AQMSTP. Additionally, a polynomial reduction from QAP to QMSTP was presented.

A common procedure for computing lower bounds for constrained quadratic 0-1 problems is that of Gilmore [22] and Lawler [28], the so called Gilmore-Lawler procedure/lower bound. The procedure was initially introduced for QAP and later adapted for other problems, e.g., the quadratic 0-1 knapsack problem [40]. The Gilmore-Lawler procedure has an important role in many exact solution algorithms for constrained quadratic 0-1 problems, where it is used either as a lower bounding procedure by itself or as a procedure for the resolution of subproblems in Lagrangian relaxation schemes.

Besides proving that QMSTP and AQMSTP are NP-Hard, Assad and Xu [7] developed a lower bounding scheme for QMSTP. It can be seen as a dual ascent algorithm for obtaining near optimal multipliers in a Lagrangian relaxation of a QMSTP formulation. Lagrangian subproblems were solved by means of the Gilmore-Lawler procedure. A branch-and-bound (BB) algorithm based on that bounding scheme managed to solve QMSTP instances defined over complete graphs with up to 12 vertices. For the AQMSTP case, due to the particular cost structure, they were able to reduce the computational complexity of the bounding scheme. In that case, the BB algorithm solved instances with up to 15 vertices. Three QMSTP heuristic algorithms were also proposed. Two of them were simple constructive heuristics, the other one was a Lagrangian heuristic that resulted from their lower bounding procedure. In general, these heuristics were not able find the optimal solutions for the problem instances used in that work.

Evolutionary algorithms were proposed by Zhout and Gen [48] and by Soak et al. [45]. The algorithm in [48] provided results of better quality compared to the heuristics in [7]. That algorithm was in turn outperformed by the best evolutionary algorithm in [45].

Öncan and Punnen [35] proposed a local search algorithm based on tabu thresholding. Computational experiments were presented where their heuristic compared favorably to the heuristics in [7] and [45]. Building upon the Lagrangian relaxation of Assad and Xu [7], Öncan and Punnen [35] also proposed a lower bounding procedure for QMSTP. They introduced new valid inequalities, that were relaxed and dualized in a Lagrangian fashion. Their lower bounding procedure relied on subgradient optimization for multiplier adjustment, and the Gilmore-Lawler procedure for solving Lagrangian subproblems. An exact BB algorithm was not implemented.

Three different heuristic approaches, based on simulated annealing, evolutionary algorithms, and tabu search, were proposed by Palubeckis et al. [38]. The algorithm based on tabu search outperformed the other two, but a comparison to other algorithms in the literature was not provided. Another heuristic algorithm, based on artificial bee colony, was proposed by Sundar and Singh [46]. Computational results were presented where that algorithm outperformed the approaches in Zhout and Gen [48] and Soak et al. [45].

Cordone and Passeri [13] proposed two heuristic algorithms, based on tabu search and variable neighbourhood search. They also proposed refinements for the QMSTP exact strategies of Assad and Xu [7]. Mainly, they proposed a modified version of the lower bounding procedure of [7], where a relaxation of the Lagrangian subproblem is solved. By doing so, they obtain weaker lower bound at a lower computational complexity. In addition to that, they introduced some variable fixing tests for the BB algorithm in that same reference. Their revised exact algorithm managed to solve QMSTP instances defined over complete graphs with up to 15 vertices, as well as larger sparse instances, with up to 20 vertices. They also presented results showing that their best performing heuristic, the one based on tabu search, provided better results than the previous heuristics in the literature.

More recently, a bi-objective version of AQMSTP was investigated by Maia et al. [31]. The authors considered $\sum_{i \in E} q_{ii} x_i$ and $\sum_{i,j \in E, i \neq j} q_{ij} x_i x_j$ as two separate objective functions. Their approach consisted in generating solutions in the Pareto front. To that aim, they proposed a local search heuristic and an adaptation of the BB algorithm of [7]. The adapted BB algorithm managed to generate the Pareto front for instances with up to 20 vertices. For those instances, the heuristic algorithm was capable of generating about two thirds of the solutions in the Pareto front.

A common approach to solve a problem formulated as a quadratic 0-1 program consists of linearizing the non-linear terms, in order to obtain a (mixed) integer linear program. Assuming that a set $\mathbf{x} = (x_i)_{i \in M}$ of variables is used to formulate the problem, the linearization is accomplished by introducing additional variables $\mathbf{y} = (y_{ij})_{i,j \in M}$, that replace the quadratic terms $x_i x_j$ in the objective function, together with new constraints to make sure that the condition $y_{ij} = x_i x_j$ holds. After linearization takes place, one is interested in describing the convex hull of integer points $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{x}$ is a feasible solution for the problem, e.g., $\mathbf{x}$ is the incidence vector of a spanning tree of G in the QMSTP case, and $\mathbf{y}$ satisfies $y_{ij} = x_i x_j$ for all $i, j \in E$.

The boolean quadric polytope (BQP) refers to the linearized polytope that results when $\mathbf{x}$ is any binary vector. Thus, valid inequalities for BQP are valid for all constrained quadratic 0-1 problems, QMSTP included. BQP has been widely studied, see Padberg [36], for instance. Closely related to the linearized QMSTP polytope is the so called boolean quadric forest polytope (BQFP). The BQFP case is a generalization of the QMSTP case, where $\mathbf{x}$ must be the incidence vector of a forest of G. Many facet defining inequalities for the BQFP were studied by Lee and Leung [29]. However, to our knowledge, no computational experiments with those inequalities were ever reported.

1.4 Notation

Throughout the thesis, the following notation is used. Given a subset $V' \subseteq V$ we denote by $E(V') = \{\{u, v\} \in E : u, v \in V'\}$ the set of edges with both endpoints in V' and by $\delta(V') = \{\{u, v\} \in E : u \in V', v \notin V'\}$ the set of edges with exactly one endpoint in V'. For simplicity, we use $\delta(v)$ instead of $\delta(\{v\})$ when referring to the set of edges that have $v \in V$ as an endpoint. Given any formulation P for QMSTP, we let $Z(P)$ denote its LP relaxation lower bounds. Finally, we denote the set $\{0, 1\}$ by $\mathbb{B}$.

Chapter 2

The Quadratic Minimum Spanning Tree Problem

In this Chapter, we address QMSTP. The chapter is divided in five sections. In the first one, Section 2.1, we propose new formulations for the problem. More precisely, in Section 2.1.1 we propose a formulation based on RLT and give a Lagrangian relaxation scheme for evaluating its LP relaxation bounds. That formulation is generalized into a hierarchy of formulations in Section 2.1.2. We present a deep study of formulations in the hierarchy and develop two more Lagrangian relaxation schemes. Two of these bounding schemes are then used to develop two QMSTP BB algorithms, in Section 2.2. Computational experiments are conducted in Section 2.3. Concluding remarks are given in Section 2.4.

2.1 Formulations, Linear Programming and Lagrangian Relaxation Bounds

2.1.1 Lagrangian Bounds from a Partial RLT Application

Consider a vector $\mathbf{x} = (x_i)_{i \in E}$ of binary variables such that $x_i = 1$ if and only if edge i is selected to be part of the QMSTP solution. A quadratic 0-1 programming formulation for QMSTP is given by:

$$\min \left\{ \sum_{i,j \in E} q_{ij} x_i x_j : \mathbf{x} \in X \cap \mathbb{B}^m \right\},$$

where X denotes the set of points in $\mathbb{R}^m$ that satisfy:

$$\sum_{i \in E} x_i = n-1, \tag{2.1}$$

$$\sum_{i \in E(S)} x_i \leq |S|-1, \quad S \subset V, |S| \geq 2, \tag{2.2}$$

$$0 \leq x_i \leq 1, \qquad i \in E. \tag{2.3}$$

Proposition 2.1. *X is the convex hull of the incidence vectors of spanning trees of G (This well known result is due to Edmonds [16]).* □

Proposition 2.2. *Given $i \in E$, the convex hull of the incidence vectors of spanning trees of G that contain i is given by $X_i = X \cap \{\mathbf{x} \in \mathbb{R}^m : x_i = 1\}$.*

Proof. Let X_i' be the convex hull of the incidence vectors of the spanning trees of G that contain i. Clearly, $X_i' \subseteq X_i$. Since $x_i \leq 1$ is valid for X and there is a spanning tree of G containing i, X_i induces a non-empty face of X. Thus any extreme point of X_i is also an extreme point of X. Together with Proposition 2.1, this implies that the extreme points of X_i are integer, which implies $X_i \subseteq X_i'$. □

To obtain a 0-1 IP formulation for QMSTP, we apply a scheme based on the reformulation-linearization technique (Caprara [11] gives an exposition of this approach for constrained quadratic 0-1 programs in general). The scheme consists of two steps :

Reformulation step. Each constraint (2.1)-(2.3) is multiplied by each variable x_i, $i \in E$, resulting in new, non-linear, constraints.
Linearization step. Linearization variables $\mathbf{y} = (y_{ij})_{i,j \in E}$ are introduced. On each constraint resulting from the multiplication by x_i, we replace the product $x_i x_j$, $i, j \in E$, by y_{ij}. On the objective function we also substitute y_{ij} for the product $x_i x_j$ on the term with coefficient q_{ij}.

Observe that we make explicit distinction between $y_{ij} = x_i x_j$ and $y_{ji} = x_j x_i$. This is done so that a special structure in the resulting formulation can be exploited. For convenience, we also replace the powers x_i^2 by y_{ii} instead of simply x_i. For each $i \in E$, let $\mathbf{y}_i = (y_{ij})_{j \in E}$ denote the row of $\mathbf{y}$ indexed by i. The application of the reformulation-linearization procedure yields the constraints

$$\sum_{j \in E} y_{ij} = (n-1)x_i, \tag{2.4}$$

$$\sum_{j \in E(S)} y_{ij} \leq (|S|-1)x_i, \quad S \subset V, |S| \geq 2, \tag{2.5}$$

$$y_{ii} = x_i, \tag{2.6}$$

$$0 \leq y_{ij} \leq x_i, \qquad j \in E. \tag{2.7}$$

Notice that (2.4)-(2.7) is simply X_i multiplied by x_i. Thus, once the linearization scheme has been applied, the following 0-1 IP formulation is obtained

$$F_1 : \qquad \min \left\{ \sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}) \in P_1 \cap \mathbb{B}^{m+m^2} \right\},$$

where P_1 refers to the polyhedral region defined by:

$$\mathbf{x} \in X, \tag{2.8}$$

$$\mathbf{y}_i \in (X_i \times x_i), \qquad i \in E, \tag{2.9}$$

$$y_{ij} = y_{ji}, \qquad i < j \in E. \tag{2.10}$$

By constraint (2.8), $\mathbf{x}$ must be the incidence vector of a spanning tree. By constraints (2.9), for each edge i with $x_i = 1$, $\mathbf{y}_i$ must be the incidence vector of a spanning tree containing i, and for each edge i with $x_i = 0$, $\mathbf{y}_i$ must be the null vector. Finally, constraints (2.10) enforces the equality of the trees implied by the vectors $\mathbf{y}_i$ and $\mathbf{x}$.

Observe that the tree defined by $\mathbf{y}_i$ determines the edges with which i interacts. For this reason, we refer to it as the *interaction tree* for edge i. Formulation F_1 requires that all interaction trees be identical, but this condition will be relaxed in our solution strategy.

The reformulation process just applied consists of a partial application of the first RLT level [4, 44]. The complete first RLT level would also involve the multiplication of (2.1)-(2.3) by $(1 - x_i)$, for all $i \in E$, followed by the linearization step. It can be seen that the application of this operation to (2.1) and (2.3) results in redundant constraints, while the following inequalities are obtained from (2.2):

$$\sum_{j \in E(S)} (x_j - y_{ij}) \leq (|S| - 1)(1 - x_i), \qquad i \in E, S \subset V, |S| \geq 2. \tag{2.11}$$

It was shown by Lee and Leung [29] that (2.5), (2.7), and (2.11) define facets of BQFP. Besides these, they showed that facets are also induced by the clique and the cut inequalities of BQP [36], and by an extension of the latter, also presented in [29]. They give LP based polynomial time separation algorithms for (2.5) and (2.11). We remark that these two sets of valid inequalities can also be separated by modified versions of the algorithm of Padberg and Wolsey [37] for separating subtour breaking constraints. We present these modifications in Section A of the Appendix. We are not aware of any polynomial time separation algorithms for the remaining inequalities presented in [29].

Preliminary computational experiments we conducted on the QMSTP instances with 15 vertices introduced in [35] indicated that constraints (2.11) do not significantly strengthen $Z(F_1)$. According to our experiments, such LP bounds increased by only 0.08%, while the average computational time needed for their evaluation (solving LPs by means of a a cutting planes approach) increased by 69.8%. For this reason, our formulation and solution techniques do not consider those inequalities.

We now present other formulations in the QMSTP literature and investigate how F_1 compares to them. Consider the following constraints:

$$\sum_{i \in E} y_{ij} = (n-1)x_j, \qquad j \in E, \tag{2.12}$$

$$\sum_{i \in \delta(v)} y_{ij} \geq x_j, \qquad j \in E, v \in V. \tag{2.13}$$

Assad and Xu [7] introduced the QMSTP formulation

$$F_{\text{AX92}}: \qquad \min \left\{ \sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}) \in P_{\text{AX92}} \cap \mathbb{B}^{m+m^2} \right\},$$

where $P_{\text{AX92}} = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{m+m^2} : (\mathbf{x}, \mathbf{y}) \text{ satisfies (2.8)-(2.9) and (2.12)} \right\}$. Öncan and Punnen [35] added the valid inequalities (2.13) to P_{AX92} and formulated QMSTP as

$$F_{\text{OP10}}: \qquad \min \left\{ \sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}) \in P_{\text{OP10}} \cap \mathbb{B}^{m+m^2} \right\},$$

where $P_{\text{OP10}} = P_{\text{AX92}} \cap \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{m+m^2} : (\mathbf{x}, \mathbf{y}) \text{ satisfies (2.13)} \right\}$.

Proposition 2.3. $P_{\text{AX92}} \supseteq P_{\text{OP10}} \supseteq P_1$

Proof. Constraints (2.12) are clearly implied by (2.4) and (2.10). To check that constraints (2.13) are also implied by P_1, formulate (2.5) in terms of an edge j and $S = V \setminus \{v\}$, for any $v \in V$. Then subtract the resulting inequality from (2.4) (also formulated for $j \in E$), to obtain:

$$\sum_{i \in \delta(v)} y_{ji} \geq x_j,$$

which, together with (2.10), implies (2.13). □

Corollary 2.1. $Z(F_{\text{AX92}}) \leq Z(F_{\text{OP10}}) \leq Z(F_1)$. □

Due to the large number of variables and constraints that define F_1, computing $Z(F_1)$ by directly solving the LP relaxation of F_1, even with a cutting plane algorithm where (2.2) and (2.5) are dynamically separated, is not practical. Moreover, formulating X and X_i, $i \in E$, by means of network flow strategies would yield a QMSTP formulation with $O(n^7)$ variables and constraints. Solving the LP relaxation of that formulation would also not be practical. Therefore, we adopt an alternative strategy. We relax and dualize constraints (2.10) by attaching to them unconstrained Lagrangian multipliers $\theta = (\theta_{ij})_{i<j \in E}$ (assume $\theta_{ij} = -\theta_{ji}$ in case $i > j \in E$), to obtain the problem:

$$F_1'(\theta): \qquad L_1'(\theta) = \min \left\{ \sum_{i,j \in E} q_{ij}' y_{ij} : (\mathbf{x}, \mathbf{y}) \in P_1' \cap \mathbb{B}^{m+m^2} \right\},$$

where polytope P_1' is obtained by relaxing (2.10) in the definition of P_1, i.e., P_1' is given by (2.8) and (2.9). Lagrangian modified costs are defined as $q_{ij}' = q_{ij} + \theta_{ij}$ for $i \neq j \in E$ and $q_{ii}' = q_{ii}$ for $i \in E$. The corresponding Lagrangian dual is:

$$DF_1: \qquad L_1'^{*} = \max \left\{ L_1'(\theta) : \theta \in \mathbb{R}^{\frac{m(m-1)}{2}} \right\}.$$

Next, we present a result that states that P_1' has integer extreme points. This result is useful for determining the strength of DF_1.

Proposition 2.4. *P_1' is an integral polytope.*

Proof. Assume $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ is a fractional extreme point of P_1'. Assume further that $\overline{\mathbf{x}}$ is integer. Then, for any $i \in E$, $\overline{\mathbf{y}}_i \in X_i$. Thus, we can write $\overline{\mathbf{y}}_i = \sum_{\mathbf{w} \in \mathcal{E}_i} \lambda^{\mathbf{w}} \mathbf{w}$, where $\mathcal{E}_i$ is a subset of the extreme points of X_i, $\lambda^{\mathbf{w}} > 0$ for all $\mathbf{w} \in \mathcal{E}_i$, and $\sum_{\mathbf{w} \in \mathcal{E}_i} \lambda^{\mathbf{w}} = 1$. Observe that

$$(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \overline{\mathbf{y}}_{j_2}, \ldots, \overline{\mathbf{y}}_i, \ldots, \overline{\mathbf{y}}_{j_m}) = \sum_{\mathbf{w} \in \mathcal{E}_i} \lambda^{\mathbf{w}} (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \overline{\mathbf{y}}_{j_2}, \ldots, \mathbf{w}, \ldots, \overline{\mathbf{y}}_{j_m}),$$

and thus $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ cannot be an extreme point of P_1'.

Now, assume that $\overline{\mathbf{x}}$ is fractional. Since $\overline{\mathbf{x}} \in X$, we have $\overline{\mathbf{x}} = \sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{w}$, where $\mathcal{E}$ is a subset of the extreme points of X, $\lambda^{\mathbf{w}} > 0$ for all $\mathbf{w} \in \mathcal{E}$, and $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}} = 1$. For each $\mathbf{w} \in \mathcal{E}$ and $i \in E$ consider $\mathbf{z}_i^{\mathbf{w}} \in \mathbb{B}^m$, defined as follows. If $w_i = 1$, let $\mathbf{z}_i^{\mathbf{w}} = \overline{\mathbf{y}}_i/\overline{x}_i$, otherwise, let $\mathbf{z}_i^{\mathbf{w}} = 0$. Note that $(\mathbf{w}, \mathbf{z}_{j_1}^{\mathbf{w}}, \ldots, \mathbf{z}_{j_m}^{\mathbf{w}}) \in P_1'$.

Consider some $i \in E$. If $\overline{x}_i = 0$, $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{z}_i^{\mathbf{w}} = \overline{\mathbf{y}}_i = \mathbf{0}$. On the other hand, if $\overline{x}_i > 0$, $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{z}_i^{\mathbf{w}} = \left(\sum_{\mathbf{w}\in\mathcal{E}:w_i=1} \lambda^{\mathbf{w}}\right) \frac{\overline{\mathbf{y}}_i}{\overline{x}_i} = \overline{\mathbf{y}}_i$. Thus,

$$(\overline{\mathbf{x}}, \overline{\mathbf{y}}) = (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \overline{\mathbf{y}}_{j_2}, \ldots, \overline{\mathbf{y}}_{j_m}) = \sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}(\mathbf{w}, \mathbf{z}_{j_1}^{\mathbf{w}}, \ldots, \mathbf{z}_{j_m}^{\mathbf{w}}),$$

and $(\overline{\mathbf{x}}, \mathbf{y})$ cannot be an extreme point. □

From proposition 2.4 and from a well known result in Lagrangian duality theory [20], we have the following result.

Corollary 2.2. $L_1'^* = Z(F_1)$. □

Observe that after the relaxation of (2.10), the interaction trees for each $i \in E$ become independent from each other. Consequently, it is possible to develop a specialized procedure for solving $F_1'(\theta)$. Let $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ be an optimal solution for $F_1'(\overline{\theta})$, for some set of multipliers $\overline{\theta}$. If $\overline{x}_i = 0$ for some $i \in E$, then $\overline{\mathbf{y}}_i = \mathbf{0}$. On the other hand, if $\overline{x}_i = 1$, $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ can be so that $\overline{\mathbf{y}}_i$ is the incidence vector of a spanning tree that solves:

$$\overline{q}_i = \min\left\{\sum_{j\in E} q'_{ij}y_{ij} : \mathbf{y}_i \in X_i \cap \mathbb{B}^m\right\}. \tag{2.14}$$

This means that the selection of edge i implies in a cost given by $\overline{q}_i$. Consequently, one can solve $F_1'(\overline{\theta})$ by computing the spanning tree $\overline{\mathbf{x}}$ that minimizes:

$$\overline{q}_0 = \min\left\{\sum_{i\in E} \overline{q}_i x_i : \mathbf{x} \in X \cap \mathbb{B}^m\right\}, \tag{2.15}$$

and setting $\overline{\mathbf{y}}$ accordingly. Algorithm 1 summarizes the main steps of the procedure. Note that when $\overline{\theta} = \mathbf{0}$, Algorithm 1 provides the Gilmore-Lawler lower bound for QMSTP.

Algorithm 1:
Input: A QMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. A set of Lagrangian multipliers $\overline{\theta} \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: A solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ for $F_1'(\overline{\theta})$.

1. For each edge $i \in E$, solve (2.14) and denote the solution vector by $\widetilde{\mathbf{y}}_i$.
2. Solve (2.15), denote by $\widetilde{\mathbf{x}}$ the solution vector.
3. An optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ of cost $L_1(\overline{\theta}) = \overline{q}_0$ for $F_1'(\overline{\theta})$ is given by $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$ and $\overline{\mathbf{y}}_i = \widetilde{x}_i \widetilde{\mathbf{y}}_i$ for all $i \in E$.

Each minimum spanning tree problem in Algorithm 1 can be solved in $O(m \log n)$ time complexity (with Prim's algorithm [42]). Thus, Algorithm 1 can be implemented to run in $O(m^2 \log n)$.

An approximate solution for the Lagrangian dual DF_1 can be obtained by means of the subgradient method [25]. We denote by Lag_1 the algorithm that uses the subgradient method to solve DF_1 while solving subproblems by means of Algorithm 1. Based on the observations above, for the practical evaluation of $Z(F_1)$, it is reasonable to expect Lag_1 to perform better than directly solving the LP relaxation of F_1. In Section 2.2 we develop a BB algorithm that uses Lag_1 as its bounding procedure. Computational results for Lag_1 are reported in Section 2.3.

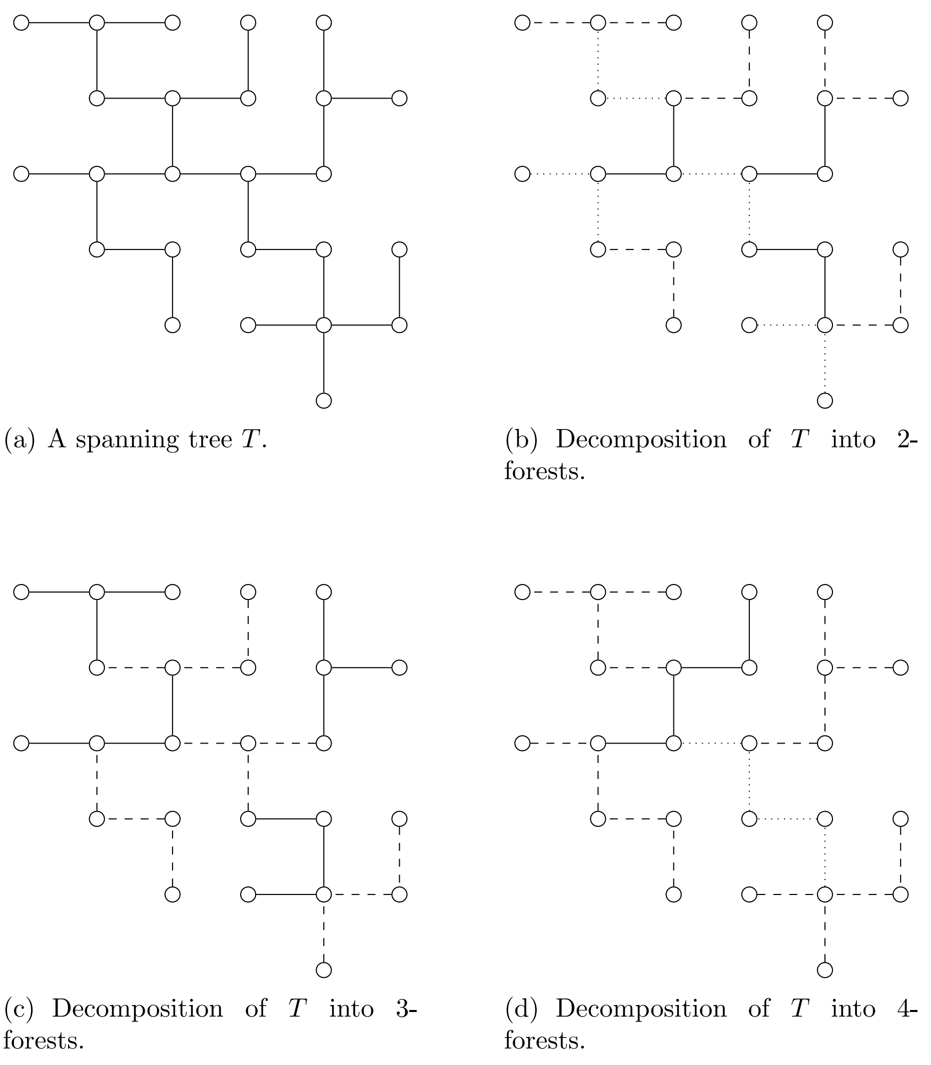
Although formulation F_1 is at least as strong as F_{AX92} and F_{OP10}, duality gaps implied by $Z(F_1)$ are sometimes quite large. This observation motivates the study of stronger lower bounding approaches for QMSTP.

2.1.2 Lagrangian Bounds Based on the Decomposition of Spanning Trees into Forests of Fixed Size

Let K be a positive integer that divides $n-1$. Then, the edge set of any spanning tree T of G can be partitioned into $(n-1)/K$ acyclic subsets, with K edges each. In other words, T can be decomposed into $(n-1)/K$ forests of K edges each. We will refer to such forests as K-forests. Examples are given in Figure 2.1 for $K = 2$, 3, and 4.

The basic idea behind the formulation we introduce in this section is to combine K-forests to create spanning trees. To formulate the interaction costs, instead of defining interaction trees for edges, as we did in F_1, we define interaction trees for K-forests. In doing so, even if we relax the requirement that these trees must be equal, the interaction trees for the edges in the same K-forest will remain the same. Therefore, the larger the value of K, the closer we are to satisfying the equality constraints. This suggests that this formulation could be stronger than F_1.

Some additional notation is needed before we introduce the formulation. Denote


(a) A spanning tree T.

(b) Decomposition of T into 2-forests.

(c) Decomposition of T into 3-forests.

(d) Decomposition of T into 4-forests.

Figure 2.1. Decomposition of a spanning tree into K-forests, for $K = 2$, 3 and 4. K-forests are represented by connected components with the same drawing style. In the examples, all K-forests are connected subgraphs of G. Observe however, that this is not always possible in general.

by $E^K = \{H \subseteq E : H \text{ is } K\text{-forest-inducing}\}$ the set of all K-forest-inducing subsets of E and let $o = |E^K|$. Define $E_i^K = \{H \in E^K : i \in H\}$ as the set of the elements of E^K that contain $i \in E$. Finally, for all $H \in E^K$, let $X_H = X \cap \{\mathbf{x} \in \mathbb{R}^m : x_i = 1, i \in H\}$. Using similar arguments to those used in the proof of Proposition 2.2 we can show that X_H is the convex hull of the incidence vectors of spanning trees containing all edges in H.

Besides variables $\mathbf{x}$ and $\mathbf{y}$ introduced before, the formulation also employs binary variables $\mathbf{s} = (s_H)_{H \in E^K}$, such that $s_H = 1$ if and only if $H \in E^K$ is selected to be part of the solution. To formulate interaction trees, binary variables $\mathbf{t} = (t_{Hi})_{H \in E^K, i \in E}$ are used. We denote the row of $\mathbf{t}$ indexed by $H \in E^K$ by $\mathbf{t}_H = (t_{Hi})_{i \in E}$, i.e., $\mathbf{t}_H$ defines the interaction tree for the K-forest induced by H. QMSTP can be formulated as:

$$F_2: \qquad \min \left\{ \sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P_2 \cap \mathbb{B}^{m+m^2+o+om} \right\},$$

where polytope P_2 is given by:

$$\mathbf{x} \in X, \tag{2.16}$$

$$x_i = \sum_{H \in E_i^K} s_H, \qquad i \in E, \tag{2.17}$$

$$\mathbf{t}_H \in (X_H \times s_H), \quad H \in E^K, \tag{2.18}$$

$$\mathbf{y}_i = \sum_{H \in E_i^K} \mathbf{t}_H, \qquad i \in E, \tag{2.19}$$

$$y_{ij} = y_{ji}, \qquad i < j \in E. \tag{2.20}$$

Constraints (2.16) require that $\mathbf{x}$ define a spanning tree of G. The decomposition of the spanning tree into K-forests is enforced by constraints (2.17). By (2.18), if a K-forest-inducing set H is used in the decomposition, then an interaction tree must be defined for it, otherwise, $\mathbf{t}_H = \mathbf{0}$. Constraints (2.19) state that, in case edge i appears in a K-forest that belongs to the solution, then the interaction tree $\mathbf{y}_i$ for i must be equal to the interaction tree for that K-forest, otherwise, $\mathbf{y}_i = \mathbf{0}$. Finally, by (2.20), all spanning trees must be equal.

Note that the size of E^K is $O(m^K)$, which is polynomial in n if K is a constant. Thus, the size of F_2 in relation to F_1 is polynomially bounded. Regarding the strength of F_2, when $K = 1$, formulations F_1 and F_2 are equivalent. If $K = n-1$, E^K will be the set of all spanning trees of G and, consequently, $Z(F_2)$ will be the optimal solution value of QMSTP. For general values of K, we have the following result.

Proposition 2.5. *Given a factor $K > 0$ of $n-1$, denote by $P_2(K)$ the polytope defined by* (2.16)-(2.20) *for this particular value of K. Let $Proj_{\mathbf{xy}}(P_2(K))$ be the projection of $P_2(K)$ onto the vector space of the variables $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{m+m^2}$. For two factors K and L of $n-1$, $L > K$, the following holds:*

$$Proj_{\mathbf{xy}}(P_2(L)) \subseteq Proj_{\mathbf{xy}}(P_2(K)).$$

Proof. Consider a vector $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}}) \in P_2(L)$. We are going to show that there is a vector $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}, \widetilde{\mathbf{s}}, \widetilde{\mathbf{t}}) \in P_2(K)$ such that $\widetilde{\mathbf{x}} = \overline{\mathbf{x}}$ and $\widetilde{\mathbf{y}} = \overline{\mathbf{y}}$.

For every $H \in E^L$ define the set $E^K(H) = \{I \in E^K : H \cap I = K\}$, i.e., $E^K(H)$ is the set of the subsets of H that contain K edges. Edges in H appear in exactly

$c = (L-1)!/((L-K)!(K-1)!)$ elements of $E^K(H)$.

Now, for all $I \in E^K$ let

$$\widetilde{s}_I = \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} \overline{s}_H, \tag{2.21}$$

and

$$\widetilde{\mathbf{t}}_I = \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} \overline{\mathbf{t}}_H. \tag{2.22}$$

Thus, from (2.21), for any $i \in E$, we have

$$\overline{x}_i = \sum_{H \in E_i^L} \overline{s}_H = \sum_{H \in E_i^L} \frac{1}{c} \sum_{I \in E^K(H) : i \in I} \overline{s}_H = \sum_{I \in E_i^K} \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} \overline{s}_H = \sum_{I \in E_i^K} \widetilde{s}_I = \widetilde{x}_i,$$

which shows that (2.16) and (2.17) are satisfied in $P_2(K)$. From (2.22) we have

$$\overline{\mathbf{y}}_i = \sum_{H \in E_i^L} \overline{\mathbf{t}}_H = \sum_{H \in E_i^L} \frac{1}{c} \sum_{I \in E^K(H) : i \in I} \overline{\mathbf{t}}_H = \sum_{I \in E_i^K} \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} \overline{\mathbf{t}}_H = \sum_{I \in E_i^K} \widetilde{\mathbf{t}}_I = \widetilde{\mathbf{y}}_i,$$

which shows that (2.19) and (2.20) are also satisfied in $P_2(K)$. Finally, we need to show that (2.18) is also satisfied. We will only show that, for any $I \in E^K$, the cardinality constraint $\sum_{i \in E} t_{Ii} = (n-1)s_I$ is satisfied. The satisfaction of the remaining constraints can be proved in a similar fashion. Using both (2.21) and (2.22),

$$\sum_{i \in E} \widetilde{t}_{Ii} = \sum_{i \in E} \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} \overline{t}_{Hi} = \frac{1}{c} \sum_{H \in E^L : I \in E^K(H)} (n-1)\overline{s}_H = (n-1)\widetilde{s}_I,$$

which proves the satisfaction of the desired constraint. □

For practical considerations, if K does not divide $n-1$, we can add artificial vertices to the graph, together with (zero cost) edges to keep the graph connected. Note also that the decomposition of a spanning tree of G into K-forests might not be unique. However, all that is needed to properly formulate QMSTP is granting that at least one such decomposition exists for any spanning tree. By eliminating redundant possibilities from E^K, we can reduce the number of variables of F_2 and maybe improve its LP relaxation bound. The next result shows how that can be accomplished for $K \leq 4$.

Proposition 2.6. *Let T be a spanning tree of G and K be a factor of $n-1$. If $K = 2$, T can be decomposed into trees of two edges each. If $K = 3$ or $K = 4$, then T can be* *decomposed into K-forests, none of which has more than two connected components.*

Proof. Note that, for $K = 2$, either T has edges $\{i, j\}$ and $\{j, k\}$ such that i and k are leaves, or i is a leaf and j is not connected to any vertex other than i or k. No matter the case, we remove these two edges to obtain a subgraph of T that is connected and has an even number of edges. The argument is then applied recursively.

For $K = 3$, remove $(1/3)(n - 1)$ edges $\{i, j\}$ of T, one at a time, under the condition that i is a leaf. The remaining subgraph has $(2/3)(n - 1)$ edges and is connected; apply the procedure for $K = 2$ to this subgraph. For each resulting set of two edges, add one of the edges that were previously removed.

For $K = 4$, apply the procedure for $K = 2$, group the resulting pairs of adjacent edges into sets of four edges. □

Even in the light of Proposition 2.6, computing $Z(F_2)$ by explicitly solving LPs is impractical, due to the large number of variables and constraints that compose F_2. Recall that formulation F_2 was developed with the relaxation of constraints (2.20) in mind. We will now investigate the relaxation and the dualization of those constraints in a Lagrangian fashion. To that aim, consider again that unconstrained multipliers $\theta = (\theta_{ij})_{i<j\in E}$, defined as before, are assigned to (2.20). Such a relaxation strategy leads to the following Lagrangian subproblem:

$$F_2'(\theta): \quad L_2'(\theta) = \min \left\{ \sum_{i,j\in E} q'_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P_2' \cap \mathbb{B}^{m+m^2+o+om} \right\},$$

where P_2' is obtained by relaxing (2.20) in P_2, i.e., P_2' is represented by (2.16)-(2.19). Lagrangian modified costs are defined as $q'_{ij} = q_{ij} + \theta_{ij}$ for all $i \neq j \in E$ and $q'_{ii} = q_{ii}$ for all $i \in E$.

Observe that, by (2.19), the objective function of F_2' can be written as:

$$\sum_{i,j\in E} q'_{ij} y_{ij} = \sum_{i,j\in E} \sum_{H\in E_i^K} q'_{ij} t_{Hj} = \sum_{H\in E^K} \sum_{i\in H} \sum_{j\in E} q'_{ij} t_{Hj}.$$

Therefore, using the fact that in F_2' the choice of the spanning tree $\mathbf{t}_H$ depends only on s_H, it can be concluded that in an optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ for $F_2'(\overline{\theta})$, for some particular choice of multipliers $\overline{\theta}$, if $\overline{s}_H = 1$, $\overline{\mathbf{t}}_H$ will be the incidence vector of a spanning tree that minimizes:

$$\overline{q}_H = \min \left\{ \sum_{i\in H} \sum_{j\in E} q'_{ij} t_{Hj} : \mathbf{t}_H \in X_H \cap \mathbb{B}^m \right\}. \tag{2.23}$$

Thus, problem F_2' can be solved with the resolution of

$$\overline{q}_0 = \min \left\{ \sum_{H \in E^K} \overline{q}_H s_H : \mathbf{x} \in X; x_i = \sum_{H \in E_i^K} s_H, \forall i \in E; (\mathbf{x}, \mathbf{s}) \in \mathbb{B}^{m+o} \right\}, \tag{2.24}$$

followed by the appropriate adjustment of $\mathbf{y}$ and $\mathbf{t}$. This process is summarized in the following algorithm.

Algorithm 2:
Input: A QMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. A factor K of $n-1$. A set of Lagrangian multipliers $\overline{\theta} \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: A Solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ for $F_2'(\overline{\theta})$.

1. For every $H \in E^K$, solve (2.23) and denote by $\widetilde{\mathbf{t}}_H$ its solution vector.
2. Solve problem (2.24) to obtain a solution $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{s}})$.
3. Obtain a solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ of cost $L_2'(\overline{\theta}) = \overline{q}_0$ for $F_2'(\overline{\theta})$ by making $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$, $\overline{\mathbf{s}} = \widetilde{\mathbf{s}}$, $\overline{\mathbf{t}}_H = \overline{s}_H \widetilde{\mathbf{t}}_H$ for all $H \in E^K$, and $\overline{\mathbf{y}}_i = \sum_{H \in E_i^K} \overline{\mathbf{t}}_H$ for all $i \in E$.

While Algorithm 2 actually solves F_2', the problem is in fact NP-Hard for $K \geq 3$. Consequently, it is unlikely that one can come up with an efficient algorithm to solve step 2, in particular.

Proposition 2.7. *Problem F_2' is NP-Hard for $K \geq 3$*

Proof. Deciding whether a $(K+1)$-uniform hypergraph has a spanning tree is NP-Complete for $K \geq 3$. For $K = 2$, while the decision problem can be solved in polynomial time, the minimization problem is still an open problem [43]. The idea of the proof is to reduce the decision problem to F_2', defined over forests of size K. The detailed proof is presented in Section B.1 of the Appendix. □

Given the complexity of solving F_2', we study two possible alternative approaches to derive lower bounds from relaxations of that formulation.

2.1.2.1 First Approach - Selecting Edge-disjoint K-Forests

Consider the subtour elimination constraints (SECs) (2.2). For integer solutions of F_2, these constraints are already implied by other constraints in the formulation. To check that, observe that

$$y_{ij} = y_{ji} \leq x_j, \qquad i, j \in E,$$

and as $\mathbf{y}_i$ defines a spanning tree, $\mathbf{x}$ also defines a spanning tree. Moreover, by further relaxing and dualizing (2.2) in F_2', the resulting problem consists of selecting $(n-1)/K$ non-overlapping K-forests, i.e., their union do not need to be acyclic, while also selecting independent interaction trees for each of them. This problem is easy to solve for a particular value of K. Such a relaxation is studied in what follows.

Let $\mu = (\mu_S)_{S \subset V, |S| \geq 2}$, be a set of non-negative multipliers associated to constraints (2.2). As before, let $\theta = (\theta_{ij})_{i<j \in E}$ be unconstrained multipliers associated to (2.20). After the two sets of constraints are relaxed and dualized, the following Lagrangian subproblem results:

$$F_2''(\theta, \mu) : L_2''(\theta, \mu) = C + \min \left\{ \sum_{i,j \in E} q'_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P_2'' \cap \mathbb{B}^{m+m^2+o+om} \right\},$$

where P_2'' is obtained by relaxing (2.20) and (2.2) in P_2, i.e., P_2'' is defined by (2.1), (2.3), (2.17)-(2.19). Lagrangian costs q'_{ij} are defined as $q'_{ij} = q_{ij} + \theta_{ij}$ if $i \neq j \in E$, $q'_{ii} = q_{ii} + \sum_{S \subset V, |S| \geq 2, i \in E(S)} \mu_S$, $i \in E$, and $C = -\sum_{S \subset V, |S| \geq 2} \mu_S (|S| - 1)$.

The associated Lagrangian dual is:

$$DF_2 : \qquad L_2''^{*} = \max \left\{ L_2''(\theta, \mu) : (\theta, \mu) \in \mathbb{R}^{\frac{m(m-1)}{2}} \times \mathbb{R}_+^{|\{S \subset V, |S| \geq 2\}|} \right\}.$$

Let $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ be an optimal solution for $F_2''(\overline{\theta}, \overline{\mu})$ for some particular choice of Lagrangian multipliers $(\overline{\theta}, \overline{\mu})$. If $\overline{s}_H = 1$, then $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ can be so that $\overline{\mathbf{t}}_H$ is the incidence vector of the spanning tree that minimizes (2.23). Also, after the relaxation of (2.2), variables $\mathbf{x}$ can be eliminated from the formulation. This way, $F_2''(\overline{\theta}, \overline{\mu})$ can be solved with the resolution of

$$\overline{q}_0 = C + \min \left\{ \sum_{H \in E^K} \overline{q}_H s_H : \sum_{H \in E_i^K} \mathbf{s}_H \leq 1, \forall i \in E; \sum_{H \in E^K} \mathbf{s}_H = \frac{n-1}{K}; \mathbf{s} \in \mathbb{B}^o \right\}, \quad (2.25)$$

and the appropriate adjustment of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{t}$. Problem (2.25) asks for the optimal selection of $(n-1)/K$ K-forests of G such that no edge i appears in more than one of them. In other words, (2.25) is a set packing problem with an additional cardinality constraint. This problem can be efficiently solved for $K = 2$ but is NP-Hard for $K \geq 3$.

Proposition 2.8. *Problem F_2'' is NP-Hard for $K \geq 3$.*

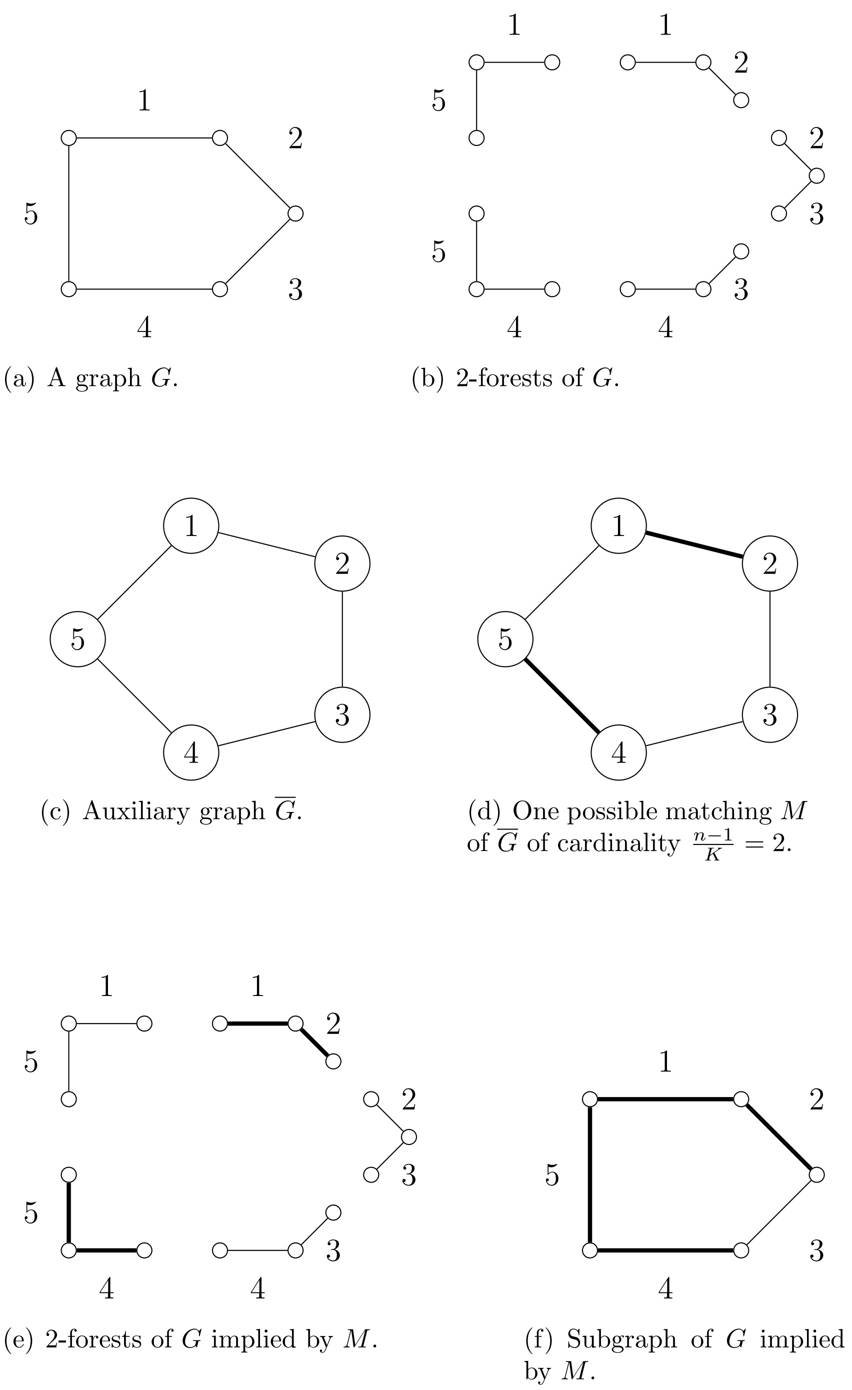
Proof. We provide a polynomial reduction from the problem of deciding whether a K-uniform hypergraph has a perfect matching. That problem is NP-Complete for $K \geq 3$. Details of the reduction are given in Section B.2 of the Appendix. $\square$

If $K = 2$, problem (2.25) requires the optimal selection of $(n-1)/2$ non-overlapping 2-forests. In other words, one has to find a minimum cost matching of cardinality $(n-1)/2$, in an auxiliary graph $\overline{G} = (\overline{V}, \overline{E})$, defined with vertex set $\overline{V} = E$ and edge set $\overline{E} = E^K$. An example is given in Figure 2.2. Finding a matching with fixed cardinality of a graph can be done in polynomial time, an algorithm is given in [41]. That algorithm is used as the basis of Algorithm 3 below, for the resolution of (2.25).

Algorithm 3:
Input: QMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. Set of costs $\overline{q}_H$ for each $H \in E^K$, $K = 2$.
Output: Solution $(\overline{\mathbf{x}}, \overline{\mathbf{s}})$ for (2.25).

```
1. Define an auxiliary graph $\overline{G} = (\overline{V}, \overline{E})$, where $\overline{V} = E$ and $\overline{E} = E^K$.
2. Let $U$ be a set of $m - (n-1)$ auxiliary vertices. Set $\overline{V} = \overline{V} \cup U$.
3. Let $J = \{\{u, v\} : u \in U, v \in \overline{V}\}$ and $\overline{q}_H = 0$ for all $H \in J$. Set $\overline{E} = \overline{E} \cup J$.
4. Find a minimum cost perfect matching of $\overline{G}$, let $\overline{\mathbf{s}}$ be its incidence vector. For
   all $i \in E$, if there is an $H \in E_i^K$ such that $s_H = 1$, set $\overline{x}_i = 1$. Otherwise, set
   $\overline{x}_i = 0$.
```

By assumption, G is connected and $n-1$ is even. Thus, there is a set of $(n-1)/2$ non-overlapping 2-forests, e.g., the decomposition of a spanning tree into 2-forests. A set of $(n-1)/2$ non-overlapping 2-forests of G implies a matching with $(n-1)/2$ edges ($n-1$ vertices) for the subgraph of $\overline{G}$ induced by $\overline{V} \setminus U$. The remaining $m-(n-1)$ vertices of $\overline{V} \setminus U$ can be matched to the vertices of U. Conversely, in any perfect matching of $\overline{G}$, the $m-(n-1)$ vertices of U can only be connected to $m-(n-1)$ vertices of $\overline{V}$. The remaining $n-1$ vertices of $\overline{V} \setminus U$ are matched to each other, resulting in a matching of $(n-1)/2$ edges or $(n-1)/2$ non-overlapping 2-forests for G. Moreover, since the edges in J have no cost, a perfect matching of $\overline{G}$ and a set of $(n-1)/2$ non-overlapping 2-forests of G have the same cost. Thus, Algorithm 3 indeed solves (2.25).



(a) A graph G.

(b) 2-forests of G.

(c) Auxiliary graph $\overline{G}$.

(d) One possible matching M of $\overline{G}$ of cardinality $\frac{n-1}{K} = 2$.

(e) 2-forests of G implied by M.

(f) Subgraph of G implied by M.

Figure 2.2. Using an auxiliary graph $\overline{G}$ in order to find a set of $(n-1)/K$ non-overlapping K-forests of a graph G, for $K = 2$.

In order to solve $F_2''(\overline{\theta},\overline{\mu})$ for $K = 2$, we can proceed by computing the costs (2.23), followed by the resolution of (2.25) by Algorithm 3. Algorithm 4 summarizes the main steps.

Algorithm 4:
Input: QMSTP instance given by $G = (V,E)$ and $Q \in \mathbb{R}_+^{m^2}$. Set of Lagrangian multipliers $(\overline{\theta},\overline{\mu}) \in \mathbb{R}^{\frac{m(m-1)}{2}} \times \mathbb{R}_+^{|\{S\subset V,|S|\geq 2\}|}$.
Output: Solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ for $F_2''(\overline{\theta},\overline{\mu})$ for $K = 2$.

1. Solve problem (2.23) for each $H \in E^K$ and let $\widetilde{\mathbf{t}}_H$ be the minimizing vector.
2. Solve (2.25) by Algorithm 3. Let $(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}})$ denote a solution.
3. Obtain a solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ of cost $L_2''(\overline{\theta},\overline{\mu}) = \overline{q}_0$ for $F_2''(\overline{\theta},\overline{\mu})$ by letting $\overline{\mathbf{s}} = \widetilde{\mathbf{s}}$ and $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$. Let $\overline{\mathbf{t}}_H = \overline{s}_H\widetilde{\mathbf{t}}_H$ for all $H \in E^K$, and $\overline{\mathbf{y}}_i = \sum_{H\in E_i^K} \overline{\mathbf{t}}_i$ for all $i \in E$.

The first step of Algorithm 4 can be implemented to run in $O(om\log n) = O(m^3\log n)$ time complexity. Algorithm 3 can be implemented to run in $O(|\overline{V}|^2|\overline{E}|) = O(m^4)$ [21]. This gives the complexity of step 2, which determines the overall worst case time complexity of Algorithm 4.

As a result of the discussion above, the solutions for the Lagrangian subproblem $F_2''(\theta,\mu)$ implicitly satisfy all valid inequalities for the matching polytope. Since the blossom inequalities [15] (facet defining inequalities for the matching polytope) are missing from F_2'', the Lagrangian dual bound provided by DF_2 might well be stronger than $Z(F_2)$.

Proposition 2.9. *$L_2''^* \geq Z(F_2)$.* □

The evaluation of $L_2''^*$ requires finding optimal multipliers for an exponential number of constraints (2.2). One of the known algorithmic alternatives to deal with exponentially many inequalities candidates to Lagrangian dualization is the relax-and-cut approach [30]. Due to the already excessive (though polynomial in n,m) number of other dualized constraints, the benefits of implementing a relax-and-cut algorithm for the evaluation of $L_2''^*$ are quite small: in practice, small lower bound improvements are obtained at a substantial increase of CPU time. Thus, we consider an algorithm for computing QMSTP lower bounds based on DF_2 where $\mu = \mathbf{0}$, θ is adjusted by the subgradient method, and each subproblem is solved by means of Algorithm 4. This algorithm is denoted Lag_2. A BB algorithm based on Lag_2 is presented in Section 2.2. We report computational results for Lag_2 in Section 2.3.

2.1.2.2 Second Approach - Variable Splitting

In this section, we present a reformulation of F_2 that preserves its LP relaxation bounds. The interesting aspect of this reformulation is that a Lagrangian relaxation scheme, with easy to solve subproblems, can be developed to evaluate its LP lower bounds. On the downside, the Lagrangian relaxation scheme needs to deal with a very large number of dualized constraints.

Consider the replacement of each variable s_H, for all $H \in E^K$, by K new binary variables s_{iH}, one for each $i \in H$, so that now we have $\mathbf{s} = (s_{iH})_{i \in E, H \in E_i^K}$. We denote by $\mathbf{s}_i = (s_{iH})_{H \in E_i^K}$ the row of $\mathbf{s}$ indexed by $i \in E$. Likewise, consider the replacement of each vector $\mathbf{t}_H$ by K new binary vectors $\mathbf{t}_{iH}$, one for each $i \in H$. Then, $\mathbf{t} = (t_{iHj})_{i,j \in E, H \in E_i^K}$ and we denote by $\mathbf{t}_i = (\mathbf{t}_{iH})_{H \in E_i^K}$ the entries of $\mathbf{t}$ with first index $i \in E$, and by $\mathbf{t}_{iH} = (t_{iHj})_{j \in E}$ the row of $\mathbf{t}_i$ indexed by $H \in E^K$. QMSTP can be formulated as:

$$F_3 : \qquad \min \left\{ \sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P_3 \cap \mathbb{B}^{m+m^2+Ko+Kom} \right\}, \tag{2.26}$$

where P_3 denotes the polytope given by:

$$\mathbf{x} \in X, \tag{2.27}$$

$$x_i = \sum_{H \in E_i^K} s_{iH}, \qquad i \in E, \tag{2.28}$$

$$\mathbf{y}_i = \sum_{H \in E_i^K} \mathbf{t}_{iH}, \qquad i \in E, \tag{2.29}$$

$$\mathbf{t}_{iH} \in (X_H \times s_{iH}), \qquad i \in E, H \in E_i^K, \tag{2.30}$$

$$y_{ij} = y_{ji}, \qquad i < j \in E, \tag{2.31}$$

$$s_{iH} = s_{jH}, \qquad i < j \in E, H \in E_i^K \cap E_j^K. \tag{2.32}$$

Later, we will show that the relaxation of constraints (2.31) and (2.32) results in a problem that is easy to solve for any factor K, what allows us to develop a tractable lower bounding procedure based on F_3. Before discussing that, observe that constraints of type (2.32) were not imposed for $\mathbf{t}$, what implies that $Z(F_3)$ may be weaker than $Z(F_2)$. This can be overcome by conveniently rewriting the objective function of F_3. Notice that if $t_{iHj} = 1$, for some $i, j \in E$ and $H \in E_i^K$, then because of (2.30) and

(2.32),

$$\sum_{k\in H} t_{kHj} = K. \tag{2.33}$$

Using (2.29) and (2.33), the objective function in (2.26) can be rewritten as:

$$\sum_{i,j\in E} q_{ij}y_{ij} = \sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E} q_{ij}t_{iHj} = \sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E}\sum_{k\in H}\frac{1}{K}q_{ij}t_{kHj}t_{iHj}.$$

Note that $t_{iHj}t_{kHj} = t_{iHj} = t_{kHj}$ for any $i, j, k \in E$ and $H \in E_i^K \cap E_k^K$. Therefore

$$\begin{aligned}\sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E}\sum_{k\in H}\frac{1}{K}q_{ij}t_{kHj}t_{iHj} &= \sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E}\sum_{k\in H}\frac{1}{K}q_{ij}t_{kHj}\\ &= \sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E}\sum_{k\in H}\frac{1}{K}q_{kj}t_{iHj} \qquad (2.34)\end{aligned}$$

In other words, (2.34) states that the cost of the tree defined by $\mathbf{t}_{iH}$, for some $i \in E$ and $H \in E_i^K$, depends equally on all the edges in H and their interaction costs. In the remainder of this section, we assume that the objective in (2.26) is rewritten according to (2.34). Bearing that in mind, we have the following result.

Proposition 2.10. *$Z(F_2) = Z(F_3)$.*

Proof. We make use of an argument based on the application of Lagrangian relaxation to the LP relaxations of F_2 and F_3. We dualize constraints (2.31) with unconstrained Lagrangian multipliers $\theta = (\theta_{ij})_{i<j\in E}$, defined as before. This gives the the Lagrangian subproblem

$$F_3'(\theta) : L_3'(\theta) = \min\left\{\sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in E}\sum_{k\in H}\frac{1}{K}q'_{kj}t_{iHj} : (\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{t}) \in P_3' \cap \mathbb{B}^{m+m^2+Ko+Kom}\right\},$$

where P_3' is obtained by relaxing (2.31) in P_3, i.e., P_3' is defined by (2.27)-(2.30) and (2.32). Lagrangian modified costs are defined as $q'_{ij} = q_{ij} + \theta_{ij}$ for $i \neq j \in E$ and $q'_{ii} = q_{ii}$ for $i \in E$.

We now show that for any $\overline{\theta} \in \mathbb{R}^{\frac{m(m-1)}{2}}$, $Z(F_3'(\overline{\theta})) = Z(F_2'(\overline{\theta}))$, which proves the claim.

Given a feasible solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ for the LP relaxation of $F_2'(\overline{\theta})$, letting $\widetilde{\mathbf{x}} = \overline{\mathbf{x}}$, $\widetilde{\mathbf{y}} = \overline{\mathbf{y}}$, $\widetilde{s}_{iH} = \overline{s}_H$ and $\widetilde{\mathbf{t}}_{iH} = \overline{\mathbf{t}}_H$ for all $i \in H$ and $H \in E_i^K$, we obtain a feasible solution $(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$, with the same objective value for the LP relaxation of $F_3'(\overline{\theta})$.

Conversely, by (2.34), given a solution for the linear relaxation of $F_3'(\overline{\theta})$, there is always a feasible solution $(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$ with the same objective value such that $\widetilde{\mathbf{t}}_{iH} = \widetilde{\mathbf{t}}_{jH}$

for any $i, j \in E$ and $H \in E_i^K \cap E_j^K$. Letting $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$, $\overline{\mathbf{y}} = \widetilde{\mathbf{y}}$, $\overline{s}_H = \widetilde{s}_{iH}$, and $\overline{\mathbf{t}}_H = \widetilde{\mathbf{t}}_{iH}$, for all $H \in E^K$ and some $i \in H$, we obtain a feasible solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ with the same objective value for the LP relaxation of $F_2'(\overline{\theta})$. $\square$

When constraints (2.31) and (2.32) are relaxed in F_3, we obtain a problem that is easy to solve for any value of K. Consider again unconstrained dual multipliers $\theta = (\theta_{ij})_{i<j\in E}$ attached to (2.31). Define unconstrained multipliers $\pi = (\pi_{ijH})_{i<j\in E, H\in E_i^K\cap E_j^K}$, attached to (2.32). Assume $\pi_{iiH} = 0$ for all $i \in E$, and $\pi_{ijH} = -\pi_{jiH}$, for all pairs $i > j \in E$ and $H \in E_i^K \cap E_j^K$. Then, we obtain the Lagrangian subproblem:

$$F_3''(\theta,\pi): \quad L_3''(\theta,\pi) = \min\left\{\sum_{i\in E}\sum_{H\in E_i^K}\sum_{j\in H}(\pi_{ijH}s_{iH} + \sum_{k\in E}\frac{1}{K}q'_{jk}t_{iHk}) \right.$$
$$\left. : (\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{t}) \in P_3'' \cap \mathbb{B}^{m+m^2+Ko+Kom}\right\},$$

where P_3'' is obtained by relaxing (2.31) and (2.32) in P_3, i.e., P_3'' is defined by (2.27)-(2.30). The modified costs are defined as $q'_{ij} = q_{ij} + \theta_{ij}$ for $i \neq j \in E$ and $q'_{ii} = q_{ii}$ for $i \in E$. The associated Lagrangian dual is:

$$DF_3: \qquad L_3''^{*} = \max\left\{L_3''(\theta,\pi) : (\theta,\pi) \in \mathbb{R}^{\frac{m(m-1)}{2}+\frac{oK(K-1)}{2}}\right\}.$$

To see how $F_3''(\overline{\theta},\overline{\pi})$ can be solved for any choice of multipliers $(\overline{\theta},\overline{\pi}) \in \mathbb{R}^{\frac{m(m-1)}{2}+\frac{oK(K-1)}{2}}$, consider an optimal solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$. We see that if $\overline{x}_i = 1$ for some $i \in E$, and $\overline{s}_{iH} = 1$ for some $H \in E_i^K$, then $\overline{\mathbf{t}}_{iH}$ is the incidence vector of the spanning tree that minimizes

$$\overline{q}_{iH} = \frac{1}{K}\min\left\{\sum_{j\in H}\sum_{k\in E} q'_{jk}t_{iHk} : \mathbf{t}_{iH} \in X_H \cap \mathbb{B}^m\right\}. \tag{2.35}$$

Therefore, if $\overline{x}_i = 1$, $i \in E$, we have $\overline{s}_{iH} = 1$ for the element of H_i^K that minimizes

$$\overline{q}_i = \min\left\{\overline{q}_{iH} + \sum_{j\in H}\pi_{ijH} : H \in E_i^K\right\}. \tag{2.36}$$

Thus, F_3'' can be solved by solving

$$\overline{q}_0 = \min\left\{\sum_{i\in E}\overline{q}_i x_i : x \in X \cap \mathbb{B}^m\right\}, \tag{2.37}$$

followed by the appropriate adjustment of $\mathbf{y}$, $\mathbf{s}$, and $\mathbf{t}$. The following algorithm summarizes the procedure.

Algorithm 5:
Input: QMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. Set of Lagrangian multipliers $(\overline{\theta}, \overline{\pi}) \in \mathbb{R}^{\frac{m(m-1)}{2} + \frac{oK(K-1)}{2}}$.
Output: Solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ for $F_3''(\overline{\theta}, \overline{\pi})$.

1. Solve (2.35) and obtain $\overline{q}_{iH}$ for each $i \in E$ and $H \in E_i^K$. Denote the minimizing vector by $\widetilde{\mathbf{t}}_{iH}$. Observe that (2.35) needs to be solved only once for each $H \in E^K$, i.e., find $\overline{q}_{iH}$ for some $i \in H$ and let $\overline{q}_{jH} = \overline{q}_{iH}$ and $\widetilde{\mathbf{t}}_{jH} = \widetilde{\mathbf{t}}_{iH}$ for all $j \neq i \in H$.
2. Solve (2.36) for each $i \in E$ to obtain $\overline{q}_i$. Let $\widetilde{s}_{iH} = 1$ for the minimizing element $H \in E_i^K$ and $\widetilde{s}_{iI} = 0$ for the remaining $I \neq H \in E_i^K$.
3. Solve the minimum spanning tree problem in (2.37) and denote the minimizing vector by $\widetilde{\mathbf{x}}$.
4. Obtain a solution vector $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ of cost $\overline{q}_0 = L_3''(\overline{\theta}, \overline{\pi})$ for $F_3''(\overline{\theta}, \overline{\pi})$ by letting $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$, $\overline{s}_{iH} = \widetilde{x}_i \widetilde{s}_{iH}$ and $\overline{\mathbf{t}}_{iH} = \overline{s}_{iH} \widetilde{\mathbf{t}}_{iH}$ for all $i \in E$ and $H \in E_i^K$, and $\overline{\mathbf{y}}_i = \sum_{H \in E_i^K} \mathbf{t}_{iH}$ for all $i \in E$.

Steps 1 and 2 can be implemented to run in $O(om \log n) = O(m^{K+1} \log n)$ and $O(Ko) = O(Km^K)$ time, respectively. An $O(m \log n)$ implementation can be given for Step 3. Thus, the overall time complexity of the algorithm is defined by Step 1.

In order to evaluate the strength of DF_3, we first present the following result.

Proposition 2.11. *P_3'' is an integral polytope.*

Proof. The proof is quite similar to the proof of Proposition 2.4 and is presented in Section B.3 of the Appendix. □

By Propositions 2.10 and 2.11, we obtain

Corollary 2.3. $L_3''^* = Z(F_2) = Z(F_3)$. □

To solve DF_3, we have to deal with the large number of dualized constraints, $O(Km^K)$ in general. We faced convergence difficulties when applying subgradient optimization to this relaxation. For this reason, we devised the following heuristic to adjust the multipliers assigned to (2.32).

First, let us we clarify the reasoning behind the heuristic. Given $i \in E$ and a certain $\overline{H}$ that minimizes (2.36), observe that for any $H \neq \overline{H} \in E_i^K$, $\overline{q}_{iH}+\sum_{j\in H}\pi_{ijH} \geq \overline{q}_{i\overline{H}}+\sum_{j\in\overline{H}}\pi_{ij\overline{H}}$. Notice that there is a margin for the decrease of π_{ikH}, for some $k \in H$, without any change in $\overline{q}_i$. This decrease, and consequently the increase of π_{kiH}, can cause the increase of $\overline{q}_k$, in case H is the minimizing element of (2.36) for k. Note that $\sum_{j\in H}\pi_{ijH}$ can be decreased by at most $\lambda = \overline{q}_{iH} + \sum_{j\in H}\pi_{ijH} - (\overline{q}_{i\overline{H}} + \sum_{j\in\overline{H}}\pi_{ij\overline{H}})$, before $\overline{q}_i$ changes. These ideas are employed in the algorithm below. Given multipliers $\overline{\theta}$, the algorithm will implicitly find multipliers $\overline{\pi}$ and a solution for $F_3''(\overline{\theta},\overline{\pi})$ such that $L_3''(\overline{\theta},\overline{\pi}) \geq L_3''(\overline{\theta},\mathbf{0})$.

Algorithm 6:
Input: QMSTP instance given by $G = (V,E)$ and $Q \in \mathbb{R}_+^{m^2}$. Set of Lagrangian multipliers $\overline{\theta} \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: Solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ for $F_3''(\overline{\theta},\overline{\pi})$ for an implicit set of multipliers $\overline{\pi} \in \mathbb{R}^{\frac{oK(K-1)}{2}}$ such that $L_3''(\overline{\theta},\overline{\pi}) \geq L_3''(\overline{\theta},\mathbf{0})$.

1. Let $\overline{q}_i = \infty$ for all $i \in E$.
2. For each $H \in E^K$:
   a) Let $\widetilde{\mathbf{t}}_H$ be the minimizing vector of
   $$\overline{q}_H = \min\{\sum_{j\in H}\sum_{k\in E} q'_{jk}t_{Hk} : \mathbf{t}_H \in X_H \cap \mathbb{B}^m\}$$
   b) Consider $\lambda_i = \overline{q}_H/K$ for all $i \in H$.
   c) Assume an ordering $(e_1,\ldots,e_K)$ for the elements of H and for i going from 1 to K do the following.
   If $\overline{q}_H < \overline{q}_{e_i}$, let $\overline{q}_{e_i} = \overline{q}_H$, $\widetilde{s}_{e_iH} = 1$ and $\widetilde{s}_{e_iI} = 0$ for $I \neq H \in E_i^K$.
   If $\overline{q}_H > \overline{q}_{e_i}$, let $\lambda_{e_j} = \lambda_{e_j} + (\overline{q}_H - q_{e_i})/(K-i)$ for $i < j \leq K$.
3. Solve the minimum spanning tree problem (2.37) and denote by $\widetilde{\mathbf{x}}$ the minimizing vector.
4. Obtain the solution vector $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ by letting $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$, $\overline{s}_{iH} = \widetilde{x}_i\widetilde{s}_{iH}$ and $\overline{\mathbf{t}}_{iH} = \overline{s}_{iH}\widetilde{\mathbf{t}}_H$ for all $i \in E$ and $H \in E_i^K$, and $\overline{\mathbf{y}}_i = \sum_{H\in E_i^K}\mathbf{t}_{iH}$ for all $i \in E$.

The second step of Algorithm 6 can be implemented to run in $O(om\log n) =$

$O(m^{K+1} \log n)$ time complexity and defines the overall complexity of the algorithm.

We denote by Lag_3 the QMSTP lower bounding algorithm based on DF_3 that optimizes θ by means of subgradient optimization and uses Algorithm 6 to simultaneously guess good multipliers π and solve Lagrangian subproblems.

In Section 2.3, we conduct computational experiments to compare the bounds given by Lag_1, Lag_2, and Lag_3. Before that, in the next section, we describe how two BB algorithms based on Lag_1 and Lag_2 were implemented. As we will demonstrate in Section 2.3, Lag_3 is computationally expensive for $K \geq 3$ and the evaluation of the implied Lagrangian bounds suffers very badly from convergence problems. For $K = 2$, a preliminary BB algorithm that makes use of Lag_3 was largely outperformed by the one based on Lag_2. As such, we decided not to proceed with the implementation of a BB algorithm based on Lag_3.

2.2 Branch-and-Bound Algorithms

In this section, we describe the main implementation details of two BB algorithms, BB_1 and BB_2, respectively based on the Lagrangian relaxation lower bounding procedures Lag_1 and Lag_2. BB_1 and BB_2 are quite similar, differing only when explicitly mentioned in the exposition that follows.

2.2.1 Initial Upper Bounds

Initial valid QMSTP upper bounds are obtained by means of the following multi-start heuristic. We first randomly select an initial spanning tree T of G and then apply local search. The latter is implemented by evaluating the trees obtained by inserting into T an edge i not in T and removing the edges in the resulting cycle. If the best removal for edge i results in a tree with better cost than T, that tree immediately becomes the current solution. The process stops when the insertion of every edge not in T is evaluated and no cost improvement is detected. The procedure is repeated 100 times, each one starting with a different initial spanning tree.

2.2.2 Lower Bounds and Node Selection

Algorithm BB_1 makes use of Lag_1 as its bounding procedure, while algorithm BB_2 employs Lag_2. In an attempt to accelerate the resolution of the problem at each non-root BB node, Lagrangian multipliers at a given node are initialized with the best multipliers found for the parent node. This way, we expect to obtain near optimal multipliers with a smaller number of steps of the subgradient method. As a drawback, a table of size $O(m^2)$ needs to be stored at each node. For this reason, a best bound search strategy becomes prohibitive, since a huge amount of memory is needed to store the node list. Consequently, both algorithms implement a depth-first search.

2.2.3 Branching and Variable Selection

Assume that $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are respectively the $\mathbf{x}$ and $\mathbf{y}$ components of the solution vector that attains the best value for the Lagrangian subproblem at a given BB node.

if $\overline{y}_{ij} = \overline{y}_{ji}$ for all $i < j \in E$, the subproblem given by the current node has been solved to optimality. Otherwise, there is either an edge i such that $\overline{x}_i = 1$ and $\overline{y}_{ji} = 0$ for an edge j with $\overline{x}_j = 1$ (Edge i was selected but was not used in some interaction tree) or an edge i such that $\overline{x}_i = 0$ and $\overline{y}_{ji} = 1$ for an edge j with $\overline{x}_j = 1$. (Edge i was not selected but was used in some interaction tree.) Any edge in one of these situations is candidate for branching. Once the branching edge is determined, two new nodes are created. For one of them, we force the edge to be selected. For the other, the edge is forbidden.

The way branching constraints are enforced is one of the few differences between BB_1 and BB_2. For BB_1, it suffices to force (resp. to prevent) the appearance of the edge in the trees obtained in steps 1 and 2 of Algorithm 1. For BB_2, if an edge i is imposed (resp. forbidden) it is necessary to grant that in Algorithm 3 exactly one (resp. none) of the 2-forests containing i is (resp. are) selected. To guarantee such a condition, we remove from the auxiliary graph $\overline{G}$ any edge that connects i to a vertex of U (resp. $\overline{V}$).

In order to select the branching variable, we do the following. For each candidate branching variable, we compute the Gilmore-Lawler bounds for the cases in which the variable is imposed and forbidden. If any of these two bounds is larger than the best known upper bound, the variable is then fixed accordingly. If any variable is fixed in this step, we apply another round of subgradient optimization, starting with the best multipliers obtained during the previous round. Otherwise, we select the branching variable according to a strong branching rule [1], that means that the branching variable is the one for which the minimum between the two values is maximum.

2.2.4 Redistribution of the Costs of Fixed Variables

Assume that at a given BB node, a nonempty set F of edges known to be included in the (integer) solution for that node. The objective function can be written as $\sum_{i\in E\setminus F}\sum_{j\in E} q_{ij}x_ix_j + \sum_{i\in F}\sum_{j\in E} q_{ij}x_ix_j$. Multiplying the second term by $\sum_{i\in E} x_i/(n-1) = 1$, using $x_i = 1$ for $i \in F$ and using the linearization variables, we obtain the following expression for the objective function:

$$\sum_{i\in E\setminus F}\sum_{j\in E}(q_{ij} + \frac{\sum_{k\in F} q_{kj}}{n-1})y_{ij} + \sum_{i\in F}\sum_{k\in E}\frac{\sum_{j\in F} q_{jk}}{n-1}y_{ij}.$$

For the optimal Lagrangian multipliers, the bounds are unaffected by the change indicated above. However, since multipliers are not necessarily optimized exactly by the subgradient method and convergence is often an issue when a large number of multipliers is dualized, this reformulation might help to obtain good lower bounds earlier in the process.

In order to give some intuition for the above expression, consider as an example the Lagrangian relaxation scheme Lag_1. During the resolution of the Lagrangian subproblems, we compute the best interaction tree for each edge $i \in E$. With the reformulation above, if $i \notin F$, the cost of selecting $j \in E$ for its interaction tree becomes $q_{ij} + \sum_{k\in F} q_{kj}/(n-1)$. That means that it is necessary to take into account a fraction of the interaction costs of the fixed edges and j, leading to a better estimate of the actual cost of using i in a solution. On the other hand, all $i \in F$ will have the same interaction tree.

2.2.5 Parallelization

Our computational experiments are conducted in a multi-processor shared memory system. In order to take advantage of this hardware, our BB implementations use parallel programming techniques, following the guidelines proposed in [32] for a parallel QAP BB algorithm. We give a brief description of their strategy and show how we improved that implementation, by introducing an effective load balancing mechanism.

As in [32], BB nodes are kept in disjoint lists: a global list and a local list for each processor. Each processor explores its local list independently. Whenever a processor detects that its local list is empty, it requires access to the global list. After obtaining access to the global list, the processor explores that list until a node at level d or greater is found. Then, it adds this node to its local list, releases the access to the global list, and goes back to exploring its own list.

In [32], a processor stops after it observes that the global list is empty. This might lower parallel efficiency, since after that moment that processor no longer works. In order to overcome that, we proceed in a different way. After detecting that the global list is empty, the processor waits, periodically checking the global list for work.

On the other hand, a processor that is working, periodically checks whether there are processors waiting for work. In positive case, that processor removes some nodes from the head of its local list and add them to the global list. Those nodes will be available to the waiting processors and parallel efficiency should improve. Under this policy, a processor only stops when its local list is empty and all the other processors are waiting.

2.3 Computational Experiments

In this section, we conduct computational experiments in order to evaluate the quality and the practical performance of our solution techniques, compared to other approaches in the literature. Let us first describe the test instances we employed.

2.3.1 Test Instances

The algorithms were tested with two sets of instances from the literature. We denote by CP the first set, introduced by Cordone and Passeri [13]. These instances comprise graphs with $n \in \{10, 15, \ldots, 50\}$ and densities $d \in \{33\%, 67\%, 100\%\}$. Depending on how the diagonal (q_{ii}) and the off-diagonal (q_{ij}) entries of Q are defined, four types of instances were generated, for each tuple (n, d). These types are denoted CP1, CP2, CP3, and CP4. For CP1, values for q_{ii} and q_{ij} correspond to integers randomly chosen from $\{1, \ldots, 10\}$, with uniform probability. Similarly, for CP2, $q_{ii} \in \{1, \ldots, 10\}$ and $q_{ij} \in \{1, \ldots, 100\}$. For CP3, $q_{ii} \in \{1, \ldots, 100\}$ and $q_{ij} \in \{1, \ldots, 10\}$. Finally, for CP4, $q_{ii}, q_{ij} \in \{1, \ldots, 100\}$.

The second set, denoted by OP, was introduced by Öncan and Punnen [35]. These instances comprise complete graphs of different sizes $n \in \{6, 7, \ldots, 18, 20, 30, 40, 50\}$. For each n, ten instances of three different types, OP1, OP2, and OP3 were generated. Instances of type OP1 have integer costs q_{ii} and q_{ij} randomly chosen from $\{1, \ldots, 100\}$ and $\{1, \ldots, 20\}$, respectively. For OP2, an integer weight w_v randomly chosen from $\{1, \ldots, 10\}$ is assigned to each vertex $v \in V$. Given two different edges $i = \{a, b\}$ and $j = \{c, d\}$, $q_{ij} = w_a w_b w_c w_d$. For an edge i, q_{ii} is an integer randomly chosen from $\{1, \ldots, 10000\}$. For OP3, each vertex represents a 2-dimensional point with coordinates randomly chosen in the interval $[0, 100]$. The value of q_{ii} is given by the euclidean distance between the extremities of i, while q_{ij}, $i \neq j$, is the distance between the midpoints of i and j.

2.3.2 Computational Results

Computational experiments were performed on a machine with the following configuration: Two Intel Xeon processors, each one with six cores running at 2.4GHz, a total of 32GB of shared RAM memory, and Ubuntu 12.04 operating system. All algorithms were coded in C++ and compiled with G++ 4.6.3, optimization flag O3 turned on. OpenMP was used to implement the parallel BB algorithms.

Our subgradient algorithm implementation has the following particular settings. At the root node, 5000 iterations are performed, with an initial step size of 2 in the direction of the normalized subgradient. The step size is halved whenever 500 iterations have past with no improvement in the Lagrangian dual function. At non-root nodes, 100 iterations are performed with the same initial step size of 2. The step size is halved at every 10 iterations without improvements.

In Table 2.1, we report lower bounds for the formulations of Assad and Xu [7] (F_{AX92}), Öncan and Punnen [35] (F_{OP10}), F_1, F_2 for $K = 2$, and F_3 for $K \in \{2, 3, 4\}$. The bounds we indicate in columns under headings F_1 and F_2 are $L_1'^*$ and $L_2''^*$, respectively. These bounds were respectively approximated with the Lagrangian relaxation schemes Lag_1 and Lag_2. Likewise, columns under headings F_3 depict an approximation of the bound $L_3''^*$ provided by Lag_3, for values of $K \in \{2, 3, 4\}$.

The lower bounds we report for F_{AX92} and F_{OP10} were evaluated by ourselves, by means of Lagrangian relaxation algorithms implemented as described in those references. We found differences between the bounds we evaluated and those reported by Öncan and Punnen [35]. In order to further validate the correctness of the Lagrangian bounds provided by Lag_1, we computed the LP relaxation bounds $Z(F_1)$, by means of a LP based cutting plane algorithm. The computational results indicate that the bounds provided by Lag_1 are close to $Z(F_1)$, but never exceed them. However, the bounds reported for F_{OP10} in [35] quite often exceed $Z(F_1)$. We provide an in depth discussion of this matter in section C of the Appendix.

The first four columns of Table 2.1 give the number of vertices (n), the number of edges (m), the type (type), and the best known upper bound (ub), for each instance. Subsequent columns provide the lower bound (lb) and the computational time (t) (in seconds) taken by each formulation/lower bounding procedure. A time limit of 10 hours was imposed. The best overall lower bounds are indicated in boldface.

In Table 2.2, we compare algorithms BB_1, BB_2, and BB_{CP}, the BB algorithm in [13]. The computational results we report for BB_{CP} are those provided in [12]. For BB_1 and BB_2, a time limit of 100 hours was imposed. The stopping criteria for BB_{CP}, however, was not the same for all instances. For some of them, the algorithm was stopped after a time limit of 3600 seconds was reached. For others, after 10^6 nodes were investigated. Differently from Table 2.1, where each row refers to a particular instance, Table 2.2 presents results aggregated for ranges of n and m. Detailed computational results for each instance in our test bed are provided in section D of the Appendix.

The first three columns of Table 2.2 present the range of n and m, and the type of instances being considered. Next, for each algorithm, we present the total number of instances solved to optimality (solv.), the maximum number of nodes investigated (max nodes) and the maximum time (max t) in seconds needed by the algorithm to solve a single instance in the range (considering only those instances solved to optimality). An entry "-" indicates that all instances in the range were left unsolved by the algorithm under consideration.

From both tables, it is clear that the bounds implied by F_1 are much stronger than the previous bounds in the literature (16,6% stronger than the bounds of [7] and 26,7% stronger than the bounds of [35], for OP1 instances). Compared to the other schemes, the Lagrangian relaxation algorithm Lag_1 seems to offer a good trade-off between lower bound quality and computational effort. That claim is validated by how BB_1 and BB_2 do compare to each other.

Formulation F_2 (Lag_2) provides lower bounds that are significantly stronger than those provided by F_1 (Lag_1), F_{AX92}, and F_{OP10} (65% stronger than F_1 (Lag_1), 90,6% stronger than F_{AX92}, and 81,2% stronger than F_{OP10}, for CP instances). Consequently, the number of nodes investigated by BB_2 is orders of magnitude smaller than BB_1 and BB_{CP} counterparts. However, BB_2 is dominated by BB_1 in terms of computational time, due to the high costs demanded to run Lag_2.

Lower bounds given by F_3 (Lag_3) with $K = 2$ are quite close to those given by F_2 (Lag_2), but demand less computational effort. As expected, these bounds get stronger as K grows. Bounds provided by F_3 with $K = 4$ are the overall best, but require a high computational effort. Due to convergence difficulties and the fact that multipliers π are adjusted heuristically, Lag_3 actually provided a poor approximation for the true bound $L_3''^{*}$. That behavior can be observed, for example, for OP2 and OP3 instances with $n = 13$ vertices.

Compared to BB_1, BB_{CP} explores many more nodes. A fair comparison between BB_1 and BB_{CP} is not trivial to state, since they were tested in different computational environments and make use of different stopping criteria. In addition, the enumeration tree of BB_1 (and BB_2) was explored in parallel (parallel efficiencies around 80% were achieved) whereas that of BB_{CP} was not.

With BB_1, for the first time in the QMSTP literature, the following sets of instances were solved to proven optimality: all instances of Cordone and Passeri with

Instance				F_{AX}		F_{OP}		F_1		F_2 $(K=2)$		F_3 $(K=2)$		F_3 $(K=3)$		F_3 $(K=4)$	
n	m	type	ub	lb	t	lb	t	lb	t	lb	t	lb	t	lb	t	lb	t
25	100	CP1	2185	1115.5	0	1193.2	3	1285.1	2	1718.7	40	1715.8	20	1764	1592	**1877.3**	18998
25	100	CP2	19976	8170.6	0	8988.8	3	10061.1	2	14860.8	52	14829.5	20	15371.2	1595	**16617.2**	19846
25	100	CP3	2976	2069.1	0	1961.3	3	2289.2	2	2652.5	41	2645.9	20	2660.9	1592	**2749.9**	19174
25	100	CP4	21176	9296.8	0	10089.2	3	11190	2	15977	55	15947.7	20	16470.8	1601	**17736.8**	20125
25	200	CP1	2023	755.1	0	801.3	12	828	3	1316.7	305	1315	135	1431.6	21610	**1522.7**	36000
25	200	CP2	18251	4154.4	0	4564.5	12	5028.6	3	10497.4	362	10480	136	11780.2	21618	**12795.7**	36000
25	200	CP3	2546	1468.1	0	1385.3	12	1626.8	3	2071	261	2068.2	136	2135.2	21472	**2185**	36000
25	200	CP4	19207	5183.6	0	5560.5	12	6065.4	3	11522	360	11504.6	134	12798.5	21666	**13812.8**	36000
25	300	CP1	1943	668.5	0	705.2	23	715.4	6	1143.2	1012	1141.3	392	1268.7	36000	**1348.7**	36000
25	300	CP2	17411	2879.4	0	3161.8	24	3443.2	6	8533.5	1118	8518.3	394	9920.1	36000	**10868.2**	36000
25	300	CP3	2471	1279.2	0	1213.9	23	1405.6	6	1875.3	918	1871.8	400	**1957.4**	36000	1874.4	36000
25	300	CP4	18370	3865.2	0	4086.6	24	4451.3	6	9563.4	1135	9542.9	397	10947.8	36000	**11854.5**	36000
13	78	OP1	1022	513.7	0	475.9	2	606.3	0	842.6	21	838.9	14	847.5	655	**901**	9947
13	78	OP1	1089	592.9	0	532.1	2	702	0	900.1	22	891.5	14	905.5	648	**941.9**	9892
13	78	OP1	1163	609.8	0	576.3	2	697	0	945.2	22	941.5	14	952.1	654	**1003.8**	9763
13	78	OP1	1129	703.5	0	659	1	803.1	0	1033.3	21	1028.9	14	1041	640	**1086**	9763
13	78	OP1	1023	663.2	0	588.4	2	748.8	0	1001.4	21	997.3	14	1005.8	664	**1054.7**	9838
13	78	OP1	982	586.9	0	546.2	1	715.4	0	933	21	928.6	14	931	652	**970.8**	9714
13	78	OP1	1048	520.8	0	466	2	613.3	0	838.3	22	833.7	14	832	656	**880.4**	9866
13	78	OP1	1045	611.8	0	571.1	2	712.8	0	929.4	22	926.6	14	935.8	645	**976.2**	9724
13	78	OP1	1065	637.6	0	594.3	2	741.2	0	980.3	22	974	14	978.1	653	**1017.2**	9755
13	78	OP1	1160	618.2	0	572.1	2	720.3	0	978.6	21	976.6	14	991.4	648	**1050.7**	9575
13	78	OP2	45586	44885	0	44693	1	**45586**	0	**45586**	12	42642.2	4	38844.2	195	37228.7	2940
13	78	OP2	49313	48747.1	0	45717	1	**49313**	0	**49313**	11	45373.6	4	37705.3	183	35443.9	2825
13	78	OP2	44513	44257.5	0	43676.5	1	**44513**	0	**44513**	11	36545.3	4	34665.2	181	34181.8	2866
13	78	OP2	37250	37250	0	37054	1	**37250**	0	**37250**	11	31793.1	4	30504.9	183	25435.2	2811
13	78	OP2	50990	49908	0	46969	1	**50990**	0	**50990**	11	48493.8	4	45486.7	198	43695.1	2846
13	78	OP2	43261	42380	0	41140	1	**43261**	0	**43261**	12	33263	4	24401.7	181	25020	2797
13	78	OP2	36085	35809.1	0	35135	1	**36085**	0	**36085**	11	34055.6	4	32091.6	190	30169.3	2804
13	78	OP2	34474	34442.6	0	33775	1	**34474**	0	**34474**	10	30467.7	4	26480.4	180	24826.8	2829
13	78	OP2	28566	28360.2	0	27653	1	**28566**	0	**28566**	10	24213.1	4	22879.2	178	21686	2842
13	78	OP2	34847	34493	0	33909	1	**34847**	0	**34847**	13	32670.2	4	26926.7	187	27950.4	2902
13	78	OP3	1731	1595.6	0	1648.3	1	**1731**	0	**1731**	9	1720.2	8	1730.4	601	1730.5	9058
13	78	OP3	2484	2341.4	0	2318.7	1	**2484**	0	**2484**	10	2332.5	4	2210.2	218	2270.2	3524
13	78	OP3	2440	2228.8	0	2297.2	1	2436.6	0	**2440**	12	2407.4	9	2430.1	595	2426.5	5881
13	78	OP3	2489	2307.5	0	2272.5	1	2453.2	0	**2483**	23	2420.1	6	2453.9	591	2187	3398
13	78	OP3	2044	1932.8	0	1915	1	**2044**	0	**2044**	11	1940.9	4	2029.6	609	1910.1	3421
13	78	OP3	1806	1655.7	0	1634	1	1805	0	**1806**	11	1754.8	5	1692.4	268	1796.6	8852
13	78	OP3	2185	2041.9	0	2035	1	**2185**	0	**2185**	10	2162.6	5	2184.1	615	2167	4859
13	78	OP3	2275	2081	0	2134.2	1	2272.8	0	**2275**	11	2269.9	11	2265.8	597	2270.2	8853
13	78	OP3	1968	1741.5	0	1857.6	1	1943.1	0	**1957.7**	21	1931.5	13	1942.4	616	1948	7916
13	78	OP3	2331	2241.4	0	2252	1	**2331**	0	**2331**	10	2283.1	5	2097.8	226	2330.4	7964

Table 2.1. Lower bound comparisons.

Instance			BB_{CP}			BB_1			BB_2		
n	m	type	solv.	max nodes	max t	solv.	max nodes	max t	solv.	max nodes	max t
10-20	15-105	CP	28/28	51880837	887	28/28	144309	946	28/28	24106	3170
20	127	CP	0/4	-	-	4/4	24431331	271761	0/4	-	-
10-15	45-105	OP1	60/60	7922195	184	60/60	19239	174	60/60	4057	775
16-17	120-136	OP1	0/20	-	-	20/20	449565	6386	20/20	48463	20080
18	153	OP1	0/10	-	-	10/10	5351735	93178	0/10	-	-
10-20	45-190	OP2	100/100	583379	29	100/100	7	15	100/100	3	133
30	435	OP2	0/10	-	-	10/10	1	151	10/10	1	1900
50	1225	OP2	0/10	-	-	10/10	1	1731	0/10	-	-
10-20	45-190	OP3	98/100	979125	36	100/100	21	12	100/100	13	218
30	435	OP3	0/10	-	-	10/10	129	491	10/10	81	17857
50	1225	OP3	0/10	-	-	10/10	735	19045	0/10	-	-

Table 2.2. Comparison of branch-and-bound algorithms.

20 vertices and 127 edges, OP2 and OP3 instances of Öncan e Punnen with $n \in \{30, 50\}$ and all OP1 instances with $n = \{16, 17, 18\}$. Instances of type CP2 and CP3 were easily solved; most of them were solved at the root node by BB_1 and BB_2.

2.4 Conclusions

In this chapter, we investigated formulations and exact solution approaches for the quadratic minimum spanning tree problem. Initially, we introduced an IP formulation based on RLT and derived a Lagrangian relaxation algorithm based on it. We showed that the formulation over which the Lagrangian subproblem is defined has the integrality property and presented an efficient algorithm for solving it. This lower bounding scheme was embedded in a BB algorithm.

We also introduced a hierarchy of IP formulations, based on the idea of decomposing spanning trees into forests with a fixed number of edges. That type of formulation was used to derive two Lagrangian relaxation bounding procedures. A second BB algorithm based on one of them was implemented. Although the Lagrangian bounds behind the second method are stronger than those provided by the simple RLT formulation, the second algorithm was dominated by the first, in terms of overall running time. That happens because the evaluation of its lower bounds demands excessive CPU running time.

The first BB algorithm benefits a lot from the good trade-off between lower bound quality and the computational effort involved in its evaluation. As a result, a parallel implementation of that BB algorithm managed to solve several instances in the literature for the first time, including some with $n = 50$ vertices.

Chapter 3

The Adjacent-Only Quadratic Minimum Spanning Tree Problem

3.1 Introduction

In this Chapter, we address AQMSTP. The chapter is divided in six main sections. Motivation for the study conducted in the chapter is presented below, in Section 3.1.1. In Section 3.2, we introduce a new AQMSTP formulation, it employs the stars of the graph as modeling entities. Given the large number of variables and constraints in the formulation, in Section 3.3 we develop a row-and-column generation algorithm for evaluating its LP bounds, along with an AQMSTP branch-and-price algorithm based on it. A different approach for evaluating the LP bounds provided by the formulation is studied in Section 3.4: We project out the variables associated to stars in the initial formulation. This results in a reformulation defined on a compact space, with exponentially many additional projection constraints. We use the reformulation to develop a cutting-plane based algorithm for evaluating the LP bounds of the star formulation. This bounding scheme is then used as the basis for a branch-and-cut algorithm. Computational experiments are presented in Section 3.5. Concluding remarks are given in Section 3.6.

3.1.1 Motivation

It should be clear that any of the QMSTP lower bounding procedures outlined in the preceding chapter could be applied to AQMSTP. However, Gilmore-Lawler based bounds tend to be much weaker when applied to AQMSTP. We try to give an argument to support this claim in the discussion below.

Consider an AQMSTP instance. Assume that G is complete and $Q \geq \mathbf{0}$. Let T be a spanning tree induced by a Hamiltonian path of G. Then, the edges at the extremities of the path interact with exactly one other edge each, while each one of the remaining $(n-3)$ edges interact with exactly two other edges. Thus, a total of $2(n-3)+2 = 2(n-2)$ interactions happen in T. This is the minimum number of interactions that is possible in any spanning tree of G. Consider now the application of the Gilmore-Lawler procedure, Algorithm 1, to this AQMSTP instance. When solving (2.14) for an edge $i = \{u, v\} \in E$, the solution spanning tree will employ only one of the edges adjacent to i, namely, edge

$$j \in \arg\min \{q_{ik} : k \in \delta(u) \cup \delta(v)\}.$$

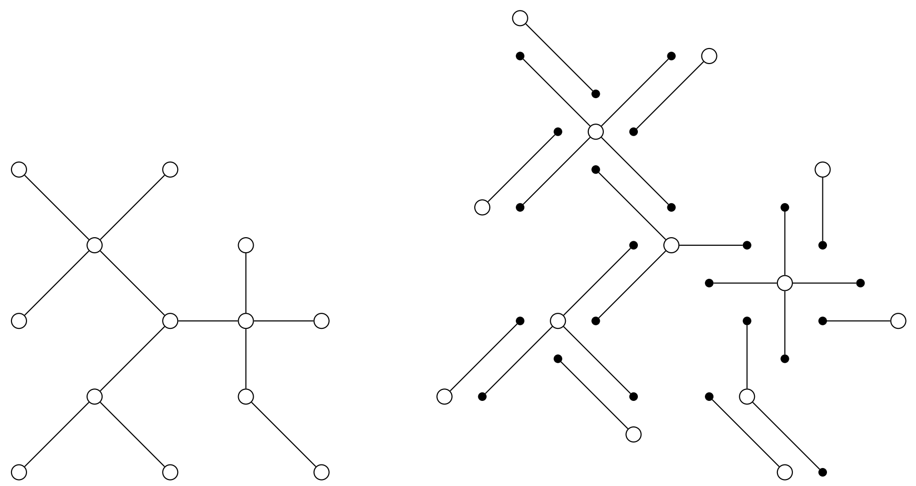
In fact, Assad and Xu [7] used this observation in order to accelerate the computation of Gilmore-Lawler bounds in their AQMSTP bounding procedure. Thus, the Gilmore-Lawler bound will take into account only one interaction for each edge, i.e., $n-1$ interactions, which is about half the minimum number of interactions that can happen. This underestimation on interactions that can happen in an AQMSTP solution leads to very weak lower bounds. This motivates the need for an AQMSTP formulation that takes its particular cost structure into account.

3.2 An AQMSTP Formulation Based on the Stars of G

Since interaction costs are only incurred by pairs of edges that share an endpoint, we can consider the stars of G as modeling entities to obtain a linear integer programming (IP) formulation for AQMSTP. An example of a decomposition of a spanning tree into stars is given in Figure 3.1.

In order to use the stars of G to formulate the problem, let $S^v = \{H \subseteq \delta(v)\}$ be the set of all stars of G centered at $v \in V$ and let $S = \cup_{v \in V} S^v$ be the union of all such sets. Consider a vector of binary decision variables $\mathbf{t} = (t_H)_{H \in S}$, used to indicate whether $(t_H = 1)$ or not $(t_H = 0)$ a star $H \in S$ is included in the AQMSTP solution. Consider also variables $\mathbf{x}$, defined as before. For all $H \in S$, let

$$q_H = \sum_{i,j \in H : i \neq j} q_{ij} + \frac{1}{2} \sum_{i \in H} q_{ii}.$$



(a) A spanning tree T.

(b) Decomposition of T into stars of G. White vertices represent the centers of the stars.

Figure 3.1. Decomposition of a spanning tree into stars.

A binary IP formulation for AQMSTP is given by

$$
\begin{aligned}
(F^*) \qquad & \min \sum_{H \in S} q_H t_H && && (3.1)\\
& \text{s.t.} \sum_{H \in S^v} t_H = 1, && v \in V, && (3.2)\\
& \sum_{H \in S^v : i \in H} t_H = x_i, && i \in \delta(v), v \in V, && (3.3)\\
& \mathbf{x} \in X, && && (3.4)\\
& (\mathbf{x}, \mathbf{t}) \in \mathbb{B}^{m+|S|}. && && (3.5)
\end{aligned}
$$

Constraints (3.2) state that, for each $v \in V$, exactly one star centered at v must be selected. Constraints (3.3) couple variables $\mathbf{x}$ and $\mathbf{t}$, i.e., they grant that x_i assumes value 1 if and only if edge i appears in both stars centered at its endpoints. Constraints (3.4) impose that the edges selected must imply a spanning tree of G.

3.3 A Branch-and-Cut-and-Price Algorithm for AQMSTP

Since F^* employs exponentially many variables $\mathbf{t}$ and subtour elimination constraints (SECs) (2.2), the direct resolution of its LP relaxation is not a practical way to evaluate $Z(F^*)$. To go around that, we develop a row-and-column generation algorithm, described next. That algorithm is then used as the basis for an AQMSTP branch-and-cut-and-price algorithm, described in section 3.3.3.

3.3.1 Row-and-Column Generation Algorithm for Solving the LP Relaxation of F^*

Let $\overline{R} \subseteq \{V' \subset V : |V'| \geq 2\}$ denote a (tiny) subset of the sets for which the SECs are formulated. For each $v \in V$, let $\overline{S}^v \subseteq S^v$ denote a (tiny) subset of the stars of G centered at v. For the sake of solving the linear relaxation of F^*, our algorithm initially considers only the stars in $\{\overline{S}^v : v \in V\}$ and the SECs implied by the sets in $\overline{R}$. Let $\overline{F}^*$ denote the resulting linear master program, restricted in stars, and relaxed in SECs. If the optimal solution for $\overline{F}^*$ satisfies all SECs and if all variables $\mathbf{t}$ have non-negative reduced cost, that solution is also an optimal solution for the linear relaxation of F^*. Otherwise, we either update $\overline{R}$, with vertex sets that imply violated SECs, or update $\{\overline{S}^v\colon\ v \in V\}$, with stars whose variables have negative reduced cost. We then re-optimize a new linear program $\overline{F}^*$. The process is repeated until an optimal solution for the linear relaxation of F^* is obtained. The detailed row-and-column generation algorithm we propose, denoted *RCG*, is presented below.

RCG
Input: AQMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. Initial sets $\overline{R} \subseteq \{V' \subset V : |V'| \geq 2\}$ and $\overline{S}^v \subseteq S^v$ for all $v \in V$. An integer valued parameter $K_{\text{enum}} \geq 0$.
Output: Solution $(\overline{\mathbf{x}}, \overline{\mathbf{t}})$ for the linear relaxation of F^*.

1. Formulate $\overline{F}^*$ in terms of the current sets $\overline{R}$ and $\{\overline{S}^v : v \in V\}$. Let $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{t}})$ be an optimal primal solution for $\overline{F}^*$ and let $\widetilde{\beta} = (\widetilde{\beta}^v)_{v\in V}$, and $\widetilde{\gamma} = (\widetilde{\gamma}_i^v)_{v\in V, i\in\delta(v)}$ be optimal dual variables respectively assigned to constraints (3.2) and (3.3). Denote the row of $\widetilde{\gamma}$ indexed by $v \in V$ as $\widetilde{\gamma}^v = (\widetilde{\gamma}_i^v)_{i\in\delta(v)}$.

2. For any star $H \in S^v$ centered at a vertex $v \in V$, let $\overline{q}_H$ denote its reduced cost, i.e.,
$$\overline{q}_H = q_H - \widetilde{\beta}^v - \sum_{i\in H} \widetilde{\gamma}_i^v.$$
Let $\widetilde{S}^v = \{H \in S^v : |H| \leq K_{\text{enum}}, \overline{q}_H < 0\}$ be the set of the stars centered at v with negative reduced cost and at most K_{enum} edges. If $\widetilde{S}^v \neq \emptyset$, update $\overline{S}^v \leftarrow \overline{S}^v \cup \widetilde{S}^v$ and go back to step 1. Otherwise, proceed to step 3.

3. Solve the separation problem for the SECs:

$$\max \left\{ \sum_{i \in E(U)} \widetilde{x}_i - |U| + 1 : U \subset V, |U| \geq 2 \right\}. \tag{3.6}$$

   Let $\overline{V}$ be a vertex set attaining the minimum. If the corresponding SEC is violated, update $\overline{R} \leftarrow \overline{R} \cup \{\overline{V}\}$ and go back to step 1. Otherwise, proceed to step 4.

4. For each $v \in V$, solve the pricing problem

$$(PP(v)) \qquad \min \{ \overline{q}_H : H \in S^v \}.$$

   Let $\overline{H}$ be a star attaining the minimum. If $\overline{q}_{\overline{H}} < 0$, update $\overline{S}^v \leftarrow \overline{S}^v \cup \{\overline{H}\}$ and go back to step 1. Otherwise, $(\overline{\mathbf{x}}, \overline{\mathbf{t}}) = (\widetilde{\mathbf{x}}, \widetilde{\mathbf{t}})$ is an optimal solution for the linear relaxation of F^*.

Before moving on, let us point out that in order to obtain a valid lower bound for AQMSTP, we do not need to wait until no negative reduced cost stars are found, in step 4. By adding $\overline{q}_{\overline{H}}$ to each $\widetilde{\beta}^v$, we obtain a set of dual multipliers for which no star of negative reduced costs exists, i.e., we obtain a feasible solution for the dual of the linear relaxation of F^*, whose corresponding objective function value is

$$\sum_{v \in V} \left(\sum_{H \in \overline{S}^v} q_H \overline{t}_H + \overline{q}_{\overline{H}} \right). \tag{3.7}$$

As such, this value is a lower bound on the optimal AQMSTP solution.

In our implementation of *RCG*, we initialize the algorithm with $\overline{R} = \emptyset$. SECs are then added to $\overline{R}$ only when the separation problem is solved, in step 3. To that aim, we make use of the separation algorithm introduced in Padberg and Wolsey [37], which involves solving $O(n)$ max-flow problems, in conveniently defined networks, at a $O(n^4)$ worst case complexity. Besides the most violated, that algorithm might also return other violated SECs. If that is the case, we also add them to $\overline{R}$.

In order to obtain a linear program $\overline{F}^*$ for which an initial basic feasible solution exists, we initialize the sets $\{\overline{S}^v : v \in V\}$ with the stars in a feasible AQMSTP solution. That solution is obtained by the randomized AQMSTP heuristic described in Section 2.2. Additional columns are added to the model when the sets $\{\overline{S}^v : v \in V\}$ are updated, in steps 2 and 4. In step 2, the pricing problems are solved heuristically, while, in step 4, they are solved exactly.

Step 4 accounts for most of the CPU time of the algorithm, that is why it is the last step to be executed. Step 2 is an attempt to accelerate the method by enumerating and computing the reduced costs of promising stars. More precisely, the algorithm computes the reduced cost for all stars $\{H \in S : |H| \leq K_{\text{enum}}\}$, for a small valued algorithm parameter K_{enum}. Such a pricing heuristic is grounded on the fact that the cost q_H of a star $H \in S$ grows very rapidly with the number of edges in the star. Consequently, it is unlikely that stars employed at the optimal solution of the linear relaxation of F^* (as well as in optimal solutions for AQMSTP) have too many edges. Since the problems solved for each $v \in V$ in steps 2 and 4 are independent, they are solved in parallel.

We now discuss how the exact pricing problem is solved.

3.3.2 Solving the Pricing Problem $\mathrm{PP(v)}$

Our solution strategy for exactly solving the pricing problem consists of modeling it as an IP, which is then solved by means of an IP commercial solver. The first step in that direction is to formulate $PP(v)$ as a quadratic binary program. In order to do so, we consider binary variables $\mathbf{w} = (w_i)_{i \in \delta(v)}$, used to select the edges in a star of G centered at v. More precisely, if an edge $i \in \delta(v)$ is included in the star, $w_i = 1$. Otherwise, $w_i = 0$. We can now formulate $PP(v)$ as the following unconstrained quadratic binary program:

$$\min \left\{ \sum_{i,j \in \delta(v)} \overline{q}_{ij} w_i w_j : \mathbf{w} \in \mathbb{B}^{|\delta(v)|} \right\} - \widetilde{\beta}^v, \tag{3.8}$$

where $\overline{q}_{ii} = q_{ii} - \widetilde{\gamma}_i^v$ and $\overline{q}_{ij} = q_{ij}$ for $i \neq j \in \delta(v)$.

Although polynomial time algorithms do exist for solving quadratic binary optimization problems in which the diagonal entries of the cost matrix are non-negative and the off-diagonal entries are non-positive [11], we are not aware of any polynomial time algorithm for solving problems like (3.8), where the opposite holds. Indeed, the general unconstrained quadratic binary optimization problem is NP-Hard [47].

In order to solve $PP(v)$, we linearize (3.8) and solve the resulting IP. To that aim, consider the introduction of linearization variables $\mathbf{z} = (z_{ij})_{i,j \in \delta(v), i<j}$. Let $\mathbf{z}_i = (z_{ij})_{j \in \delta(v): j>i}$ for all $i \in \delta(v)$, and $z_{ji} = z_{ij}$ if $j > i$. A binary IP for $PP(v)$ is

$$-\widetilde{\beta}^v + \min \sum_{i,j \in \delta(v): i \neq j} \overline{q}_{ij} z_{ij} + \sum_{i \in \delta(v)} \overline{q}_{ii} w_i \tag{3.9}$$

$$
\begin{aligned}
\text{s.t.} \qquad && z_{ij} \le w_i, \qquad && i,j \in \delta(v), i < j, && (3.10)\\
&& z_{ij} \le w_j, \qquad && i,j \in \delta(v), i < j, && (3.11)\\
&& w_i + w_j - 1 \le z_{ij}, \qquad && i,j \in \delta(v), i < j, && (3.12)\\
&& (\mathbf{w}, \mathbf{z}) \in \mathbb{B}^{|\delta(v)|+\frac{1}{2}(|\delta(v)|^2-|\delta(v)|)}. && && (3.13)
\end{aligned}
$$

To solve (3.9)-(3.13), we propose a strategy that consists of a preprocessing phase followed by a cut-and-branch phase. In the first, we try to fix variables and generate optimality cuts for the problem. We also try to use the fact that the solution space was partially explored by enumeration to avoid entering the second phase, where the IP (3.9)-(3.13) is solved. If the second phase is needed, we carry the optimality cuts and variable fixing information to (3.9)-(3.13) and then solve the problem via BB. To implement the second phase we make use of a commercial IP solver. The strategies employed in the first phase are described next.

3.3.2.1 Optimality Cuts and Acceleration Strategies

In the following, assume that an upper bound u on the optimal AQMSTP solution value is known and that no negative reduced cost star has K_{enum} or fewer edges.

Let $K_{\max}$ be the smallest integer k such that $q_H \ge u$ for all $H \in S^v$, $|H| > k$. Then, the optimality cut (3.14) below states that no star centered at v could have more than $K_{\max}$ edges in an optimal AQMSTP solution:

$$\sum_{i \in \delta(v)} w_i \le K_{\max}. \tag{3.14}$$

This observation is valid, since under non-negative costs, using a star with more than $K_{\max}$ edges would lead to a spanning tree whose cost is at least u. If $K_{\max} \le K_{\text{enum}}$, we can stop the resolution of the pricing problem for this particular $v \in V$, since all stars with $K_{\max}$ or fewer edges were already dismissed. Note that $K_{\max}$ can be a precomputed-computed value that is updated every time a better upper bound u is found. In our implementation, this value is computed with the initial upper bound u given to our BB algorithm and remains the same until its termination.

Since we assumed $Q \ge \mathbf{0}$, if $\overline{q}_{ii} \ge 0$, we can set $w_i = 0$ without worsening the cost of any solution. For this reason, we assume $\overline{q}_{ii} < 0$ for all $i \in \delta(v)$ in what follows.

Let $(\overline{\mathbf{w}}, \overline{\mathbf{z}})$ be a feasible solution for (3.9)-(3.13). Observe that if $\sum_{j \in \delta(v) \setminus \{i\}} \overline{w}_j (\overline{q}_{ij} + \overline{q}_{ji}) \ge |\overline{q}_{ii}|$ for some $i \in \delta(v)$ such that $\overline{w}_i = 1$, a solution not worse than $(\overline{\mathbf{w}}, \overline{\mathbf{z}})$ could be obtained by setting $\overline{w}_i = 0$. The following optimality cut follows directly from this observation:

$$\sum_{j\in\delta(v):j\neq i} (\overline{q}_{ij}+\overline{q}_{ji})z_{ij} \leq |\overline{q}_{ii}|w_i.$$

The observation above is also helpful in devising the following variable fixing test. Given $i \in \delta(v)$, sort the elements of $\delta(v)\setminus\{i\}$ as $(j_1, ..., j_{|\delta(v)|-1})$ in non-decreasing order of the values $\overline{q}_{ij}+\overline{q}_{ji}$. Let $\overline{k}$ be the smallest k for which $\sum_{l=1}^{k}(\overline{q}_{ij_l}+\overline{q}_{j_l i}) \geq |\overline{q}_{ii}|$. Then, we can set $\overline{w}_i = 0$ in any solution $(\overline{\mathbf{w}}, \overline{\mathbf{z}})$ that uses i along with $\overline{k}$ or more other edges from $\delta(v)$ without worsening the solution. Thus, we are interested in solutions in which i is selected along with at most $\overline{k}-1$ other edges. If $\overline{k} \leq K_{\text{enum}}$, these solutions were already dismissed and we can set $w_i = 0$. After running this test for all $i \in \delta(v)$, if any variable has been fixed, the process is repeated, considering then only free variables. When all edges $i \in \delta(v)$ have been tested and no further variables could be fixed, we obtain two optimality cuts:

$$w_i + \sum_{l=1}^{\overline{k}} w_{j_l} \leq \overline{k},$$

and

$$\sum_{j\in\delta(v)\setminus\{i\}} z_{ij} \leq (\overline{k}-1)w_i.$$

Finally, for all k such that $K_{\text{enum}} < k < K_{\max}$, a lower bound on the reduced cost of the star containing exactly k edges is given by:

$$\min_{H\in S^v:|H|=k}\{q_H\} - \widetilde{\beta}^v - \max_{H\in S^v:|H|=k}\{\widetilde{\gamma}_i^v\}.$$

Thus, if the bounds are non-negative for all k above, we can conclude that there are no stars with negative reduced cost in S^v. In our implementation, the minimization term in the above expression is a precomputed-computed value, while the maximization term is computed by sorting the edges in $\delta(v)$ in non-increasing order of the costs $\{\widetilde{\gamma}_i^v : i \in \delta(v)\}$ and taking the sum over the first k edges.

3.3.3 Branch-and-Cut-and-Price Implementation Details

We combine the lower bounding procedure *RCG* with branch-and-bound to develop a branch-and-cut-and-price algorithm, *BCP*, for AQMSTP.

BCP is initialized with a valid upper bound provided by the heuristic described in Section 2.2. Such a spanning tree also provides the initial stars $\{\overline{S}^v : v \in V\}$ used to obtain an initial basic feasible solution for *RCG*. For the remaining *BCP* nodes, the sets $\{\overline{S}^v : v \in V\}$ and $\overline{R}$ used to formulate the very first linear program are those available when the last linear program was solved at the parent node.

In order to avoid solving too many pricing problems by the exact procedure, *RCG* is always aborted right after the first round of exact resolution of pricing problems is concluded, no matter which node, the root or its descendants, is being investigated. The lower bound (3.7) obtained at that moment is taken as a lower bound on the optimum for that node. To deal with possible infeasibility caused by branching along the tree, artificial variables with large costs are added to F^*. A node is declared infeasible whenever an artificial variable has a positive value in the optimal solution of the linear relaxation of that node.

Assume that a fractional vector $(\overline{\mathbf{x}}, \overline{\mathbf{t}})$ solves the linear relaxation of a *BCP* node. Before branching, we first try to fix non-basic variables $\mathbf{x}$ by means of their reduced costs. After that, a different variable fixing procedure is carried out as follows. We first create a candidate list C of the fractional variables. To that aim, let $f_i = 1/2 - |1/2 - \overline{x}_i|$ and $f^* = \max\{f_i : i \in E\}$. Initialize $C \leftarrow \emptyset$. Then, add fractional edges to C according to a probability given by f_i/f^*. After that, for each edge $i \in C$, let h_i^1 (resp. h_i^0) be the lower bound obtained by *RCG* when the constraint $x_i = 1$ (resp. $x_i = 0$) is further imposed to the the current node. If either h_i^0 or h_i^1 is larger than a known valid upper bound for AQMSTP, the corresponding variable can be fixed. After examining all edges in C, if any of them has been fixed, we solve the node again. When no variable can be fixed by the procedures outlined above, we branch on the edge

$$i \in \arg\max \left\{\min\left\{h_j^0, h_j^1\right\} : j \in C\right\}.$$

Two new subproblems are then created. For one of them we impose $x_i = 1$, and for the other, $x_i = 0$. Observe that integrality of $\mathbf{x}$ implies integrality of $\mathbf{t}$. Thus, there is no need to branch on $\mathbf{t}$. The procedure we just described for selecting the branching variable is a strong branching strategy [1]. Pseudo-cost based branching was also tested, but better results were obtained with the policies we described above. In order to save computer memory, *BCP* performs a depth-first search.

3.4 A Branch-and-Cut Algorithm for AQMSTP

In this section, we study the projection of F^* on the space of the $\mathbf{x}$ variables. The projection constitutes a reformulation of F^*, with the same linear relaxation bounds. If, on the one hand, the projection has a compact variable space, on the other, an exponential number of additional constraints is necessary in its description. Still, we investigate the resolution of the linear relaxation of this reformulation as an alternative for evaluating $Z(F^*)$. We discuss some advantages of this approach in Section 3.4.2. In order to solve that relaxation, we implement a cutting plane algorithm that separates two classes of valid inequalities: SECs and projection cuts. The reformulation and the cutting plane algorithm are presented in the next two sections. After that, in Section 3.4.3, we discuss a branch-and-cut algorithm based on the cutting plane algorithm.

3.4.1 Projecting Out Variables $\mathbf{t}$

First, recall that integrality of $\mathbf{x}$ implies integrality of $\mathbf{t}$. Therefore, we assume that $\mathbf{t} \in \mathbb{B}^{|S|}$ is replaced by $\mathbf{t} \in \mathbb{R}_+^{|S|}$ in our AQMSTP reformulation F^*. Moreover, since the objective function is written in terms of those variables, we make use of artificial continuous variables $\mathbf{p} = (p_v)_{v \in V}$ to state the objective function of F^* as

$$\min \sum_{v \in V} p_v,$$

provided that constraints

$$p_v \geq \sum_{H \in S^v} q_H t_H, \quad v \in V \tag{3.15}$$

are appended to the formulation.

Assume that dual multipliers $\alpha = (\alpha^v)_{v \in V}$, $\beta = (\beta^v)_{v \in V}$, and $\gamma = (\gamma_i^v)_{v \in V, i \in \delta(v)}$, are respectively associated to constraints (3.15), (3.2), and (3.3). Denote by $\gamma^v = (\gamma_i^v)_{i \in \delta(v)}$ the row of γ indexed by $v \in V$. Then, the projection cone for the variables associated to stars in S^v for any $v \in V$, is given by:

$$(C^v) \qquad \sum_{i \in H} \gamma_i^v + \beta^v \leq q_H \alpha^v, \qquad H \in S^v, \tag{3.16}$$
$$(\alpha^v, \beta^v, \gamma^v) \in \mathbb{R}_+ \times \mathbb{R} \times \mathbb{R}^{\delta(v)}.$$

Using $\{C^v \colon v \in V\}$, we obtain the projection of F^* on the $(\mathbf{x}, \mathbf{p})$ space:

$$(F^*_{\mathbf{x}}) \qquad \min \sum_{v \in V} p_v$$
$$\text{s.t.} \quad \alpha^v p_v \geq \beta^v + \sum_{i \in \delta(v)} \gamma_i^v x_i, \qquad v \in V, (\alpha^v, \beta^v, \gamma^v) \in T(C^v), \tag{3.17}$$
$$\mathbf{x} \in X,$$
$$(\mathbf{x}, \mathbf{p}) \in \mathbb{B}^m \times \mathbb{R}^n, \tag{3.18}$$

where $T(C^v)$ denotes the set of the extreme rays of the projection cone C^v.

From projection theory, we know that $(\overline{\mathbf{x}}, \overline{\mathbf{p}})$ is feasible for the linear relaxation of $F^*_{\mathbf{x}}$ if and only if there exists $\overline{\mathbf{t}}$ such that $(\overline{\mathbf{x}}, \overline{\mathbf{t}}, \overline{\mathbf{p}})$ is feasible for the linear relaxation of F^*. Consequently, F^* and $F^*_{\mathbf{x}}$ yield the same linear relaxation bound.

3.4.2 A Cutting Plane Algorithm for Solving the LP Relaxation of $\mathbf{F}^*_{\mathbf{x}}$

Let $\overline{T}(C^v) \subseteq T(C^v)$, $v \in V$, be a (tiny) subset of the extreme rays of C^v. Once again, consider $\overline{R} \subseteq \{V' \subset V : |V'| \geq 2\}$. Instead of explicitly taking into account all SECs and constraints (3.17) implied by $\{T(C^v)\colon\ v \in V\}$ in the linear relaxation of $F^*_{\mathbf{x}}$, we consider only those respectively implied by the subsets $\overline{R}$ and $\{\overline{T}(C^v) : v \in V\}$. The resulting relaxation is denoted $\overline{F}^*_{\mathbf{x}}$. If $\overline{F}^*_{\mathbf{x}}$ is solved and its optimal solution satisfies all SECs and projection constraints (3.17), the solution vector also solves the linear relaxation of $F^*_{\mathbf{x}}$. Otherwise, violated SECs and projection cuts are used to reinforce the relaxation. More precisely, we update $\overline{R}$ with vertex sets of violated SECs or $\{\overline{T}(C^v) : v \in V\}$ with extreme rays for which projection cuts (3.17) are violated. A strengthened relaxation $\overline{F}^*_{\mathbf{x}}$ is then optimized. The process is repeated until an optimal solution for the linear relaxation of $F^*_{\mathbf{x}}$ is found. The main details are given in the algorithm below, denoted CP:

CP

Input: AQMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}^{m^2}_{+}$. Sets $\overline{R} \subseteq \{V' \subset V : |V'| \geq 2\}$ and $\overline{T}(C^v) \subseteq T(C^v)$ for all $v \in V$.

Output: Solution $(\overline{\mathbf{x}}, \overline{\mathbf{p}})$ for the linear relaxation of $F^*_{\mathbf{x}}$.

1. Formulate and solve the relaxation $\overline{F}^*_{\mathbf{x}}$ implied by the current restricted sets $\overline{R}$ and $\overline{T}(C^v)$. Let $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{p}})$ be an optimal solution.

2. Solve (3.6), the SEC separation problem, for $\widetilde{\mathbf{x}}$. Let $\overline{V}$ be a vertex set attaining the minimum. If the corresponding SEC is violated by $\widetilde{\mathbf{x}}$, update $\overline{R} \leftarrow \overline{R} \cup \overline{V}$.

3. For every $v \in V$, solve the separation problem for the projection cuts (3.17):

$$
\begin{aligned}
(SP(v)) \qquad & \max \beta^v + \sum_{i \in \delta(v)} \gamma^v_i \widetilde{x}_i - \alpha^v \widetilde{p}^v && (3.19)\\
& \text{s.t.} \quad (\alpha^v, \beta^v, \gamma^v) \in C_v, && (3.20)\\
& \qquad \alpha^v = 1. && (3.21)
\end{aligned}
$$

Let $(\overline{\alpha}^v, \overline{\beta}^v, \overline{\gamma}^v)$ be an optimal solution. If (3.17) with $(\overline{\alpha}^v, \overline{\beta}^v, \overline{\gamma}^v)$ is violated by $(\widetilde{\mathbf{x}}, \widetilde{\mathbf{p}})$, update $\overline{T}(C^v) \leftarrow \overline{T}(C^v) \cup \{(\overline{\alpha}^v, \overline{\beta}^v, \overline{\gamma}^v)\}$.

4. If violated constraints were found in steps 2 or 3, go back to step 1. Otherwise, stop since $(\overline{\mathbf{x}}, \overline{\mathbf{p}}) = (\widetilde{\mathbf{x}}, \widetilde{\mathbf{p}})$ solves the linear relaxation of $F^*_{\mathbf{x}}$.

Similarly to steps 2 and 4 of RCG, the separation of projection cuts, step 3 of CP, is executed in parallel for each $v \in V$ in our implementation. Note that we assume $\alpha^v = 1$ in these cuts. Before discussing how they can be separated, we first show that this assumption makes each $SP(v)$ bounded and does not imply in loosing any eventually violated constraint (3.17). To that aim, assume for a moment that $\alpha^v > 0$. In this case, since the other variables can be conveniently scaled, we can assume $\alpha^v = 1$. Consider now the dual of $SP(v)$:

$$
\begin{aligned}
(DSP(v)) \qquad & \min \sum_{H \in S^v} q_H t_H && && (3.22)\\
& \text{s.t.} \sum_{H \in S^v} t_H = 1, && && (3.23)\\
& \sum_{H \in S^v : i \in H} t_H = \widetilde{x}_i, && i \in \delta(v), && (3.24)\\
& \mathbf{t} \in \mathbb{R}_+^{|S|}. && && (3.25)
\end{aligned}
$$

$DSP(v)$ can be seen as the problem of finding the cheapest way to represent $\widetilde{\mathbf{x}}$ as a convex combination of stars centered at v. Note that the stars centered at v are simply the vertices of the 0-1 box in $\mathbb{R}^{|\delta(v)|}$. Since the point $\widetilde{\mathbf{x}}$ to be separated satisfies $\widetilde{\mathbf{x}} \in [0,1]^m$, $DSP(v)$ is always feasible, what implies that $SP(v)$ is bounded. Now, assume that there is a violated constraint (3.17) for which $\alpha^v = 0$, i.e., there is some $\widetilde{\beta}^v$ and $\widetilde{\gamma}^v$ such that $\widetilde{\beta}^v + \sum_{i \in \delta(v)} \widetilde{\gamma}_i^v \widetilde{x}_i > 0$. This would be an improving ray for $SP(v)$, contradicting the fact that it is bounded. Thus, we can safely assume $\alpha^v = 1$.

Note that $DSP(v)$ closely resembles F^*. In fact, the reduced cost expression for variables $\mathbf{t}$ is precisely the same as in F^*. Thus, we can apply the same algorithm outlined in section 3.3.2 to find stars of negative reduced cost. Therefore, instead of solving $SP(v)$, we actually solve $DSP(v)$ by column generation. To that aim, consider the restricted problem $\overline{DSP}(v)$, which is obtained from $DSP(v)$ by considering only those variables $\mathbf{t}$ implied by stars in the restricted set $\overline{S}^v \subseteq S^v$. The column generation algorithm, denoted CG, is outlined below.

CG
Input: AQMSTP instance given by $G = (V, E)$ and $Q \in \mathbb{R}_+^{m^2}$. A vertex $v \in V$. Values $\{0 \leq \widetilde{x}_i \leq 1, i \in \delta(v)\}$ and $\widetilde{p}^v \in \mathbb{R}$. A set $\overline{S}^v \subseteq S^v$.
Output: A solution $(1, \overline{\beta}^v, \overline{\gamma}^v)$ for $SP(v)$.

1. Formulate and solve the problem $\overline{DSP}(v)$ implied by the current set $\overline{S}^v$. Let $\widetilde{\mathbf{t}}$ be an optimal primal solution. Let $\widetilde{\beta}^v$ and $\widetilde{\gamma}^v$ be optimal dual variables respectively associated to constraints (3.23) and (3.24).
2. Let $\widetilde{S}^v = \{H \in S^v : |H| \leq K_{\text{enum}}, \overline{q}_H < 0\}$. If $\widetilde{S}^v \neq \emptyset$, update $\overline{S}^v \leftarrow \overline{S}^v \cup \widetilde{S}^v$ and go back to step 1. Otherwise, proceed to step 3.
3. Solve $PP(v)$. Let $\overline{H}$ be a star attaining the minimum. If $\overline{q}_{\overline{H}} < 0$, update $\overline{S}^v \leftarrow \overline{S}^v \cup \{\overline{H}\}$ and return to step 1. Otherwise, $\widetilde{\mathbf{t}}$ is an optimal solution for $DSP(v)$ and $(1, \overline{\beta}^v, \overline{\gamma}^v) = (1, \widetilde{\beta}^v, \widetilde{\gamma}^v)$ is an optimal solution for $SP(v)$.

To grant that a feasible solution for $DSP(v)$ is readily available at each call to CG, we initialize $\overline{S}^v$ with the stars generated by the following algorithm.

Algorithm 7
Input: Graph $G = (V, E)$, vertex $v \in V$, and values $\{0 \leq \widetilde{x}_i \leq 1, i \in \delta(v)\}$.
Output: Set $\widetilde{S}^v$ that gives a feasible solution for $DSP(v)$.

1. Let $\widehat{\mathbf{x}} = \widetilde{\mathbf{x}}$ and consider a set $\widehat{S}^v = \emptyset$.
2. Update $\widehat{S}^v \subseteq S^v$ as follows:
   a) Let $H = \{i \in \delta(v) : \widehat{x}_i > 0\}$ and add H to $\widehat{S}^v$.
   b) Let $\widehat{t}_H = \min\{\widehat{x}_i : i \in \delta(v), \widehat{x}_i > 0\}$.
   c) Set $\widehat{x}_i = \widehat{x}_i - \widehat{t}_H$ for all $i \in \delta(v)$ with $\widehat{x}_i > 0$.
   d) If $\widehat{\mathbf{x}} = \mathbf{0}$ go to step 3, otherwise repeat step 2.
3. Let $H_\emptyset$ be the empty star and set $\widehat{t}_{H_\emptyset} = 1 - \sum_{H \in \widehat{S}^v} \widehat{t}_H$. The set $\widetilde{S}^v = \widehat{S}^v$ gives a feasible solution for $DSP(v)$.

To see that the set $\widetilde{S}^v$ obtained by Algorithm 7 indeed gives a feasible solution for $DSP(v)$, observe that

$$\widetilde{x}_i = \sum_{H \in \widetilde{S}^v : i \in H} \widehat{t}_H, \quad \forall i \in \delta(v) \qquad \text{and} \qquad \sum_{H \in \widetilde{S}^v} \widehat{t}_H = 1.$$

In CP, the most expensive step is the separation of (3.17) in Step 3, since separating those constraints involves the exact resolution of $PP(v)$. In order to accelerate the convergence of the separation algorithm CG and solve as few exact pricing problems as possible, we keep in $\overline{S}^v$ the stars generated during the previous calls to CG. Another strategy to reduce CPU times is to abort CG as soon as the first exact pricing problem is solved. In that case, a feasible solution for the dual of $DSP(v)$ can be obtained by adding $\overline{q}_{\overline{H}}$ to $\widetilde{\beta}^v$. Since the dual of $DSP(v)$ is $SP(v)$, this solution yields a valid constraint (3.17). In fact, we resort to this strategy in our branch-and-cut algorithm.

Each $PP(v)$ in CG is solved as discussed in Section 3.3.2. However, in CG it is possible to develop another strategy for accelerating the resolution of that pricing problem. Basically, all edges i such that $\widetilde{x}_i = 0$ in the point $\widetilde{\mathbf{x}}$ to be separated can be forbidden in the resolution of the pricing. Consider the dual variables $\widetilde{\gamma}^v$ in the optimal solution for the dual of $\overline{DSP}(v)$, obtained in Step 1 of CG. We know from Section 3.3.2 that if $\widetilde{\gamma}^v_i \leq 0$ for some $i \in \delta(v)$, we can safely assume that i will not be used in the solution of $PP(v)$. Now, if $\widetilde{\gamma}^v_i > 0$, setting $\widetilde{\gamma}^v_i = 0$ still gives multipliers that are feasible for the dual of $\overline{DSP}(v)$. If $\widetilde{x}_i = 0$, they are also optimal. Consequently, we can set $\widetilde{\gamma}^v_i = 0$ for all i such that $\widetilde{x}_i = 0$ and possibly accelerate the exact resolution of $PP(v)$. While this scheme indeed reduces the overall computational effort of the exact pricing, the convergence of CP is badly affected, since sparser cuts are generated by CG, i.e., $\overline{\gamma}^v_i = 0$ for all i such that $\widetilde{x}_i = 0$.

To overcome this undesired side effect, we seek to reduce $\widetilde{\gamma}^v_i$ by the smallest possible amount while still granting that i will be fixed out of the solution of $PP(v)$. Again, let $(j_1, ..., j_{|\delta(v)|-1})$ be a sorting of $\delta(v) \setminus \{i\}$ in non-decreasing order of $\overline{q}_{ij} + \overline{q}_{ji}$. Assume $\widetilde{\gamma}^v_i > q_{ii}$ and $\sum_{l=1}^{K_{\text{enum}}} (\overline{q}_{ij_l} + \overline{q}_{j_l i}) < |\overline{q}_{ii}| = \widetilde{\gamma}^v_i - q_{ii}$, since otherwise i would have been ruled out by the methods discussed in Section 3.3.2. We then reduce $\widetilde{\gamma}^v_i$ to $\sum_{l=1}^{K_{\text{enum}}} (\overline{q}_{ij_l} + \overline{q}_{j_l i}) + q_{ii}$. In doing so, we guarantee that i is fixed out of $PP(v)$, while keeping part of the dual information obtained through enumeration. Consequently, the resolution of $PP(v)$ becomes faster without slowing down the overall convergence of CP.

Although in CP we still have to solve the same type of pricing problems solved by RCG, there are some advantages, from a computational perspective, in evaluating $Z(F^*)$ by means of CP. We highlight some of them below.

If we wanted to further strengthen F^*, we would have to employ inequalities involving variables $\mathbf{t}$, since the spanning tree polytope is already completely described by (2.1)-(2.3). These inequalities would certainly affect the structure of $PP(v)$. On the other hand, in $F^*_{\mathbf{x}}$, we can apply general valid inequalities generating procedures, e.g., Chvátal-Gomory cuts, to obtain new inequalities that eventually will strengthen the formulation. The difference is that projection inequalities can be used to obtain new valid inequalities without affecting $PP(v)$, while in F^* it is troublesome to use variables associated to stars to obtain new valid inequalities. Therefore, general classes of valid inequalities separated by most commercial solvers can be applied to $F^*_{\mathbf{x}}$ (with smaller coding effort) in a branch-and-cut algorithm, in order to further strengthen $Z(F^*)$.

Moreover, since only cutting planes are being generated, the implementation of a branch-and-cut algorithm is straightforward. It suffices to embed CP in a branch-and-cut framework, available in most commercial optimization packages. Another advantage that follows from the same reasoning is that this branch-and-cut algorithm can benefit from the expertise in cut generation, variable fixing and selection, branching, and tree management, available in most commercial packages.

3.4.3 Branch-and-Cut Implementation Details

We combine the lower bounding procedure CP with BB to develop BC, a branch-and-cut algorithm for AQMSTP. To implement BC, we simply embed CP into the branch-and-cut framework of a commercial optimization package. We let the solver manage the search, all configuration parameters are left at their default values.

There are two differences between the implementation of CP used in BC and the one described in section 3.4.2. In BC, for each $v \in V$, the separation routine CG of CP is stopped as soon as the first exact pricing problem is solved. The dual feasible solution obtained by adding $\overline{q}_{\overline{H}}$ to $\widetilde{\beta}^v$ is then used for generating a projection cut. Additionally, we only add to $\overline{R}$ the projection cuts that are violated by at least $0.01 \times \widetilde{p}^v$.

As in BCP, the BC search tree is initialized with the best AQMSTP feasible solution obtained with the multi-start heuristic described in Section 2.2.

To conclude this section, we would like to point out that the projection approach just described is related to Benders decomposition [8, 33]. $F^*_{\mathbf{x}}$ can be seen as a Benders master program, while $SP(v)$, $v \in V$, is the corresponding Benders subproblem. Instead of solving an integer Benders master program, we obtain lower bounds by solving its continuous relaxation. These bounds are then used inside a BB framework to obtain optimal solutions. Additionally, our analogous to the Benders subproblem involves an exponential number of variables, and is solved by column generation.

3.5 Computational experiments

The algorithms presented here were implemented with the help of the commercial optimization package IBM ILOG CPLEX 12.5. The CPLEX LP solver was used to solve the LPs in RCG and CG. The CPLEX IP solver was used to solve IPs during the the exact resolution of the pricing problems. BC was implemented by embedding CP into the CPLEX branch-and-cut framework. In all cases, CPLEX parameters were left at their default values. The OpenMP API was used to parallelize steps 2 and 4 of RCG and step 3 of CP. All algorithms were coded in C++ and compiled with G++ 4.6.3, optimization flag O3 turned on. The same computational environment described in Section 2.3 was used.

Since AQMSTP instances in the literature were not available to us, new test instances were generated according to the guidelines in [7]. Instances considered here are defined over complete graphs with $n \in \{15, 20, 30, 40, 50\}$ vertices. For each size n, ten instances were generated. Diagonal entries of Q are randomly chosen integers in the range $[0, 100]$, while off-diagonal entries are integers in $[0, 20]$, also chosen with uniform probability.

Keeping in mind that the solution strategies described in Chapter 2 dominate other solution strategies proposed in the literature, we compare the AQMSTP specialized algorithms proposed in this chapter with them.

The code we use for the lower bound evaluation precisely implements the lower bounding procedures RCG and CP, respectively described in Sections 3.3.1 and 3.4.2. Recall that in BCP, however, we stop RCG as soon as the first round of pricing problems is solved. The same applies for the separation procedure CG in BC. Additionally, in BC we only add to the relaxation the projection cuts that are violated by more than a threshold value.

In Table 3.1, AQMSTP lower bounds are presented. The table is divided in five sets of columns. In the first we present instance information: the size n, an integer id in the range $[1, 10]$ that stands as an instance identifier, and the optimal cost ub^*. The next three sets present computational results for F^*, F_1, F_2 for $K = 2$, and the formulation of Assad and Xu [7], F_{AX92}. For each formulation we present the lower bound lb, the implied duality gap gap, and the computational time $t(s)$ (in seconds) taken by the respective algorithm to evaluate the bound. The lower bounds implied by F^* were evaluated by RCG and CP. The lower bounds implied by F_1 and F_2 were respectively evaluated by Lag_1 and Lag_2. The lower bounds implied by F_{AX92} were evaluated by our implementation of the dual ascent procedure presented in [7].

Computational results in Table 3.1 help supporting the claim that Gilmore-Lawler based bounds are not adequate for AQMSTP. They indicate that F^* formulates the AQMSTP cost structure much more appropriately than F_1, F_2, and F_{AX92}. Taking $n = 50$ as a baseline, we see that average duality gaps implied by F^* (4.6%) are respectively, 12.4, 8.3, and 16.6 times stronger than F_1 (57.2%), F_2 (38.5%), and F_{AX92} (76.4%). Formulation F_2, which provided strong lower bounds for the general QMSTP case, was largely dominated by F^*, in terms of lower bound quality and computational time. Additionally, the Gilmore-Lawler based bounds seem to worsen more rapidly than F^* as n increases.

Compared to RCG, CP demands a much higher computational effort to evaluate $Z(F^*)$, as n grows. The reason being that, every time projection cuts are separated, the CG algorithm needs to be executed. Thus, more exact pricing problems need to be solved exactly by CP than by the RCG algorithm.

In Table 3.2, we present a comparison of BCP, BC, and BB_1, which was the best performing algorithm for QMSTP in the previous chapter. For each algorithm, we report the best upper (ub) and lower (lb) bounds found during the search, the duality gap ($100\frac{ub-lb}{ub}$) at termination, the total number of nodes investigated during the search (nodes), and the CPU time ($t(s)$, in seconds) to obtain such results. Each algorithm was allowed to run for as much as 10 hours, for each instance. Since BCP and BC solved all instances with up to 40 vertices to proven optimality whereas BB_1 could not solve a single instance with 30 or more vertices, computational results for that algorithm are not provided for $n \geq 40$.

Computational results in Table 3.2 suggest that both BCP and BC are much faster than the Gilmore-Lawler based BB algorithm BB_1. Although lower bounds given Lag_1 can be evaluated in CPU times that are at least one order of magnitude smaller than the RCG and CP counterparts, the strength of $Z(F^*)$ makes up for the additional computational effort. To validate such a claim, observe that, on the average, BCP and BC solve instances with $n \leq 20$ two orders of magnitude faster than BB_1.

For the instances in our test set, BC clearly outperformed BCP, not only in terms of CPU times, but also in terms of the size of the instances that could be solved to proven optimality. Whereas BCP solved all instances with up to 40 vertices, BC also solved all instances with 50 vertices. This is a surprising result, since RCG was much faster than CP, in the results given in Table 3.1. One of the reasons is that, differently from the CP implementation used to compute the data in Table 3.1, in the CP implementation used in BC the separation procedure for projection cuts is aborted early and only cuts that are violated by more than a threshold are added to the formulation. These two strategies speed up the resolution of the root and other BB nodes in BC. In Appendix D, we provide more detailed AQMSTP BB results, the claim above is validated by the data provided in those results.

A second reason is that most of the projection cuts are generated by BC at the root node; only few additional cuts are generated at the descendant nodes. Thus, the resolution of the remaining nodes is much less time demanding than the root node. A third aspect that might contribute in that direction is the fact that, for BC, the search tree was managed by CPLEX, whereas, for BCP, the search tree was implemented by ourselves in a depth-first fashion. BC is likely to benefit from several policies for variable fixing, variable selection, and heuristics, that, being hidden to us, could not be implemented within BCP.

3.6 Conclusion

In this Chapter, we investigated the adjacent-only version of the quadratic minimum spanning tree problem. We discussed how Gilmore-Lawler based approaches fail to provide good lower bounds for the problem. As an attempt to obtain better bounds, a formulation based on the stars of G was introduced. We presented two strategies to compute the LP relaxation bounds provided by that formulation. The first is a combined row-and-column generation procedure while the second is a cutting plane method, based on a projection of the proposed model. Two exact solution approaches were implemented and tested: a branch-and-cut-and-price algorithm based on the first and a branch-and-cut procedure based on the second. Computational experiments conducted here indicate that our lower bounds are much stronger than previous bounds in the literature and the lower bounds presented in Chapter 2. As a consequence, the exact methods introduced in this chapter managed to solve instances with up to 50 vertices, while the best performing algorithm in Chapter 2 could not solve instances with more than 20 vertices.

Instance			F^*				F_1			F_2 $(K=2)$			F_{AX92}		
					$t(s)$										
n	id	ub^*	lb	$gap(\%)$	RCG	CP	lb	$gap(\%)$	$t(s)$	lb	$gap(\%)$	$t(s)$	lb	$gap(\%)$	$t(s)$
$n = 15$															
min gap	4	271	271	0	3.1	3.4	162.6	39.9	1.1	228.5	15.6	47.4	111.8	58.7	0
avg. gap	1-10	275.7	270	2	3.1	3.6	171.7	37.7	1.2	229.3	16.8	47.3	125.1	54.6	0
max gap	5	206	196	4.8	2.5	2.8	105.3	48.8	1.1	163.3	20.7	49.2	76.2	63	0
$n = 20$															
min gap	5	332	332	0	12.9	15.6	170.7	48.5	3.4	260.2	21.6	208.5	112.6	66	0
avg. gap	1-10	357	348.2	2.4	14.4	18	213.8	40	3.4	277.1	22.3	210.5	153.6	56.9	0
max gap	9	322	304	5.5	13.5	16	165.4	48.6	3.3	244	24.1	208.1	110	65.8	0
$n = 30$															
min gap	10	418	415	0.7	133.6	242.8	226.9	45.6	14.5	290.2	30.5	1776.5	156.4	62.5	0
avg. gap	1-10	434.9	423.3	2.6	137	256.5	223.4	48.6	14.5	299.4	31.1	1829.1	134.4	69	0
max gap	3	464	441.9	4.7	152.7	291.1	222.5	52	14.2	320.9	30.8	1823.4	125.4	72.9	0
$n = 40$															
min gap	6	466	454.9	2.3	574.7	1707.5	207.7	55.4	70.3	303.8	34.8	8368.4	105.7	77.3	0.1
avg. gap	1-10	486.7	468.5	3.7	663.1	1868.2	229	52.9	70.8	315.7	35.1	8174.1	130.9	73	0.1
max gap	4	505	472.7	6.3	661.5	1942.6	188.7	62.6	71.8	300.1	40.5	8276.2	103.5	79.4	0.1
$n = 50$															
min gap	9	498	485.7	2.4	2971.2	11189.5	214.1	56.9	185.6	320.1	35.7	32767.9	105.9	78.7	0.2
avg. gap	1-10	547	521.4	4.6	2951.4	13311.2	234	57.2	187.3	336.4	38.5	32922.3	128.8	76.4	0.3
max gap	3	589	550.1	6.5	3529.8	15280.7	262.8	55.3	187.6	361.9	38.5	33899	139.9	76.2	0.3

Table 3.1. Lower bound comparison.

Instance			*BCP*					*BC*					BB_1				
n	*id*	ub^*	*ub*	*lb*	*gap*(%)	*nodes*	$t(s)$	*ub*	*lb*	*gap*(%)	*nodes*	$t(s)$	*ub*	*lb*	*gap*(%)	*nodes*	$t(s)$
$n = 15$																	
min gap	4	271	271	271	0	1	3	271	271	0	0	3.2	271	271	0	385	12
avg. gap	1-10	275.7	275.7	275.7	0	3	3.3	275.7	275.7	0	3.2	3.4	275.7	275.7	0	446.2	11.9
max gap	5	206	206	206	0	5	2.9	206	206	0	11	2.8	206	206	0	453	13.4
$n = 20$																	
min gap	5	332	332	332	0	1	13.4	332	332	0	0	13.8	332	332	0	19583	1213
avg. gap	1-10	357	357	357	0	5.6	17.4	357	357	0	14.7	16.2	357	357	0	12731.3	752.2
max gap	9	322	322	322	0	7	18.4	322	322	0	40	16.4	322	322	0	19963	1215.3
$n = 30$																	
min gap	10	418	418	418	0	3	149.8	418	418	0	0	126.7	432	226.4	47.5	18011	36000*
avg. gap	1-10	434.9	434.9	434.9	0	13.4	332.3	434.9	434.9	0	43.7	169.9	452.3	223.2	50.6	18111	36000*
max gap	3	464	464	464	0	29	778.4	464	464	0	112	194.6	486	222.7	54.1	18011	36000*
$n = 40$																	
min gap	6	466	466	466	0	9	1784.38	466	466	0	31	817.72	-	-	-	-	-
avg. gap	1-10	486.7	486.7	486.7	0	25.8	5432.1	486.7	486.7	0	126.9	1081.5	-	-	-	-	-
max gap	4	505	505	505	0	75	17253.3	505	505	0	591	1577	-	-	-	-	-
$n = 50$																	
min gap	9	498	498	498	0	9	12054.9	498	498	0	34	4239.2	-	-	-	-	-
avg. gap	1-10	547	559.7	522.6	6.6	25.8	33605.4	547	547	0	638.7	7921.6	-	-	-	-	-
max gap	3	589	589	550.1	6.5	32	36000*	589	589	0	1042	9974.1	-	-	-	-	-

Table 3.2. Branch-and-bound comparison

Chapter 4

Future Work and the Quadratic Assignment Problem

In Section 2.1.1, we introduced an RLT based formulation for QMSTP and used it to obtain Lagrangian relaxation bounds. Since only one round of reformulation-linearization was applied, that type of formulation is referred to as a level-1 RLT formulation. The formulation could be strengthened by subsequent applications of RLT, yielding levels 2, 3, and so on. Higher RLT levels have been successfully applied to QAP [24, 2, 23].

The QMSTP Lagrangian relaxation scheme described in Section 2.1.1 has to deal with $O(m^2) = O(n^4)$ Lagrangian multipliers, attached to constraints (2.10), while Lagrangian subproblems are solved by computing $O(m)$ minimum spanning trees, at a computational complexity cost of $O(m \log n)$ each. After the application of a new round of RLT, the same type of Lagrangian scheme would still apply. However, instead of $O(m^2)$, we would have $O(m^3) = O(n^6)$ dualized constraints, while the resolution of Lagrangian subproblems would require computing $O(m^2)$ minimum spanning trees, resulting in a total complexity of $O(m^3 \log n) = O(n^6 \log n)$ for solving each Lagrangian subproblem.

While for the QAP case a specialized dual ascent method for finding good dual multipliers [24] does exist, we are not aware of such a procedure for QMSTP. One possible line of research would be investigating the application of further RLT levels to QMSTP and how to go around the high computational complexity implied by them. Parallel programming might be one possible approach.

Results obtained in Section 2.1.2 can be easily extended to other constrained 0-1 quadratic problems whose solutions satisfy a cardinality constraint similar to (2.1). Two examples are QAP and the p-dispersion problem (pDP). pDP asks for a binary vector that minimizes a quadratic cost function, such that exactly p entries of the vector have value one [39]. We show next how our results can be extended to QAP. The pDP case follows in a similar fashion. In order to highlight the similarities between the two approaches, the notation we employ is similar to the notation we used for QMSTP.

Given a bipartite graph $G = (V_1 \cup V_2, E)$, with $|V_1| = |V_2| = n$ and $|E| = m$, and a real-valued matrix $Q = (q_{ij})_{i,j \in E}$, the quadratic assignment problem [10] consists of finding, among all characteristic vectors of perfect matchings of G, the vector $\mathbf{x}$ that minimizes the quadratic function

$$\sum_{i,j \in E} q_{ij} x_i x_j.$$

Consider binary variables $\mathbf{x} = (x_i)_{i \in E}$. It is known [9] that the assignment/bipartite matching polytope, which we denote by X, is completely described by

$$\sum_{i \in \delta(v)} x_i = 1, \quad v \in V_1 \cup V_2. \tag{4.1}$$

$$x_i \geq 0, \qquad i \in E. \tag{4.2}$$

A 0-1 quadratic programming formulation for QAP is thus given by

$$\min\{\sum_{i,j \in E} q_{ij} x_i x_j : \mathbf{x} \in X \cap \mathbb{B}^m\}.$$

After introducing linearization variables $\mathbf{y} = (y_{ij})_{i,j \in E}$ and applying RLT to this formulation, we obtain the following IP formulation, originally introduced by Adams and Johnson [3]

$$F_1 : \quad \min\{\sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}) \in P_1 \cap \mathbb{B}^{m+m^2}\}.$$

P_1 denotes the polyhedron represented by

$$\mathbf{x} \in X, \tag{4.3}$$

$$\mathbf{y}_i \in (X_i \times x_i), \qquad i \in E, \tag{4.4}$$

$$y_{ij} = y_{ji}, \qquad i < j \in E, \tag{4.5}$$

where $\mathbf{y}_i = (y_{ij})_{j \in E}$, $X_i = X \cap \{\mathbf{x} \in \mathbb{R}^m : x_i = 1\}$, and $X_i \times x_i$ refers to the set of points in X_i multiplied by x_i, i.e., points $\mathbf{y}_i$ that satisfy

$$\sum_{j \in \delta(v)} y_{ij} = x_i, \quad v \in V_1 \cup V_2. \tag{4.6}$$

$$y_{ii} = x_i, \tag{4.7}$$

$$y_{ij} \geq 0, \qquad j \in E. \tag{4.8}$$

The relaxation and Lagrangian dualization of (4.5) gives the Lagrangian subproblem

$$F_1'(\theta): \qquad L_1'(\theta) = \min\{\sum_{i,j\in E} q_{ij}' y_{ij} : (\mathbf{x},\mathbf{y}) \in P_1' \cap \mathbb{B}^{m+m^2}\},$$

where $\theta = (\theta_{ij})_{i<j\in E}$ are dual multipliers associated to constraints (4.5) ($\theta_{ij} = -\theta_{ji}$ for all $i > j \in E$). The coefficients q_{ij}' are given by $q_{ij}' = q_{ij} + \theta_{ij}$ for all pairs $i \neq j \in E$ and $q_{ii}' = q_{ii}$ for all $i \in E$. Problem F_1' can be solved by the Gilmore-Lawler procedure [22, 28], presented in Algorithm 8.

Algorithm 8:
Input: A QAP instance given by $G = (V_1 \cup V_2, E)$ and $Q \in \mathbb{R}^{m^2}$. A set of Lagrangian multipliers $\overline{\theta} \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: A solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ for $F_1'(\overline{\theta})$.

1. For all $i \in E$, solve the assignment problem

$$\overline{q}_i = \min\{\sum_{j\in E} q_{ij}' y_{ij} : \mathbf{y}_i \in X_i \cap \mathbb{B}^m\},$$

and denote the minimizing vector by $\widetilde{\mathbf{y}}_i$.

2. Solve the assignment problem

$$\overline{q}_0 = \min\{\sum_{i\in E} \overline{q}_i x_i : \mathbf{x} \in X \cap \mathbb{B}^m\}.$$

and denote by $\widetilde{\mathbf{x}}$ the minimizing vector.

3. Obtain a solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \in \mathbb{B}^{m+m^2}$ for F_1' by letting $\overline{\mathbf{x}} = \widetilde{\mathbf{x}}$ and $\overline{\mathbf{y}}_i = \overline{x}_i \widetilde{\mathbf{y}}_i$, for all $i \in E$.

It is not difficult to see that the LP relaxation bounds $Z(F_1)$ are implied by the Lagrangian dual

$$DF_1: \qquad L_1'^{*} = \max\{L_1'(\theta) : \theta \in \mathbb{R}^{\frac{m(m-1)}{2}}\}.$$

An effective dual ascent algorithm for solving DF_1 was presented in Hahn and Grant [24]. That algorithm, combined with BB, represents one of the most successful approaches for QAP.

As we did for the QMSTP case, we wish to generalize the Gilmore-Lawler bound. To that aim, let $K > 0$ be a factor of n. Denote by K-matching any matching of G with exactly K edges. Let E^K be the collection of all K-matchings of G. Since K divides n, any matching can be decomposed into n/K K-matchings. The generalization of the Gilmore-Lawer bound is then computed as follows. For each $H \in E^K$ let $\overline{q}_H$ be the objective value of the best assignment implied by costs $(\sum_{i \in H} q_{ij})_{j \in E}$ that have all edges of H included in the solution. Find a set $F \subseteq E^K$ such that F induces an assignment of G and $\sum_{H \in F} \overline{q}_H$ is minimal. The cost of this assignment gives a lower bound for QAP. Observe that the cost we are paying for each edge in this assignment is at least $\overline{q}_i$, since it depends not only on the interaction costs implied by i, but also on the costs implied by all edges in a K-matching. Therefore, the bound is at least as strong as the simple Gilmore-Lawler bound.

As in the QMSTP case, we show that this bound results from Lagrangian relaxation applied to a QAP formulation. Consider variables $\mathbf{s} = (s_H)_{H \in E^K}$ and $\mathbf{t} = (t_{Hi})_{H \in E^K, i \in E}$. Let $\mathbf{t}_H = (t_{Hi})_{i \in E}$, for all $H \in E^K$, and let $E_i^K = \{H \in E^K : i \in H\}$, for all $i \in E$. For all $H \in E^K$ define X_H as $X \cap \{x \in \mathbb{R}^m : x_i = 1, \forall i \in H\}$. QAP can be formulated as follows:

$$F_2 : \qquad \min\{\sum_{i,j \in E} q_{ij} y_{ij} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P_2 \cap \mathbb{B}^{m+m^2+o+om}\},$$

where $o = |E^K|$ and P_2 is given by

$$\mathbf{x} \in X, \tag{4.9}$$

$$x_i = \sum_{H \in E_i^K} s_H, \qquad i \in E, \tag{4.10}$$

$$\mathbf{y}_i = \sum_{H \in E_i^K} \mathbf{t}_H, \qquad i \in E, \tag{4.11}$$

$$\mathbf{t}_H \in (X_H \times s_H), \qquad H \in E^K, \tag{4.12}$$

$$y_{ij} = y_{ji}, \qquad i < j \in E. \tag{4.13}$$

As in the QMSTP case, we only need to grant that at least one decomposition exists for each assignment. Thus, not all elements of E^K need to be considered. Let $P = \{P^1, \dots, P^{\frac{n}{K}}\}$ be a partition of V_1 into sets of K vertices each. Let $F = \bigcup_{k=1}^{\frac{n}{K}} F^k \subseteq E^K$, where $F^k = \{H \subseteq \delta(P^k) : H \in E^K\}$ is the set of all K-matchings that go out of P^k. Observe that F is composed of K-matchings that originate in some P^k, $1 \leq k \leq \frac{n}{K}$. It is not difficult to see that only the elements in F are needed to construct any assignment of G.

In order to obtain the generalization of the Gilmore-Lawler bound, explained in the beginning of this section, we consider the Lagrangian relaxation of constraints (4.5), with multipliers $\theta = (\theta_{ij})_{i<j\in E}$ ($\theta_{ij} = -\theta_{ji}$ for $i > j \in E$). We obtain the problem

$$F_2'(\theta): \qquad L_2'(\theta) = \min\{\sum_{i,j\in E} q'_{ij}y_{ij} : (\mathbf{x},\mathbf{y}) \in P_2' \cap \mathbb{B}^{m+m^2}\},$$

where P_2' is defined by (4.9)-(4.12) and the coefficients q'_{ij} are given by $q'_{ij} = q_{ij} + \theta_{ij}$ for all pairs $i \neq j \in E$ and $q'_{ii} = q_{ii}$ for all $i \in E$.

Given a choice $\overline{\theta}$ of Lagrangian multipliers, observe that if $s_H = 1$ for some $H \in E^K$ in an optimal solution for this problem, then $\mathbf{t}_H$ is the characteristic vector of an assignment that minimizes

$$\overline{q}_H = \min\{\sum_{i\in H}\sum_{j\in E} q'_{ij}t_{Hj} : \mathbf{t}_H \in X_H \cap \mathbb{B}^m\}.$$

Therefore, $F_2'(\overline{\theta})$ can be solved with the resolution of

$$\overline{q}_0 = \min\{\sum_{H\in E^K} \overline{q}_H s_H : \mathbf{x} \in X; x_i = \sum_{H\in E_i^K} s_H, \forall i \in E; (\mathbf{x},\mathbf{s}) \in \mathbb{B}^{m+o}\}, \tag{4.14}$$

and the appropriate adjustment of $(\mathbf{y},\mathbf{t}) \in \mathbb{B}^{m^2+om}$.

Assuming that E^K is replaced by F, once a vertex in $v \in P^k$, $1 \leq k \leq \frac{n}{K}$, is covered by some $H \in E^K$, all vertices in P^k are also covered by the same K-matching. Thus, each P^k can be effectively considered as a super vertex. Using this idea, it is possible to polynomially reduce the 3-dimensional matching problem [19] to (4.14), showing thus that problem (4.14) is NP-Hard.

If the constraints

$$\sum_{i\in\delta(v)} x_i = 1, \qquad v \in V_2,$$

are subsequently relaxed and dualized in Lagrangian fashion with unconstrained multipliers $\lambda = (\lambda_v)_{v\in V_2}$ attached to them, we obtain

$$F_2''(\theta,\lambda): \qquad L_2''(\theta,\lambda) = C + \min\{\sum_{i,j\in E} q'_{ij}y_{ij} : (\mathbf{x},\mathbf{y}) \in P_2'' \cap \mathbb{B}^{m+m^2}\},$$

and the corresponding Lagrangian dual

$$DF_2: \qquad L_2''^{*} = \max\{L_2''(\theta,\lambda) : (\theta,\lambda) \in \mathbb{R}^{\frac{m(m-1)}{2}+|V_2|}\},$$

where P''_2 is given by (4.10)-(4.12), (4.2) and $\sum_{i\in\delta(v)} x_i = 1$, for all $v \in V_1$. Coefficients q'_{ij} are given by $q'_{ij} = q_{ij} + \theta_{ij}$ for all pairs $i \neq j \in E$ and $q'_{ii} = q_{ii} + \lambda_v$ for all $i = \{u, v\} \in E$, $v \in V_2$, and C is given by $C = -\sum_{v\in V_2} \lambda_v$.

$F''_2(\theta, \lambda)$ is easy to solve for any K, since it suffices to select the best $H \in F^k$ for each P^k, $1 \leq k \leq \frac{n}{K}$.

In the future, we intend to investigate this QAP lower bounding procedure. For small values of K, it could offer a better lower bound/computational time trade-off than RLT level 2 and 3, which are very computationally expensive.

Chapter 5

Conclusion

In this thesis, we addressed the quadratic minimum spanning tree problem and one of its particular cases, the adjacent-only quadratic minimum spanning tree problem.

For the general case, we initially proposed a formulation based on RLT. We showed that existing formulations in the literature are dominated by this RLT formulation. In order to compute its LP relaxation bounds, we presented a Lagrangian relaxation scheme, Lag_1, which was then embedded in a parallel BB algorithm, BB_1.

The RLT formulation was then generalized into a hierarchy of formulations. We showed that the strength of the generalized formulation grows according to a parameter K that controls the size of the modeling entities used in it. In order to obtain a practical lower bounding scheme, we studied many of its relaxations. Two Lagrangian relaxation schemes were developed, one of them, Lag_2, for a particular value of K, and the other, Lag_3, for general values of K. The former performed better in practice, and was used for developing a second parallel BB algorithm, BB_2.

The generalization we proposed for the QMSTP RLT approach can be extended to other 0-1 quadratic problems whose feasible solutions satisfy a cardinality constraint. We demonstrated in detail how this extension can be carried out for the quadratic assignment problem.

In order to evaluate our QMSTP lower bounding procedures and exact algorithms, we performed computational experiments using instances from the literature. Strong lower bounds were obtained by our procedures compared to previous approaches. While Lag_2 and Lag_3 obtained the best lower bounds, Lag_1 provided a better lower bound/computational time tradeoff. Consequently, with the help of parallel programming, BB_1 was able to solve many previously unsolved instances, some with up to 50 vertices.

We observed that Gilmore-Lawler based lower bounding procedures, such as the ones we presented for the general case, do not perform so well in the AQMSTP case. Motivated by that, we introduced an AQMSTP formulation that used the stars of the graph to address the quadratic cost structure of the problem. In order to compute its LP relaxation lower bounds, we proposed two different approaches. One of them, RCG, was based on row-and-column generation. The other, CP, resulted after we projected the variables associated to stars out of the formulation and applied a cutting plane algorithm. Based on these two lower bounding procedures, we developed two BB algorithms, BCP, based on RCG, and BC, based on CP.

Computational results were performed to compare our AQMSTP specific approach to RLT based ones, devised by ourselves and from the literature, for the general case. Lower bounds obtained by the specific approach were much stronger. Because of that, while the best performing algorithm for the general case, BB_1, was able to solve instances with at most 20 vertices, the specific BB algorithms managed to solve instances with up to 50 vertices. The approach based on projection proved better than the row-and-column generation approach: BC was able to solve all instances in our test bed, including 10 with 50 vertices, while only one instance with 50 vertices was solved by BCP.

Appendix A

Separation Algorithms

A.1 Separation of (2.5)

In this section we present a polynomial time algorithm for the separation problem of inequalities (2.5):
Given $i \in E$, $\overline{x}_i \in \mathbb{R}$, and $\overline{\mathbf{y}}_i \in \mathbb{R}^m$, find a set $\overline{S} \subset V$, $|\overline{S}| \geq 2$, for which

$$\sum_{j \in E(S)} \overline{y}_{ij} \leq (|S|-1)\overline{x}_i, \qquad S \subset V, |S| \geq 2$$

is violated or prove that no such set exists.

Padberg and Wolsey [37] give a polynomial time algorithm that solves the separation problem of (2.2), which we now adapt for the separation problem of (2.5). Our algorithm finds $\overline{S}$ such that

$$\overline{S} \in \arg \min_{S \subset V, |S| \geq 1} \{|S|\overline{x}_i - \sum_{j \in E(S)} \overline{y}_{ij}\}. \tag{A.1}$$

Clearly, there is a violated inequality (2.5) if and only if the minimum is smaller than $\overline{x}_i$.

Algorithm 6
Input: $G = (V, E)$, an edge $i \in E$, $\overline{x}_i \in \mathbb{R}$, and $\overline{\mathbf{y}}_i \in \mathbb{R}^m$.
Output: $\overline{S}$ satisfying (A.1).

1. Create a directed graph $\widehat{G} = (\widehat{V}, \widehat{A})$ with $\widehat{V} = V \cup \{s, t\}$ and $\widehat{A} = \{(u, v), (v, u) : \{u, v\} \in E\} \cup \{(s, u), (u, t) : u \in V\}$.

2. For each $j = \{u, v\} \in E$, assign capacities $c_{uv} = c_{vu} = \frac{1}{2}\overline{y}_{ij}$ to the arcs (u, v) and (v, u) in $\widehat{A}$.
3. For each $u \in V$, assign capacities $c_{su} = \max\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\}$ and $c_{ut} = \max\{\overline{x}_i - \frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij}, 0\}$ to the arcs (s, u) and (u, t) in $\widehat{A}$.
4. Find the minimum capacity cut $(\overline{S}\cup\{s\}, V\cup\{t\}\setminus\{\overline{S}\})$ of $\widehat{G}$. $\overline{S}$ minimizes (A.1).

Assuming that G is connected, Steps 1-3 are easily seen to run in $O(E)$ time. Step 4 solves $O(n)$ maximum flow problems, which can be performed in $O(nm^2)$ each with the algorithm of Edmonds and Karp [17], for example. Thus, the complexity of Algorithm 6 is $O(n^2m^2)$. Observe, however, that when employed to solve the LP relaxation of F_1 with dynamic generation of cuts, Algorithm 6 has to be applied for each $i \in E$, which gives the total time complexity of $O(n^2m^3)$.

To prove the validity of the procedure, observe that the capacity of any cut $(\widetilde{S} \cup \{s\}, V \cup \{t\} \setminus \{\widetilde{S}\})$ is

$$\sum_{u\in\widetilde{S}} \max\left\{\overline{x}_i - \frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij}, 0\right\} + \sum_{u\in V\setminus\widetilde{S}} \max\left\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\right\} + \frac{1}{2}\sum_{\substack{i=\{u,v\}\in E,\\ u\in\widetilde{S}, v\notin\widetilde{S}}} \overline{y}_{ij}$$

$$\begin{aligned}
&= \sum_{u\in\widetilde{S}} \left(\max\{\overline{x}_i - \frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij}, 0\} - \max\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\}\right) \\
&+ \sum_{u\in V} \max\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\} + \frac{1}{2}\sum_{\substack{i=\{u,v\}\in E,\\ u\in\widetilde{S}, v\notin\widetilde{S}}} \overline{y}_{ij} \\
&= |\widetilde{S}|\overline{x}_i - \sum_{j\in E(\widetilde{S})} \overline{y}_{ij} + \sum_{u\in V} \max\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\}.
\end{aligned}$$

Since $\sum_{u\in V} \max\{\frac{1}{2}\sum_{j\in\delta(u)} \overline{y}_{ij} - \overline{x}_i, 0\}$ is constant, the set that yields the cut of minimum capacity of $\widehat{G}$ is the set $\overline{S}$ that minimizes (A.1).

A.2 Separation of (2.11)

We now discuss the separation of constraints (2.11). Their associated separation problem can be stated as:

Given $(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \in \mathbb{R}^{m+m^2}$, find an edge $i \in E$ and a set $\overline{S} \subset V$, $|\overline{S}| \geq 2$, for which

$$\sum_{j \in E(S)} (\overline{x}_j - \overline{y}_{ij}) \leq (|S| - 1)(1 - \overline{x}_i), \qquad i \in E, S \subset V, |S| \geq 2.$$

is violated or prove that no such set exists.

Again, we adapt the algorithm of Padberg and Wolsey [37]. We consider one edge $i \in E$ at a time. It is assumed that $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ satisfies $y_{ij} \leq x_j$ for all $j \in E$, which is the case when using dynamic cut generation to strengthen the LP relaxation of formulation F_1, for example.

The algorithm below will find $\overline{S}$ such that

$$\overline{S} \in \arg \min_{S \subset V, |S| \geq 1} \{|S|(1 - \overline{x}_i) - \sum_{j \in E(S)} (\overline{x}_j - \overline{y}_{ij})\}. \tag{A.2}$$

It is clear that there is a violated inequality (2.11) if and only if the minimum is smaller than $1 - \overline{x}_i$.

Algorithm 7
Input: $G = (V, E)$, $(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \in \mathbb{R}^{m+m^2}$, and $i \in E$.
Output: $\overline{S}$ satisfying (A.2).

1. Create a directed graph $\widehat{G} = (\widehat{V}, \widehat{A})$ with $\widehat{V} = V \cup \{s, t\}$ and $\widehat{A} = \{(u, v), (v, u) : \{u, v\} \in E\} \cup \{(s, u), (u, t) : u \in V\}$.

2. For each $j = \{u, v\} \in E$, assign capacities $c_{u,v} = c_{v,u} = \frac{1}{2}(\overline{x}_j - \overline{y}_{ij})$ to the arcs (u, v) and (v, u) in $\widehat{A}$.

3. For each $u \in V$, assign capacities $c_{s,u} = \max\{\frac{1}{2} \sum_{j \in \delta(u)} (\overline{x}_j - \overline{y}_{ij}) - (1 - \overline{x}_i), 0\}$ and $c_{u,t} = \max\{(1 - \overline{x}_i) - \frac{1}{2} \sum_{j \in \delta(u)} (\overline{x}_j - \overline{y}_{ij}), 0\}$ to the arcs (s, u) and (u, t) in $\widehat{A}$.

4. Find the cut $(\overline{S} \cup \{s\}, V \cup \{t\} \setminus \{\overline{S}\})$ of minimum capacity of $\widehat{G}$. $\overline{S}$ minimizes (A.2).

Algorithm 7 has the same complexity of Algorithm 6. Considering again the application of the algorithm for each $i \in E$, we obtain a total complexity of $O(n^2 m^3)$. The proof of correctness is similar to that of Algorithm 6 and is omitted.

Appendix B

Proofs

B.1 Proof of Proposition 2.7

In order to prove Proposition 2.7 we need a series of results, which are given next.

Problems F_2' and (2.24) are essentially the same. However, in F_2', the cost of employing a K-forest in the solution is at least the cost of its best interaction tree, given by (2.23), while in (2.24) the cost is part of the input. Algorithm 2 gives a polynomial reduction from F_2' to (2.24). We will now show the converse, that (2.24) can be polynomially reduced to F_2', proving thus that the two problems are equivalent. With this, we can use the simpler form (2.24) in order to derive computational complexity results for F_2'.

Lemma B.1. *Problem* (2.24) *can be polynomially reduced to* F_2'.

Proof. Assume $K \geq 2$. Given an instance for (2.24), defined in terms of a graph $\overline{G} = (\overline{V}, \overline{E})$, a set of K-forests $\overline{F} \subseteq \overline{E}^K$, and costs $\overline{Q} = (\overline{q}_H)_{H \in \overline{F}}$, we will construct a instance for F_2', defined in terms of a graph $G = (V, E)$, K-forests $F \subseteq E^K$, and matrix $Q = (q_{ij})_{i,j \in E}$. The polynomial construction will be such that for any solution for the input problem, there will be a solution with the same cost for the transformed problem, and, for any solution for the transformed problem, there will be a solution for the input problem with equal or smaller cost. This will prove the statement.

Initially, consider $V = \overline{V}$, $E = \overline{E}$, and $F = \overline{F}$. Select a vertex $v \in \overline{V}$. For all $H \in \overline{F}$, add K new vertices $u_H^1, \ldots, u_H^K$ to V and add $K-1$ new edges to E: $\{v, u_H^1\}$, and $\{u_H^i, u_H^{i+1}\}$, $1 \leq i < K-1$. Consider $\overline{F}$ as an ordered set $(H_1, ..., H_{|\overline{F}|})$. For all H_k, $1 \leq k \leq |\overline{F}|$, add to E edges $\phi_{H_k} = \{u_{H_k}^{K-1}, u_{H_1}^K\}$, edges $\{u_{H_k}^{K-1}, u_{H_k}^K\}$, for $k > 1$, and $\{u_{H_k}^{K-1}, u_{H_{k+1}}^K\}$, for $k < |\overline{F}|$. Add to F the sets $\{\{v, u_{H_k}^1\}, \ldots, \{u_{H_k}^i, u_{H_k}^{i+1}\}, \ldots, \{u_{H_k}^{K-1}, u_{H_1}^K\}\}$, the sets $\{\{v, u_{H_k}^1\}, \ldots, \{u_{H_k}^i, u_{H_k}^{i+1}\}, \ldots, \{u_{H_k}^{K-1}, u_{H_k}^K\}\}$, for $k > 1$, and $\{\{v, u^1_{H_k}\}, \ldots, \{u^i_{H_k}, u^{i+1}_{H_k}\}, \ldots, \{u^{K-1}_{H_k}, u^K_{H_{k+1}}\}\}$, for $k < |\overline{F}|$. After that, $O(K|\overline{F}|)$ vertices and edges will have been added to V and E, respectively, and $O(|\overline{F}|)$ new subsets will have been added to F. Now, define matrix $Q = (q_{ij})_{i,j \in E}$ as follows. For all $H \in \overline{F}$, fix an edge $i \in H$, define $q_{i\phi_H} = \overline{q}_H + (K-1)M$, where M is a sufficiently large number, $q_{j\phi_H} = -M$, $j \neq i \in H$, and $q_{j\phi_I} = KM$ if $j \notin I$, $j \in H$, $I \in \overline{F}$. All other entries of Q are set to zero. This step has time complexity of $O((|\overline{E}| + K|\overline{F}|)^2)$. Therefore, the whole transformation is carried out in polynomial time.

Let $F' \subset F$ be the set of K-forest-inducing edges used in a feasible solution for F_2'. Then, $F' \cap \overline{F}$ is a feasible solution for (2.24). In case $H \in F' \cap \overline{F}$ is used in the solution for the transformed problem, the cost implied by its selection will be not smaller than (2.23). To see this, observe that in a interaction tree for H, all the vertices of G, with exception of $u^K_{H_1}$, can be reached without any cost. To reach $u^K_{H_1}$, if ϕ_I, $I \in \overline{F}$, is used and if $|H \cap I| = 0$, a cost K^2M will be implied. If $|H \cap I| = a$, $0 < a < K$, the smallest cost possible is $(K-a)KM - aM > M$. If $|H \cap I| = K$, a cost $\overline{q}_H$ will be implied, which is the minimum possible. Thus, the best interaction tree for H has cost $\overline{q}_H$. Moreover, the selection of $H \in F \setminus \overline{F}$ never implies in any costs. Thus, the cost of this solution for F_2' is not smaller than the cost of the implied solution for (2.24). On the other hand, if a set of K-forest-inducing edges $\overline{F}' \subseteq \overline{F}$ is used in a solution for the input problem, then there is a solution for the transformed problem with K-forest-inducing sets $\overline{F}' \cup F'$, $F' \in F \setminus \overline{F}$. From the discussion above, an interaction tree with cost $\overline{q}_H$, but not better, can be selected for each $H \in \overline{F}$ in the transformed problem, while interaction trees for $H \in F'$ have zero cost. Thus, the solution of the transformed problem has the same cost of the solution of the input problem. The proof is complete. □

The following definitions and results are needed to derive the complexity results for F_2' and F_2''.

A hypergraph $H = (M, S)$ is a pair consisting of a set M and a collection S of non-empty sets of elements of M. The elements of M are denoted vertices, the elements of S are denoted edges. A hypergraph is K-uniform if all of its edges have K vertices each. A matching of a hypergraph $H = (M, S)$ is a set of edges $S' \subseteq S$ such that no vertex of H appears in more than one edge of S'. A matching S' of a hypergraph H is perfect if every vertex of H appear in one edge of S'.

Garey and Johnson [19] proved that

Lemma B.2. *Deciding whether a 3-uniform hypergraph admits a perfect matching is NP-Complete.* □

Thus, we have:

Lemma B.3. *Deciding whether a K-uniform hypergraph admits a perfect matching is NP-Complete for any $K \geq 3$.*

Proof. First, we assume that the number of vertices in the hypergraph is a multiple of K, for otherwise the problem is trivial. The proof is by induction, with $K = 3$ as the base case. Assume that the statement is valid for some $K \geq 3$. We will show that if the decision problem can be solved in polynomial time for $K+1$, than it can be solved in polynomial time for K. What proves that the problem is NP-Complete for $K+1$.

Consider a K-uniform hypergraph $H = (M, S)$. Create a $(K+1)$-uniform hypergraph $\overline{H} = (\overline{M}, \overline{S})$ as follows. Let $M' = \{v_1, \dots, v_{|M|/K}\}$ be a set of new vertices and let $\overline{M} = M \cup M'$. Let $\overline{S} = \{i \cup \{v_k\} : i \in S, 1 \leq k \leq |M|/K\}$. The construction is clearly polynomial. Now, let $S' = \{i_1, \dots, i_{|M|/K}\}$ be a perfect matching of H, then $\overline{S}' = \{i_k \cup \{v_k\} : 1 \leq k \leq |M|/K\}$ is a perfect matching of $\overline{H}$. Conversely, let $\overline{S}'$ be a perfect matching of $\overline{H}$, then $S' = \{i \cap M : i \in \overline{S}'\}$ is a perfect matching of H. Thus, a polynomial time algorithm for the decision problem for $K+1$ implies in a polynomial time algorithm for K and the proof is complete. □

The following definition of spanning trees in hypergraphs was given in [5]. A hypergraph $H = (M, S)$ is connected if, for any pair of distinct vertices u and v, there is a sequence $(i_1, \dots, i_k)$ of edges, such that $u \in i_1, v \in i_k$, and $i_a \cap i_{a+1} \neq \emptyset$, for all $1 \leq a < k$. A hypergraph is Berge-acyclic [18] if there does not exist any sequence $(i_1, v_1, i_2, v_2, \dots, i_k, v_k, i_{k+1})$, where $i_a \in S$ for all $1 \leq a \leq k+1$ and $v_a \in M$ for all $1 \leq a \leq k$, such that $v_a \neq v_b$ and $i_a \neq i_b$ for all $1 \leq a < b \leq k$, $i_1 = i_{k+1}$, $k \geq 2$, and $v_a \in i_a \cap i_{a+1}$ for all $1 \leq a \leq k$. A hypergraph $H' = (M', S')$ is a subhypergraph of a hypergraph $H = (M, S)$ if $M' \subseteq M$ and $S' \subseteq S$. A spanning tree of a hypergraph H is a connected and Berge-acyclic subhypergraph of H.

Lemma B.4. *Deciding whether a K-uniform hypergraph admits a spanning tree is NP-Complete for $K \geq 4$.*

Proof. Let $K \geq 4$ and let $H = (M, S)$ be a $(K-1)$-uniform hypergraph. Create a K-uniform hypergraph $\overline{H} = (\overline{M}, \overline{S})$ as follows. Let $\overline{M} = M \cup \{v\}$, where v is a new vertex. Let $\overline{S} = \{i \cup \{v\} : i \in S\}$. Clearly, the construction of $\overline{H}$ is polynomial. Now, let S' be a perfect matching of H, then, $\overline{S}' = \{i \cup \{v\} : i \in S'\}$ is a spanning tree of $\overline{H}$. On the other hand, let $\overline{S}'$ be a spanning tree of $\overline{H}$, then, $S' = \{i \setminus \{v\} : i \in \overline{S}'\}$ is a perfect matching of H. Thus, a polynomial time algorithm for determining whether a K-uniform hypergraph admits a spanning tree implies in a polynomial time algorithm for determining whether a $(K-1)$-uniform hypergraph admits a perfect matching. □

Lemma B.5. *Given a graph $G = (V, E)$ and a set F of K-forests of G, $K \geq 3$, deciding whether there is a $F' \subseteq F$ that induces a spanning tree of G is NP-Complete.*

Proof. Let $H = (M, S)$ be a $(K + 1)$-uniform hypergraph. For all $i \in S$, define $\phi(i)$ as the set of edges of any spanning tree for a (2-uniform) complete graph defined over the vertices of i. Create a (2-uniform) graph $G = (V, E)$ as follows. Let $V = M$ and $E = \cup_{i \in S} \phi(i)$. Define the set of K-forests $F = \{\phi(i) : i \in S\}$. Observe that the construction of G and F is polynomial.

Now, if $F' \subseteq F$ induces a spanning tree of G, then, $S' = \{i \in S : \phi(i) \in F'\}$ induces a spanning tree of H. On the other hand, if $S' \subset S$ induces a spanning tree of H, then, $F' = \{\phi(i) : i \in S\}$ induces a spanning tree of G. Thus, a polynomial time algorithm for deciding whether a spanning tree of G can be created from a set of K-forests implies in a polynomial time algorithm for deciding whether a $(K + 1)$-uniform hypergraph admits a spanning tree. Then, by Corollary B.4, this decision problem is NP-complete for $K \geq 3$. □

Corollary B.1. *Problem* (2.24) *is NP-Hard.* □

Finally, we have:

Proposition 2.7. *F_2' is NP-Hard.*

Proof. Follows from Corollary B.1 and Lemma B.1. □

B.2 Proof of Proposition 2.8

In order to prove Proposition 2.8 we need the following lemma.

Lemma B.6. *Problem* (2.25) *can be polynomially reduced to F_2''.*

Proof. Apply the same construction used for the proof of Lemma B.1. □

Lemma B.7. *Given a graph $G = (V, E)$ and a set F of K-forests of G, $K \geq 3$, deciding whether there is a set $F' \subseteq F$ of disjoint K-forests such that $|F'| = |V - 1|/K$ is NP-Complete.*

Proof. The proof is by polynomial reduction from the hypergraph perfect matching decision problem. Let $H = (M, S)$ be a K-uniform hypergraph such that $K \geq 3$ divides M. Assume that $M = u_1, \ldots, u_{|M|}$. We create a graph $G = (V, E)$ as follows. Let $V = \{v_1, \ldots, v_{|M+1|}\}$ and let $E = \{i_k = \{v_k, v_{k+1}\} : 1 \leq k \leq |M|\}$. Observe that there is a bijective mapping from M to E given by $\phi(u_k) = i_k$ and $\phi^{-1}(i_k) = u_k$,

$1 \leq k \leq |M|$. This mapping implies a set of K-forests $F = \{\cup_{u \in j}\{\phi(u)\} : j \in S\}$ for G.

Now, let $F' \subseteq F$ be a set of $(|V| - 1)/K = |M|/K$ disjoint K-forests, then, $S' = \{\cup_{i \in f}\{\phi^{-1}(i)\} : f \in F'\}$ is a perfect matching for H. Conversely, if $S' \subseteq S$ is a perfect matching for H, then, $F' = \{\cup_{u \in j}\{\phi(u)\} : j \in S'\}$ is a set of $(|V| - 1)/K$ $= |M|/K$ disjoint K-forests of G. Thus, a polynomial time algorithm for deciding whether we can find $|V - 1|/K$ disjoint K-forests of G in a set F, $K \geq 3$, implies in a polynomial time algorithm for the hypergraph perfect matching decision problem. The proof is complete. $\square$

Corollary B.2. *Problem* (2.25) *is NP-Hard.* $\square$

Finally, we have:

Proposition 2.8. F_2'' *is NP-Hard.* $\square$

B.3 Proof of Proposition 2.11

Proposition 2.11. P_3'' *is an integral polytope.*

Proof. Assume that $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ is a fractional point of P_3'. Clearly, if $\overline{\mathbf{y}}$ is fractional, then $\overline{\mathbf{t}}$ is fractional. It is also easy to see that if $\overline{\mathbf{t}}$ is fractional, then $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}})$ cannot be integer, otherwise we would be able to write $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ as a convex combination of integer extreme points (see the first part of the proof of Proposition 2.4). Thus, if $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ is fractional, either $\overline{\mathbf{x}}$, or $\overline{\mathbf{s}}$ is fractional.

Assume first that $\overline{\mathbf{s}}$ is fractional but $\overline{\mathbf{x}}$ is integer. Consider some $i \in E$ for which $\overline{\mathbf{s}}_i$ is fractional. Then, since $\overline{x}_i = 1$, we can write $\overline{\mathbf{s}}_i = \sum_{H \in \overline{E}_i^K} \lambda_H \mathbf{e}_H$, where $\mathbf{e}_H$ is a $|E_i^K|$-dimensional binary vector with exactly one 1-valued entry, at the position indexed by H; $\lambda_H > 0$ for all $H \in \overline{E}_i^K$, $\sum_{H \in \overline{E}_i^K} \lambda_H = 1$, and $\overline{E}_i^K \subseteq E_i^K$. Now, for each $H \in \overline{E}_i^K$ let $\mathbf{r}_i^H \in \mathbb{B}^{|E_i^K|m}$. For each $I \in \overline{E}_i^K$, $I \neq H$, let $\mathbf{r}_{iI}^H = \mathbf{0}$ and let $\mathbf{r}_{iH}^H = \overline{\mathbf{t}}_{iH}/\overline{s}_{iH}$. If we set $\mathbf{x} = \overline{\mathbf{x}}$; $\mathbf{y}_j = \overline{\mathbf{y}}_j$, $\mathbf{s}_j = \overline{\mathbf{s}}_j$, and $\mathbf{t}_j = \overline{\mathbf{t}}_j$ for all $j \in E$, $j \neq i$; $\mathbf{s}_i = \mathbf{e}_H$, $\mathbf{t}_i = \mathbf{r}_i^H$, and $\mathbf{y}_i = \mathbf{r}_{iH}^H$, for some $H \in \overline{E}_i^K$, we obtain a feasible point in P_3''.

From the definitions above, we have, for each $H \in \overline{E}_i^K$, $\sum_{I \in \overline{E}_i^K} \lambda_I \mathbf{r}_{iH}^I = \lambda_H \mathbf{r}_{iH}^H = \overline{s}_{iH} \mathbf{r}_{iH}^H = \overline{\mathbf{t}}_{iH}$. Similarly, $\sum_{H \in \overline{E}_i^K} \lambda_H \sum_{I \in \overline{E}_i^K} \mathbf{r}_{iI}^H = \sum_{H \in \overline{E}_i^K} \lambda_H \mathbf{r}_{iH}^H = \sum_{H \in \overline{E}_i^K} \overline{\mathbf{t}}_{iH} = \overline{\mathbf{y}}_i$. Thus, we have that

$$
\begin{aligned}
(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}}) &= (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \ldots, \overline{\mathbf{y}}_i, \ldots \overline{\mathbf{y}}_{j_m}, \overline{\mathbf{s}}_{j_1}, \ldots, \overline{\mathbf{s}}_i, \ldots, \overline{\mathbf{s}}_{j_m}, \overline{\mathbf{t}}_{j_1}, \ldots, \overline{\mathbf{t}}_i, \ldots, \overline{\mathbf{t}}_{j_m}) \\
&= \sum_{H \in \overline{E}_i^K} \lambda_H (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \ldots, \sum_{I \in \overline{E}_i^K} \mathbf{r}_{iI}^H, \ldots \overline{\mathbf{y}}_{j_m}, \overline{\mathbf{s}}_{j_1}, \ldots, \mathbf{e}_H, \ldots, \overline{\mathbf{s}}_{j_m}, \overline{\mathbf{t}}_{j_1}, \ldots, \mathbf{r}_i^H, \ldots, \overline{\mathbf{t}}_{j_m}).
\end{aligned}
$$

Therefore, $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ cannot be an extreme point.

Finally, assume that $\overline{\mathbf{x}}$ is fractional. Since $\overline{\mathbf{x}} \in X$, we have $\overline{\mathbf{x}} = \sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{w}$, where $\mathcal{E}$ is a subset of the extreme points of X, $\lambda^{\mathbf{w}} > 0$ for all $\mathbf{w} \in \mathcal{E}$, and $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}} = 1$. For each $\mathbf{w} \in \mathcal{E}$ and $i \in E$ consider $\mathbf{z}_i^{\mathbf{w}} \in \mathbb{B}^{|E_i^K|}$, defined as follows. If $w_i = 1$, let $\mathbf{z}_i^{\mathbf{w}} = \overline{\mathbf{s}}_i/\overline{x}_i$, otherwise, let $\mathbf{z}_i^{\mathbf{w}} = 0$. Then, $\sum_{H\in E_i^K} z_{iH}^{\mathbf{w}} = \sum_{H\in E_i^K} \overline{s}_{iH}/\overline{x}_i = 1 = w_i$. Also, let $\mathbf{r}_i^{\mathbf{w}} \in \mathbb{B}^{|E_i^K|m}$ be defined as follows. If $w_i = 1$, $\mathbf{r}_i^{\mathbf{w}} = \overline{\mathbf{t}}_i/\overline{x}_i$, otherwise, $\mathbf{r}_i^{\mathbf{w}} = 0$. Then, $\mathbf{r}_{iH}^{\mathbf{w}} \in (X_H \times (\overline{s}_{iH}/\overline{x}_i))$, or, $\mathbf{r}_{iH}^{\mathbf{w}} \in (X_H \times z_{iH}^{\mathbf{w}})$. Therefore, $\mathbf{x} = \mathbf{w}$, $\mathbf{s}_i = \mathbf{z}_i^{\mathbf{w}}$ and $\mathbf{t}_i = \mathbf{r}_i^{\mathbf{w}}$ for all $i \in E$, and $\mathbf{y}_i = \sum_{H\in E_i^K} \mathbf{r}_{iH}^{\mathbf{w}}$ is a feasible point for P_3''.

Consider some $i \in E$. If $\overline{x}_i = 0$, then $w_i = 0$ for all $\mathbf{w} \in \mathcal{E}$, what implies $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{z}_i^{\mathbf{w}} = \mathbf{0} = \overline{\mathbf{s}}_i$, $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{r}_i^{\mathbf{w}} = \mathbf{0} = \overline{\mathbf{t}}_i$, and $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}} \sum_{H\in E_i^K} \mathbf{r}_{iH}^{\mathbf{w}} = \mathbf{0} = \overline{\mathbf{y}}_i$. On the other hand, if $\overline{x}_i > 0$, $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{z}_i^{\mathbf{w}} = \left(\sum_{\mathbf{w}\in\mathcal{E}:w_i=1} \lambda^{\mathbf{w}}\right) \frac{\overline{\mathbf{s}}_i}{\overline{x}_i} = \overline{\mathbf{s}}_i$, $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}\mathbf{r}_i^{\mathbf{w}} = \left(\sum_{\mathbf{w}\in\mathcal{E}:w_i=1} \lambda^{\mathbf{w}}\right) \frac{\overline{\mathbf{t}}_i}{\overline{x}_i} = \overline{\mathbf{t}}_i$, and $\sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}} \sum_{H\in E_i^K} \mathbf{r}_{iH}^{\mathbf{w}} = \left(\sum_{\mathbf{w}\in\mathcal{E}:w_i=1} \lambda^{\mathbf{w}}\right) \sum_{H\in E_i^K} \frac{\overline{\mathbf{t}}_{iH}}{\overline{x}_i} = \overline{\mathbf{y}}_i$. Thus,

$$\begin{aligned}(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}}) &= (\overline{\mathbf{x}}, \overline{\mathbf{y}}_{j_1}, \ldots, \overline{\mathbf{y}}_{j_m}, \overline{\mathbf{s}}_{j_1}, \ldots, \overline{\mathbf{s}}_{j_m}, \overline{\mathbf{t}}_{j_1}, \ldots, \overline{\mathbf{t}}_{j_m}) \\ &= \sum_{\mathbf{w}\in\mathcal{E}} \lambda^{\mathbf{w}}(\mathbf{w}, \sum_{H\in E_{j_1}^K} \mathbf{r}_{j_1 H}^{\mathbf{w}}, \ldots, \sum_{H\in E_{j_m}^K} \mathbf{r}_{j_m H}^{\mathbf{w}}, \mathbf{z}_{j_1}^{\mathbf{w}}, \ldots, \mathbf{z}_{j_m}^{\mathbf{w}}, \mathbf{r}_{j_1}^{\mathbf{w}}, \ldots, \mathbf{r}_{j_m}^{\mathbf{w}})\end{aligned}$$

and $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ cannot be an extreme point. The proof is complete. □

Appendix C

Considerations about the Work of Öncan and Punnen [35]

In this section, we make some observations about the results in [35]. The Lagrangian relaxation in that reference is based on formulation F_{OP10}. They propose a lower bounding scheme based on Lagrangian relaxation, where constraints

$$\sum_{i \in \delta(v)} y_{ij} \geq x_j, \qquad j \in E, v \in V.$$

are relaxed and dualized in a Lagrangian fashion.

The authors claim that the dualization of those constraints will yield the following Lagrangian modified costs:

$$q'_{ij} = q_{ij} - \alpha_{jv}, \qquad i, j \in E, i \neq j, v \in V, \tag{C.1}$$

$$q'_{ii} = q_{ii} + \sum_{u \in V \setminus \{v,w\}} \alpha_{iu}, \qquad i = \{v, w\} \in E. \tag{C.2}$$

While (C.2) is correct, (C.1) is not. The correct Lagrangian modified costs should be:

$$q'_{ij} = q_{ij} - \alpha_{ju} - \alpha_{jv}, \qquad i = \{u, v\}, j \in E, i \neq j.$$

Then, they claim, in Proposition 2 of that study, that the resulting Lagrangian subproblem can be solved by the Gilmore-Lawler algorithm. However, their Lagrangian subproblem is still the QMSTP, albeit with a modified objective function. Clearly, the QMSTP is not solved by the Gilmore-Lawler algorithm alone. Thus, that is claim is not true. Indeed, constraints (2.12) are not satisfied by the solution given by the

Gilmore-Lawler algorithm. As a matter of fact, these constraints had been previously dualized in the work of Assad and Xu [7].

Nevertheless, the procedure proposed in [35], assuming correct Lagrangian modified costs, could still provide a lower bound for QMSTP, in which constraints (2.12) are in relaxed and dualized with zero valued multipliers (with no multiplier adjustment).

As we pointed out in Section 2.3, computational results reported in that work are not in accordance with Proposition 2.3 of our study. The bounds reported by the Lagrangian relaxation scheme in [35] are stronger than $Z(F_1)$. That could have happened due to the incorrect costs (C.1). To further validate the Lagrangian relaxation bounds we present here, we evaluated $Z(F_1)$ by LP means. That is accomplished by a LP cutting plane algorithm that separates (2.2) and (2.5). To solve the separation problems, we used the algorithms described in Appendix A.

Table C.1 presents average lower bounds, as evaluated by ourselves and as reported in [35]. The first three columns present n, m and the instance type. The next three columns present the lower bounds reported in [35] (Lag_{OP}), the lower bounds computed by our implementation of the procedure described in that work (Lag'_{OP}), the lower bounds computed by Lag_1, and $Z(F_1)$. In each line, we report the average for 10 instances.

Instance						
n	m	type	Lag_{OP}	Lag'_{OP}	Lag_1	$Z(F_1)$
10	45	OP1	547.9	414.8	529.5	529.6
11	55	OP1	613.2	459	584.1	584.3
12	66	OP1	652.1	500.2	648.3	648.9
13	78	OP1	713	558.1	706	707.5

Table C.1. Lower Bounds.

Note that bounds $Z(F_1)$ and those provided by Lag_1 are quite similar, what supports the validity of our Lagrangian lower bounds. The bounds reported by [35], however, are above the theoretical bound $Z(F_1)$. The bounds evaluated by our implementation of their strategy, however, are in accordance with the theoretical results.

Appendix D

Detailed Branch-and-Bound Results

In Tables D.1-D.10, we provide detailed QMSTP BB results for BB_1 and BB_2, and also results for BB_{OP} [13]. The first three columns of each table present information concerning the instances: the number of nodes (n), the number of edges (m), the type of the instance (*type*), and the best known upper bound (*ub*). For BB_{CP}, the number of nodes (n_{nodes}) and the total computational time $t(s)$ are presented. For BB_1 and BB_2, we present: the lower bound at the root node of the BB tree (lb_{root}), the computational time for solving the root node ($t_{root}(s)$), the total number of BB nodes explored (n_{nodes}), and the total time to solve the problem ($t(s)$).

In Tables D.11 and D.12, we presented detailed AQMSTP BB results for BCP, BC, and BB_1. The first two columns of each table present an instance identifier (*id*) and the optimal solution value (*ub*). For each BB algorithm we present the lower bound at the root node (lb_{root}), the computational time for solving the root node ($t_{root}(s)$), the total number of BB nodes explored (n_{nodes}), and the total time to solve the problem ($t(s)$).

We only present results for instances that were solved by at least one of the algorithms. Entries with the symbol "-" indicate that the corresponding algorithm was not able to solve the instance within the specified time limit (100 hours for QMSTP instances and 10 hours for AQMSTP instances).

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
10	33	CP1	350	19	0	350	0	1	1	350	0	1	0
10	33	CP2	3122	17	0	3122	0	1	1	3122	0	1	0
10	33	CP3	646	9	0	646	0	1	1	646	0	1	0
10	33	CP4	3486	19	0	3486	0	1	1	3486	0	1	0
10	67	CP1	255	967	0	202.1	0	21	1	242.7	2	13	3
10	67	CP2	2042	1351	0	1398.6	0	31	1	1884.9	2	15	3
10	67	CP3	488	183	0	482.8	0	1	1	487.9	1	1	1
10	67	CP4	2404	1101	0	1793	0	39	1	2255	2	11	3
10	100	CP1	239	8205	0	159.2	0	139	2	210.4	6	72	15
10	100	CP2	1815	8157	0	933.6	0	193	2	1519.3	6	101	14
10	100	CP3	426	1011	0	386	0	11	1	423.2	6	3	6
10	100	CP4	2197	7931	0	1313.8	0	157	2	1894.7	6	71	14
15	33	CP1	745	22743	0	578	0	275	2	679.4	2	106	6
15	33	CP2	6539	25289	0	4684.2	0	315	2	5789.8	3	111	6
15	33	CP3	1236	2871	0	1180	0	17	1	1232.7	4	7	4
15	33	CP4	7245	25913	0	5355.2	0	305	2	6483.7	3	143	6
15	67	CP1	659	4252005	63	384.3	0	16893	59	529.8	17	3377	215
15	67	CP2	5573	3538983	53	2579.7	0	13785	49	4211	20	2843	175
15	67	CP3	966	55881	1	846.2	0	59	5	941.4	16	13	23
15	67	CP4	6188	3706107	55	3204.9	0	14865	52	4804.5	18	3115	192
15	100	CP1	620	24242281	520	319.2	1	109729	718	475.8	44	12683	2479
15	100	CP2	5184	27631419	613	1822.5	1	118607	773	3580.7	55	15415	2412
15	100	CP3	975	802457	16	778.8	1	871	14	901.2	40	215	155
15	100	CP4	5879	33476125	736	2479.1	1	144309	946	4232.5	54	20207	3169
20	33	CP1	1379	51880837	886	886.9	1	96307	305	1137.4	12	24106	1103
20	33	CP2	12425	49240971	879	7037.5	1	90135	290	9773.4	14	22285	992
20	33	CP3	1972	2230621	38	1672.4	1	897	8	1868.8	15	157	38
20	33	CP4	13288	46873773	823	7914	1	90819	289	10662.5	15	19715	879
20	67	CP1	1252	-	-	598.3	2	24431331	271760	-	-	-	-
20	67	CP2	10893	-	-	3797.3	1	15202397	168843	-	-	-	-
20	67	CP3	1792	-	-	1306.1	2	89595	1527	-	-	-	-
20	67	CP4	11893	-	-	4671.6	1	21244515	238189	-	-	-	-

Table D.1. QMSTP branch-and-bound results. Instances of Cordone and Passeri [13].

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
10	45	OP1	613	4497	0	444,2	0	65	2	546,2	5	55	10
10	45	OP1	596	1115	0	493,5	0	11	1	596	4	1	4
10	45	OP1	757	3807	0	601,2	0	55	2	711,4	5	23	10
10	45	OP1	588	863	0	514,1	0	5	1	588	4	1	4
10	45	OP1	710	2753	0	574,5	0	35	1	681	6	11	9
10	45	OP1	647	2141	0	524,7	0	21	1	621,4	6	13	8
10	45	OP1	599	1295	0	489,7	0	13	1	585,1	6	7	8
10	45	OP1	653	2113	0	518,7	0	35	1	630,4	6	11	8
10	45	OP1	753	2577	0	627,2	0	19	1	718,1	6	15	9
10	45	OP1	623	2187	0	506,8	0	25	1	606	6	9	9
11	55	OP1	755	9799	0	541	0	103	3	693,4	9	43	16
11	55	OP1	750	3203	0	627	0	13	2	736,8	9	11	11
11	55	OP1	807	10327	0	583,3	0	131	2	732,3	8	61	16
11	55	OP1	816	8833	0	598,9	0	87	2	747,1	9	42	18
11	55	OP1	759	12111	0	539,8	0	119	2	703,6	8	39	14
11	55	OP1	877	12933	0	649,2	0	159	2	799,8	8	87	20
11	55	OP1	734	8063	0	529,7	0	73	2	685,3	7	35	13
11	55	OP1	843	13063	0	631,3	0	117	2	773,7	9	39	19
11	55	OP1	797	14593	0	575,9	0	153	2	723,5	8	49	18
11	55	OP1	721	3849	0	564,9	0	37	2	689,8	9	11	12
12	66	OP1	975	90541	1	659,2	0	701	4	833,2	15	327	46
12	66	OP1	903	28745	0	661,7	0	243	4	824	16	115	40
12	66	OP1	977	78679	1	655,4	0	591	4	848,5	16	259	43
12	66	OP1	936	11781	0	702,6	0	99	3	879,6	16	21	26
12	66	OP1	863	25075	0	601,7	0	169	3	782,4	16	69	34
12	66	OP1	991	49673	0	674,5	0	425	4	862,8	16	191	42
12	66	OP1	848	13657	0	645,7	0	69	4	786,9	16	25	27
12	66	OP1	842	52331	0	558,1	0	331	3	734,1	16	137	38
12	66	OP1	965	31605	0	699,1	0	191	3	877,6	15	83	33
12	66	OP1	885	26523	0	625	0	153	4	803,2	16	45	36

Table D.2. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 1.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
13	78	OP1	990	174049	2	606,3	0	1005	8	842,6	21	223	54
13	78	OP1	1022	83093	1	702	0	351	5	900,1	22	115	52
13	78	OP1	1089	199691	3	697	0	909	9	945,2	22	213	63
13	78	OP1	1163	183977	2	803,1	0	879	7	1033,3	21	175	53
13	78	OP1	1129	193607	3	748,8	0	869	7	1001,4	21	190	55
13	78	OP1	1023	80917	1	715,4	0	269	5	933	21	47	42
13	78	OP1	982	174309	2	613,3	0	693	7	838,3	22	249	64
13	78	OP1	1048	100115	1	712,8	0	429	5	929,4	22	123	54
13	78	OP1	1065	56937	0	741,2	0	291	6	980,3	22	49	56
13	78	OP1	1160	480411	7	720,3	0	4743	22	978,6	21	1445	131
14	91	OP1	1246	982445	18	762,3	0	2837	21	1029,1	33	799	157
14	91	OP1	1197	338485	6	783,9	0	933	12	1045,2	32	181	96
14	91	OP1	1398	2025109	37	868,4	0	12245	71	1148,9	33	2821	396
14	91	OP1	1276	1021699	19	791,2	0	3243	24	1043,9	34	1239	216
14	91	OP1	1266	603819	11	809,6	0	1879	16	1074,1	35	469	120
14	91	OP1	1188	517429	10	736,8	0	1823	18	1012	36	435	129
14	91	OP1	1312	1127341	21	806,3	0	4155	29	1076,8	35	1535	263
14	91	OP1	1171	170639	3	807,9	0	389	7	1071,8	33	53	86
14	91	OP1	1301	816627	16	830,8	0	2437	20	1097,3	32	785	176
14	91	OP1	1143	223427	4	766,5	0	679	9	1014,7	34	125	90
15	105	OP1	1401	3735849	87	804,2	1	9419	84	1155,5	45	1499	343
15	105	OP1	1404	2141949	49	841,3	1	7063	64	1178,1	43	858	216
15	105	OP1	1384	3593455	84	812,6	1	8835	79	1135,8	46	1465	333
15	105	OP1	1376	7922195	184	741,2	1	19239	158	1083,6	44	4057	775
15	105	OP1	1295	1810267	44	766,5	1	3143	33	1090,1	47	523	181
15	105	OP1	1473	2807121	64	899,2	1	6857	65	1227,4	44	1441	347
15	105	OP1	1389	4091815	95	782	1	10865	92	1128,7	44	1645	368
15	105	OP1	1407	2879425	66	847,4	1	6189	173	1176,2	44	937	246
15	105	OP1	1333	2191531	51	774,2	1	6041	55	1112,6	43	741	201
15	105	OP1	1440	3039535	71	871,7	1	7755	66	1198,9	44	1137	278

Table D.3. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 1.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
16	120	OP1	1624	-	-	863,2	1	66679	733	1241,7	73	14389	4766
16	120	OP1	1667	-	-	913	1	87299	923	1297,6	74	18641	5928
16	120	OP1	1629	-	-	900,2	1	44877	473	1265,1	84	8993	2891
16	120	OP1	1659	-	-	903,3	1	72119	783	1271,4	78	20374	6708
16	120	OP1	1695	-	-	930,7	1	67577	733	1311,9	69	15880	5170
16	120	OP1	1518	-	-	833,4	1	27547	289	1225,2	72	2449	948
16	120	OP1	1652	-	-	864,8	1	128423	1365	1256,1	74	21287	7029
16	120	OP1	1687	-	-	959,2	1	43201	457	1347,2	73	7975	2626
16	120	OP1	1543	-	-	785	1	72719	777	1162,2	71	19273	6233
16	120	OP1	1619	-	-	887,3	1	42193	454	1266,4	76	7445	2530
17	136	OP1	1843	-	-	973,2	1	191595	2712	1415,6	104	24776	10771
17	136	OP1	1828	-	-	984,8	1	101253	1546	1441,4	95	11588	5171
17	136	OP1	1859	-	-	1021,3	1	127597	1848	1459,8	98	15239	6611
17	136	OP1	1839	-	-	941,4	1	287547	4263	1398,2	97	38661	16345
17	136	OP1	1795	-	-	904,8	1	449565	6385	1359,8	116	46463	18853
17	136	OP1	1817	-	-	946,1	1	215273	3168	1388	98	29827	13176
17	136	OP1	1893	-	-	980,4	1	382019	5370	1438,8	89	48463	20079
17	136	OP1	1818	-	-	987,4	1	109887	1602	1436,4	104	12866	5936
17	136	OP1	1734	-	-	932,4	1	67545	1098	1369,1	97	7507	3392
17	136	OP1	1812	-	-	922,7	1	242045	3582	1365,3	103	35631	15514
18	153	OP1	2153	-	-	1075,6	2	1516447	26497	-	-	-	-
18	153	OP1	2125	-	-	1037,5	2	1890921	32555	-	-	-	-
18	153	OP1	2108	-	-	1094,6	2	568609	9659	-	-	-	-
18	153	OP1	2026	-	-	1029,8	2	857377	14439	-	-	-	-
18	153	OP1	2028	-	-	941,8	2	1238159	23234	-	-	-	-
18	153	OP1	2023	-	-	976,4	2	1121977	20526	-	-	-	-
18	153	OP1	1951	-	-	961,2	2	552413	9755	-	-	-	-
18	153	OP1	2089	-	-	957,3	2	3191969	55061	-	-	-	-
18	153	OP1	2138	-	-	995,5	2	5351735	93178	-	-	-	-
18	153	OP1	2169	-	-	1141	2	1010763	18211	-	-	-	-

Table D.4. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 1.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
10	45	OP2	32105	59	0	32105	0	1	1	32105	3	1	3
10	45	OP2	23863	65	0	23863	0	1	1	23863	3	1	4
10	45	OP2	18338	55	0	18338	0	1	1	18338	3	1	3
10	45	OP2	24613	105	0	24613	0	1	1	24613	3	1	3
10	45	OP2	35699	173	0	35699	0	1	1	35699	3	1	3
10	45	OP2	21401	99	0	21401	0	1	1	21401	3	1	3
10	45	OP2	26997	95	0	26997	0	1	1	26997	3	1	3
10	45	OP2	22992	123	0	22992	0	1	1	22992	3	1	3
10	45	OP2	28833	109	0	28833	0	1	1	28833	3	1	3
10	45	OP2	22597	63	0	22597	0	1	1	22597	3	1	3
11	55	OP2	40359	893	0	40359	0	1	1	40359	5	1	5
11	55	OP2	22735	175	0	22735	0	1	1	22735	5	1	5
11	55	OP2	27723	333	0	27723	0	1	1	27723	6	1	6
11	55	OP2	35474	105	0	35474	0	1	1	35474	4	1	5
11	55	OP2	29778	281	0	29778	0	1	1	29778	5	1	5
11	55	OP2	23877	219	0	23877	0	1	1	23877	5	1	5
11	55	OP2	37495	339	0	37495	0	1	1	37495	4	1	4
11	55	OP2	22705	49	0	22705	0	1	1	22705	4	1	5
11	55	OP2	19020	111	0	19020	0	1	1	19020	4	1	5
11	55	OP2	33910	311	0	33910	0	1	1	33910	4	1	5
12	66	OP2	35542	89	0	35542	0	1	1	35542	8	1	9
12	66	OP2	20585	129	0	20585	0	1	2	20585	7	1	8
12	66	OP2	38153	1103	0	38153	0	1	1	38153	8	1	8
12	66	OP2	32015	361	0	32015	0	1	1	32015	7	1	8
12	66	OP2	34136	227	0	34136	0	1	1	34136	9	1	9
12	66	OP2	42814	83	0	42814	0	1	1	42814	9	1	10
12	66	OP2	30153	353	0	30153	0	1	1	30153	8	1	8
12	66	OP2	25646	227	0	25646	0	1	1	25646	9	1	9
12	66	OP2	34183	145	0	34183	0	1	1	34183	8	1	8
12	66	OP2	32551	155	0	32551	0	1	1	32551	8	1	9
13	78	OP2	45586	347	0	45586	0	1	2	45586	12	1	13
13	78	OP2	49313	2185	0	49313	0	1	2	49313	11	1	12
13	78	OP2	44513	509	0	44513	0	1	2	44513	11	1	12
13	78	OP2	37250	91	0	37250	0	1	2	37250	11	1	12
13	78	OP2	50990	601	0	50990	0	1	2	50990	11	1	11
13	78	OP2	43261	481	0	43261	0	1	2	43261	12	1	12
13	78	OP2	36085	281	0	36085	0	1	2	36085	11	1	12
13	78	OP2	34474	63	0	34474	0	1	2	34474	10	1	11
13	78	OP2	28566	235	0	28566	0	1	2	28566	10	1	11
13	78	OP2	34847	357	0	34847	0	1	2	34847	13	1	14

Table D.5. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 2.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
14	91	OP2	41065	179	0	41065	0	1	3	41065	16	1	18
14	91	OP2	41656	747	0	41656	0	1	2	41656	18	1	19
14	91	OP2	48541	273	0	48541	0	1	3	48541	16	1	17
14	91	OP2	40335	269	0	40335	0	1	2	40335	19	1	20
14	91	OP2	39599	297	0	39599	0	1	3	39599	17	1	18
14	91	OP2	45279	813	0	45279	0	1	3	45279	16	1	17
14	91	OP2	55026	613	0	55026	0	1	2	55026	18	1	19
14	91	OP2	47163	981	0	47163	0	1	2	47163	16	1	17
14	91	OP2	43475	183	0	43475	0	1	2	43475	15	1	16
14	91	OP2	40265	237	0	40265	0	1	2	40265	17	1	18
15	105	OP2	62365	1101	0	62365	0	1	3	62365	20	1	23
15	105	OP2	50019	1477	0	50019	0	1	3	50019	23	1	25
15	105	OP2	56720	1007	0	56720	0	1	3	56720	24	1	26
15	105	OP2	58992	625	0	58992	0	1	3	58992	22	1	24
15	105	OP2	31530	421	0	31530	0	1	3	31530	21	1	22
15	105	OP2	54726	1723	0	54726	0	1	3	54726	28	1	29
15	105	OP2	49804	1435	0	49804	0	1	3	49804	26	1	28
15	105	OP2	49286	1223	0	49286	0	1	3	49286	29	1	30
15	105	OP2	41852	829	0	41852	0	1	3	41852	29	1	30
15	105	OP2	52922	1857	0	52922	0	1	3	52922	25	1	27
16	120	OP2	42342	699	0	42342	1	1	4	42342	39	1	42
16	120	OP2	45092	1819	0	45092	1	1	4	45092	33	1	35
16	120	OP2	41701	2053	0	41701	1	1	5	41701	39	1	42
16	120	OP2	46319	4351	0	46319	1	1	5	46319	32	1	36
16	120	OP2	41371	645	0	41371	1	1	4	41371	32	1	35
16	120	OP2	45108	1707	0	45108	1	1	5	45108	34	1	37
16	120	OP2	42021	1471	0	42021	1	1	4	42021	37	1	40
16	120	OP2	30880	445	0	30880	1	1	4	30880	32	1	35
16	120	OP2	37409	459	0	37409	1	1	4	37409	35	1	37
16	120	OP2	47159	16111	0	46906.4	1	7	4	46946.2	71	3	80
17	136	OP2	36696	797	0	36696	1	1	6	36696	46	1	49
17	136	OP2	48774	7163	0	48774	1	1	9	48774	46	1	52
17	136	OP2	45025	22963	0	45025	1	1	5	45025	45	1	48
17	136	OP2	33751	6503	0	33751	1	1	5	33751	61	1	63
17	136	OP2	44692	3463	0	44692	1	1	6	44692	53	1	57
17	136	OP2	45465	11419	0	45465	1	1	7	45465	49	1	54
17	136	OP2	38428	4139	0	38428	1	1	7	38428	63	1	67
17	136	OP2	40172	4933	0	40172	1	1	6	40172	50	1	54
17	136	OP2	44717	10113	0	44717	1	1	5	44717	51	1	54
17	136	OP2	40470	1987	0	40470	1	1	7	40470	47	1	52

Table D.6. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 2.

Instance				BB_{CP}		BB_1				BB_2			
n	m	$type$	ub	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
18	153	OP2	36845	6101	0	36845	1	1	8	36845	73	1	79
18	153	OP2	51694	49937	1	51694	1	1	9	51694	63	1	70
18	153	OP2	42143	14247	0	42143	1	1	8	42143	64	1	70
18	153	OP2	40601	5411	0	40601	1	1	6	40601	71	1	75
18	153	OP2	41237	1653	0	41237	1	1	8	41237	71	1	76
18	153	OP2	50000	21035	0	50000	1	1	9	50000	71	1	78
18	153	OP2	52766	29727	1	52766	1	1	9	52766	73	1	80
18	153	OP2	54200	29233	1	54200	1	1	6	54200	60	1	63
18	153	OP2	46867	3177	0	46867	1	1	7	46867	74	1	79
18	153	OP2	44949	10359	0	44949	1	1	7	44949	63	1	68
20	190	OP2	50610	29637	1	50610	2	1	14	50610	109	1	120
20	190	OP2	53427	72855	3	53427	2	1	13	53427	108	1	118
20	190	OP2	51497	87843	3	51497	2	1	12	51497	117	1	126
20	190	OP2	57638	27955	1	57638	2	1	10	57638	120	1	127
20	190	OP2	56344	330715	15	56344	2	1	14	56344	114	1	125
20	190	OP2	54615	93887	4	54615	2	1	12	54615	117	1	126
20	190	OP2	61214	85795	4	61214	2	1	15	61214	121	1	132
20	190	OP2	52650	63967	3	52650	2	1	14	52650	112	1	123
20	190	OP2	64980	583379	29	64980	2	1	11	64980	105	1	113
20	190	OP2	50287	461769	21	50287	2	1	13	50287	108	1	117
30	435	OP2	82953	-	-	82953	12	1	85	82953	1383	1	1454
30	435	OP2	76977	-	-	76977	12	1	115	76977	1186	1	1287
30	435	OP2	88098	-	-	88096	13	1	86	88096	1826	1	1899
30	435	OP2	90361	-	-	90361	12	1	99	90361	1319	1	1410
30	435	OP2	69976	-	-	69976	12	1	117	69976	1327	1	1429
30	435	OP2	78864	-	-	78864	12	1	150	78864	1245	1	1383
30	435	OP2	73015	-	-	73015	12	1	111	73015	1112	1	1209
30	435	OP2	73619	-	-	73619	12	1	133	73619	1170	1	1287
30	435	OP2	81534	-	-	81534	12	1	121	81534	1393	1	1501
30	435	OP2	74602	-	-	74602	12	1	86	74602	1362	1	1437
50	1225	OP2	172157	-	-	172157	177	1	1338	-	-	-	-
50	1225	OP2	170915	-	-	170915	175	1	1317	-	-	-	-
50	1225	OP2	160256	-	-	160256	176	1	1503	-	-	-	-
50	1225	OP2	152830	-	-	152830	174	1	1389	-	-	-	-
50	1225	OP2	174926	-	-	174926	176	1	1133	-	-	-	-
50	1225	OP2	154341	-	-	154341	173	1	1731	-	-	-	-
50	1225	OP2	180023	-	-	180023	176	1	1214	-	-	-	-
50	1225	OP2	153578	-	-	153578	175	1	1295	-	-	-	-
50	1225	OP2	179932	-	-	179932	174	1	1399	-	-	-	-
50	1225	OP2	155241	-	-	155241	175	1	1456	-	-	-	-

Table D.7. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 2.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
10	45	OP3	1047	801	0	1047	0	1	1	1047	3	1	3
10	45	OP3	1454	3383	0	1454	0	1	1	1454	3	1	3
10	45	OP3	1508	269	0	1508	0	1	1	1508	3	1	3
10	45	OP3	1490	1351	0	1470.5	0	5	1	1478	4	3	5
10	45	OP3	1880	549	0	1878.7	0	1	1	1880	3	1	3
10	45	OP3	1430	1273	0	1411.6	0	3	1	1430	3	1	3
10	45	OP3	1275	1257	0	1274	0	1	1	1275	3	1	3
10	45	OP3	1412	239	0	1393.1	0	3	1	1412	3	1	3
10	45	OP3	1522	649	0	1522	0	1	1	1522	3	1	3
10	45	OP3	1253	219	0	1248.3	0	1	1	1253	3	1	3
11	55	OP3	1175	1037	0	1170.9	0	1	1	1175	5	1	5
11	55	OP3	1614	207	0	1614	0	1	1	1614	5	1	5
11	55	OP3	2065	519	0	2045.3	0	1	1	2065	5	1	5
11	55	OP3	1854	1841	0	1830.7	0	3	1	1854	4	1	5
11	55	OP3	1424	159	0	1424	0	1	1	1424	4	1	4
11	55	OP3	1442	1123	0	1442	0	1	1	1442	4	1	4
11	55	OP3	1208	17495	0	1208	0	1	1	1208	4	1	4
11	55	OP3	1444	18889	0	1432.6	0	5	1	1437.4	9	9	12
11	55	OP3	1716	841	0	1716	0	1	1	1716	4	1	4
11	55	OP3	1509	755	0	1509	0	1	1	1509	4	1	4
12	66	OP3	1586	5449	0	1582.4	0	1	1	1586	8	1	8
12	66	OP3	2132	5079	0	2086.2	0	7	1	2121.9	16	5	19
12	66	OP3	1798	997	0	1798	0	1	1	1798	7	1	8
12	66	OP3	2227	2039	0	2226.8	0	1	1	2227	7	1	8
12	66	OP3	1768	1795	0	1768	0	1	1	1768	7	1	7
12	66	OP3	1488	40719	0	1485	0	3	1	1488	8	1	8
12	66	OP3	1813	319	0	1813	0	1	1	1813	7	1	8
12	66	OP3	2057	13491	0	2023.9	0	3	1	2042.2	17	7	19
12	66	OP3	2071	787	0	2071	0	1	1	2071	8	1	8
12	66	OP3	2076	1035	0	2076	0	1	1	2076	8	1	8
13	78	OP3	1731	15789	0	1731	0	1	2	1731	9	1	10
13	78	OP3	2484	917	0	2484	0	1	2	2484	10	1	11
13	78	OP3	2440	5533	0	2436.6	0	1	2	2440	12	1	12
13	78	OP3	2489	1881	0	2453.2	0	9	2	2483	23	3	25
13	78	OP3	2044	2549	0	2044	0	1	2	2044	11	1	12
13	78	OP3	1806	6509	0	1805	0	1	2	1806	11	1	11
13	78	OP3	2185	2363	0	2185	0	1	2	2185	10	1	11
13	78	OP3	2275	21009	0	2272.8	0	1	2	2275	11	1	11
13	78	OP3	1968	125889	2	1943.1	0	3	2	1957.7	21	3	24
13	78	OP3	2331	937	0	2331	0	1	2	2331	10	1	11

Table D.8. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 3.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
14	91	OP3	1955	35983	0	1955	0	1	2	1955	16	1	17
14	91	OP3	2555	1297	0	2555	0	1	2	2555	15	1	16
14	91	OP3	3182	10799	0	3169.8	0	3	2	3182	16	1	17
14	91	OP3	2516	72339	1	2516	0	1	2	2516	17	1	18
14	91	OP3	2551	9005	0	2551	0	1	2	2551	16	1	17
14	91	OP3	2818	119579	2	2797	0	5	4	2816.2	38	7	46
14	91	OP3	2457	23607	0	2457	0	1	2	2457	16	1	17
14	91	OP3	2491	14763	0	2482.9	0	1	3	2491	17	1	19
14	91	OP3	2293	237	0	2293	0	1	3	2293	15	1	17
14	91	OP3	2461	5509	0	2454.1	0	1	3	2460.9	19	1	20
15	105	OP3	2181	97099	2	2177.7	0	1	3	2181	22	1	23
15	105	OP3	2840	3911	0	2840	0	1	3	2840	23	1	25
15	105	OP3	2921	4111	0	2921	0	1	4	2921	22	1	24
15	105	OP3	2780	3641	0	2780	0	1	3	2780	23	1	24
15	105	OP3	2340	84369	1	2305.5	1	15	5	2340	29	1	31
15	105	OP3	2917	11983	0	2904.9	1	3	3	2917	25	1	26
15	105	OP3	2291	27405	0	2265	1	5	4	2291	24	1	25
15	105	OP3	2537	781	0	2537	0	1	4	2537	23	1	25
15	105	OP3	2504	6853	0	2495	1	3	2	2504	24	1	24
15	105	OP3	2577	4879	0	2575.9	0	1	4	2577	21	1	24
16	120	OP3	2418	211997	5	2408.5	1	1	5	2418	36	1	39
16	120	OP3	2507	26651	0	2507	1	1	4	2507	36	1	38
16	120	OP3	3241	33041	0	3214.6	1	1	7	3241	33	1	37
16	120	OP3	3600	662209	16	3585.5	1	3	5	3593	76	12	106
16	120	OP3	3097	24013	0	3022.5	1	5	6	3096.8	73	1	76
16	120	OP3	2741	8043	0	2741	1	1	5	2741	32	1	35
16	120	OP3	3264	60473	1	3216.6	1	15	7	3248.7	70	13	94
16	120	OP3	2958	88063	2	2943.7	1	1	4	2958	41	1	42
16	120	OP3	3079	38665	0	3079	1	1	4	3079	36	1	38
16	120	OP3	2896	605	0	2896	1	1	6	2896	34	1	38
17	136	OP3	2842	987395	31	2835.7	1	1	6	2842	51	1	54
17	136	OP3	3734	1000000	36	3724.9	1	3	5	3734	64	1	67
17	136	OP3	3543	687011	21	3455.7	2	9	7	3513.9	85	9	115
17	136	OP3	3165	382535	12	3150.1	1	5	7	3161.2	92	5	104
17	136	OP3	2920	4437	0	2920	1	1	7	2920	42	1	46
17	136	OP3	3445	17131	0	3442.4	1	1	7	3445	42	1	47
17	136	OP3	3483	380767	12	3419.1	2	7	9	3482.9	104	1	108
17	136	OP3	3321	352711	10	3252.6	1	21	7	3316.9	86	11	117
17	136	OP3	3460	52593	1	3460	1	1	6	3460	45	1	48
17	136	OP3	3809	71073	2	3757.4	1	3	8	3805.1	115	11	150

Table D.9. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 3.

Instance				BB_{CP}		BB_1				BB_2			
n	m	*type*	*ub*	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
18	153	OP3	2970	979125	35	2968.1	2	1	8	2970	64	1	69
18	153	OP3	3733	154609	5	3706.6	2	5	8	3730.2	156	11	217
18	153	OP3	3563	327861	10	3543.6	2	1	9	3563	68	1	75
18	153	OP3	4060	1000000	33	4044.7	2	1	10	4058.5	148	1	167
18	153	OP3	3300	127447	4	3300	1	1	8	3300	63	1	68
18	153	OP3	3191	61887	2	3191	1	1	7	3191	62	1	66
18	153	OP3	3700	376161	16	3692	2	1	7	3700	68	1	73
18	153	OP3	3560	4955	0	3546.5	2	1	11	3560	60	1	68
18	153	OP3	3990	73855	2	3964.2	2	3	10	3990	72	1	79
18	153	OP3	3623	778303	29	3561.7	2	9	11	3612.8	134	3	155
20	190	OP2	22428	17	0	22428	0	1	1	22428	0	1	0
20	190	OP2	15253	21	0	15253	0	1	1	15253	0	1	0
20	190	OP2	21570	23	0	21570	0	1	1	21570	0	1	0
20	190	OP2	19535	9	0	19535	0	1	1	19535	0	1	0
20	190	OP2	24964	15	0	24964	0	1	1	24964	0	1	0
20	190	OP2	9821	21	0	9821	0	1	1	9821	0	1	0
20	190	OP2	11810	11	0	11810	0	1	1	11810	0	1	0
20	190	OP2	13869	11	0	13869	0	1	1	13869	0	1	0
20	190	OP2	7742	13	0	7742	0	1	1	7742	0	1	0
20	190	OP2	15747	7	0	15747	0	1	1	15747	0	1	0
30	435	OP2	6992	-	-	6865.3	15	35	214	6964.2	2437	15	5157
30	435	OP2	9057	-	-	8700.3	16	129	491	8859.5	2069	81	17856
30	435	OP2	7823	-	-	7823	13	1	179	7823	1233	1	1395
30	435	OP2	7936	-	-	7886.1	14	1	200	7936	1214	1	1392
30	435	OP2	8092	-	-	8042.1	18	9	272	8092	1380	1	1597
30	435	OP2	8566	-	-	8438.3	15	15	223	8533.5	2928	9	5691
30	435	OP2	7525	-	-	7364.4	14	41	342	7472.3	2079	31	4958
30	435	OP2	8645	-	-	8409.5	15	49	318	8545.9	2211	29	8001
30	435	OP2	8692	-	-	8526.1	15	31	296	8653.4	2348	28	5015
30	435	OP2	7239	-	-	7209.2	14	1	197	7239	1422	1	1601
50	1225	OP2	17524	-	-	16933.3	195	735	19045	-	-	-	-
50	1225	OP2	16780	-	-	16558.9	192	13	6061	-	-	-	-
50	1225	OP2	13198	-	-	12940.4	189	71	8244	-	-	-	-
50	1225	OP2	15137	-	-	14708.5	193	123	11761	-	-	-	-
50	1225	OP2	16358	-	-	15677.9	195	405	14950	-	-	-	-
50	1225	OP2	14996	-	-	14781.4	190	21	8791	-	-	-	-
50	1225	OP2	17282	-	-	17222.5	188	1	5219	-	-	-	-
50	1225	OP2	14975	-	-	14959.9	185	1	4856	-	-	-	-
50	1225	OP2	13594	-	-	13591.8	185	1	3920	-	-	-	-
50	1225	OP2	18062	-	-	17736	193	111	8141	-	-	-	-

Table D.10. QMSTP branch-and-bound results. Instances of Öncan and Punnen [35], type 3.

Instance		*BCP*				*BC*				BB_1			
id	ub^*	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
$n = 15$													
1	310	297.1	2	5	2	297.1	2	9	3	221.1	1	667	11
2	267	260	2	5	3	260	3	3	3	155.6	1	341	12
3	252	252	2	1	2	249	2	0	2	168.2	1	231	10
4	271	271	2	1	3	271	3	0	3	162.6	1	385	12
5	206	196	2	5	2	196	2	11	2	105.3	1	453	13
6	273	269.5	2	3	3	265	3	3	3	141	1	559	11
7	304	298	3	5	3	218.4	3	3	3	183.3	1	649	10
8	226	226	2	1	2	226	2	0	2	160.8	1	95	12
9	333	333	3	1	4	331.3	4	0	4	218.6	1	311	12
10	315	311	4	3	5	160	4	3	5	200.7	1	771	12
$n = 20$													
1	356	356	14	1	14	354	15	0	15	239.2	3	2337	165
2	349	342.9	13	7	17	342.8	14	14	15	207.6	3	6463	382
3	386	379	12	5	15	378.8	13	9	14	237.4	3	11003	603
4	339	323	16	5	19	322.8	17	14	18	216.7	3	10908	602
5	332	332	13	1	13	332	13	0	13	170.7	3	19583	1213
6	361	357.7	14	3	15	357.7	16	0	16	211	3	22201	1245
7	391	391	13	1	14	384.1	13	7	14	258.8	3	5925	342
8	323	315.6	13	9	16	315.6	13	11	14	193.5	3	7385	436
9	322	304	13	7	18	304	14	40	16	165.4	3	19963	1215
10	411	390.2	16	17	28	390.2	19	52	21	237.5	3	21545	1315
$n = 30$													
1	421	406.5	149	11	357	406.5	166	48	191	-	-	-	-
2	450	439.5	130	7	243	439.5	165	14	174	-	-	-	-
3	464	441.9	151	29	778	441.9	162	112	194	-	-	-	-
4	410	404	101	5	134	404	109	6	113	-	-	-	-
5	460	454	137	7	192	453.9	174	9	180	-	-	-	-
6	423	407	148	13	334	407	177	36	193	-	-	-	-
7	441	421	130	21	444	420.9	150	173	188	-	-	-	-
8	400	391.6	116	13	211	391.6	140	28	153	-	-	-	-
9	462	453	143	25	476	453	170	11	182	-	-	-	-
10	418	415.5	110	3	149	414.9	122	0	126	-	-	-	-

Table D.11. AQMSTP branch-and-bound results ($n \in \{15, 20, 30\}$).

Instance			*BCP*				*BC*				BB_1			
id	ub^*	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
$n = 40$													
1	543	524.1	685	13	3632	524	926	88	1123	-	-	-	-
2	502	488.7	618	21	3913	488.6	857	46	991	-	-	-	-
3	492	475.3	731	31	5802	475.2	876	142	1339	-	-	-	-
4	505	472.7	667	75	17253	472.7	868	591	1577	-	-	-	-
5	464	441.7	524	13	3088	441.7	750	93	934	-	-	-	-
6	466	454.9	554	9	1784	454.8	729	31	817	-	-	-	-
7	471	451.5	685	21	4398	451.4	877	81	1061	-	-	-	-
8	435	423	688	9	1681	423	795	20	882	-	-	-	-
9	508	485.4	711	45	8988	485.4	865	150	1146	-	-	-	-
10	481	467.8	639	21	3779	467.7	860	27	940	-	-	-	-
$n = 50$													
1	587	-	-	-	-	513.3	4290	291	8123	-	-	-	-
2	532	-	-	-	-	516.8	4540	383	7949	-	-	-	-
3	589	-	-	-	-	550	4504	1042	9974	-	-	-	-
4	598	-	-	-	-	544.3	3812	2746	16616	-	-	-	-
5	556	-	-	-	-	525.8	3412	244	5033	-	-	-	-
6	638	-	-	-	-	580.9	4788	483	9085	-	-	-	-
7	571	-	-	-	-	533.7	3869	119	5132	-	-	-	-
8	522	-	-	-	-	489.6	3341	433	6697	-	-	-	-
9	498	485.7	2503	9	12054	485.7	3668	34	4239	-	-	-	-
10	506	-	-	-	-	473.3	3382	612	6364	-	-	-	-

Table D.12. AQMSTP branch-and-bound results ($n \in \{40, 50\}$).

Bibliography

[1] Achterberg, T., Koch, T., and Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, 33(1):42--54.

[2] Adams, W. P., Guignard, M., Hahn, P. M., and Hightower, W. (2007). A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *European Journal of Operational Research*, 180(3):1274 -- 1290.

[3] Adams, W. P. and Johnson, T. A. (1994). Improved linear programming-based lower bounds for the quadratic assignment problem. In Pardalos, P. M. and Wolkowicz, H., editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 16, pages 43--75. AMS.

[4] Adams, W. P. and Sherali, H. D. (1986). A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274 -- 1290.

[5] Andersen, L. D. and Fleischner, H. (1995). The np-completeness of finding a-trails in eulerian graphs and of finding spanning trees in hypergraphs. *Discrete Applied Mathematics*, 59(3):203--214.

[6] Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study*. Princenton University Press.

[7] Assad, A. and Xu, W. (1992). The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399--417.

[8] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238--252.

[9] Birkhoff, G. (1946). Tres observaciones sobre el algebra lineal. *Universidad Nacional de Tucumán, Revista A*, 5:147--151.

[10] Burkard, R. E., Çela, E., Pardalos, P. M., and Pitsoulis, L. S. (1998). The quadratic assignment problem. In Du, D.-Z. and Pardalos, P. M., editors, *Handbook of Combinatorial Optimization*, volume 3, pages 241 -- 337. Kluwer Academic Publishers.

[11] Caprara, A. (2008). Constrained 0–1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research*, 187(3):1494--1503.

[12] Cordone, R. (2012). The quadratic minimum spanning tree problem (qmstp).

[13] Cordone, R. and Passeri, G. (2012). Solving the quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, 218(23):11597--11612.

[14] Dantzig, G., Fulkerson, D., and Johnson, S. (1954). Solution of a large scale traveling salesman problem. *Operations Research*, 2:393--410.

[15] Edmonds, J. (1965). Maximum matching and a polyhedron with $0, 1$ vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125--130.

[16] Edmonds, J. (1971). Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127--136.

[17] Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248--264.

[18] Fagin, R. (1983). Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30(3):514--550.

[19] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co.

[20] Geoffrion, A. M. (1974). Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82--114.

[21] Gerards, A. M. H. (1995). Matching. In Ball, M., Magnanti, T., Monma, C., and Nemhauser, G., editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 3, pages 135--224. Elsevier.

[22] Gilmore, P. C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):305--313.

[23] Goncalves, A. D., Drummond, L. M., Pessoa, A. A., and Hahn, P. (2013). Improving lower bounds for the quadratic assignment problem by applying a distributed dual ascent algorithm. *ArXiv e-prints*.

[24] Hahn, P. and Grant, T. (98). Lower bounds for the quadratic assignment problem based upon a dual formulation. *Operations Research*, 46(6):912 -- 922.

[25] Held, M., Wolfe, P., and Crowder, H. (1974). Validation of subgradient optimization. *Mathematical Programming*, 6:62--88.

[26] Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica*, 25(1):53--76.

[27] Kruskal, J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48--50.

[28] Lawler, E. L. (1963). The quadratic assignment problem. *Management Science*, 9(4):586--599.

[29] Lee, J. and Leung, J. (2004). On the boolean quadric forest polytope. *INFOR*, 42(2):125--141.

[30] Lucena, A. (2005). Non delayed relax-and-cut algorithms. *Annals of Operations Research*, 140(1):375--410.

[31] Maia, S. M. D. M., Goldbarg, E. F. G., and Goldbarg, M. C. (2013). On the biobjective adjacent only quadratic spanning tree problem. *Electronic Notes in Discrete Mathematics*, 41:535 -- 542. Presented at the 2013 International Network Optimization Conference (INOC).

[32] Mans, B., Mautor, T., and Roucairol, C. (1995). A parallel depth first search branch and bound algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 81(3):617 -- 628.

[33] Martin, R. K. (1998). *Large Scale Linear and Integer Optimization: A Unified Approach*. Springer.

[34] Myung, Y.-S., Lee, C.-H., and Tcha, D.-W. (1995). On the generalized minimum spanning tree problem. *Networks*, 26(4):231--241.

[35] Öncan, T. and Punnen, A. P. (2010). The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. *Computers and Operations Research*, 37(10):1762--1773.

[36] Padberg, M. (1989). The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming*, 45(1–3):139--172.

[37] Padberg, M. W. and Wolsey, L. A. (1983). Trees and cuts. In *Combinatorial Mathematics Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75, pages 511--517.

[38] Palubeckis, G., Rubliauskas, D., and Targamadzė, A. (2010). Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control*, 39:257--268.

[39] Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems. *Computers and Operations Research*, 33(5):1380 -- 1398.

[40] Pisinger, D. (2007). The quadratic knapsack problem—a survey. *Discrete Applied Mathematics*, 155(5):623 -- 648.

[41] Plesník, J. (1999). Constrained weighted matchings and edge coverings in graphs. *Discrete Applied Mathematics*, 92(2–3):229--241.

[42] Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389--1401.

[43] Prömel, H. and Steger, A. (1997). Rnc-approximation algorithms for the steiner problem. In *STACS 97*, volume 1200 of *Lecture Notes in Computer Science*, pages 559--570. Springer Berlin Heidelberg.

[44] Sherali, H. D. and Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems. *Discrete Applied Mathematics*, 52(1):83 -- 106.

[45] Soak, S.-M., Corne, D., and Ahn, B.-H. (2006). The edge-window-decoder representation for tree-based problems. *Evolutionary Computation, IEEE Transactions on*, 10(2):124--144.

[46] Sundar, S. and Singh, A. (2010). A swarm intelligence approach to the quadratic minimum spanning tree problem. *Information Sciences*, 180(17):3182--3191.

[47] Wang, D. and Kleinberg, R. (2009). Analyzing quadratic unconstrained binary optimization problems via multicommodity flows. *Discrete Applied Mathematics*, 157(18):3746 -- 3753.

[48] Zhout, G. and Gen, M. (1998). An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers & Operations Research*, 25(3):229--237.