Polyhedral Results, Branch-and-cut and Lagrangian Relaxation algorithms for the Adjacent Only Quadratic Minimum Spanning Tree Problem

This manuscript was submitted to Networks on December 2015

Dilson Lucas Pereira
dilson.pereira@dcc.ufla.br
Universidade Federal de Lavras
Departamento de Ciência da Computação
Lavras, Brasil

Alexandre Salles da Cunha
acunha@dcc.ufmg.br
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte, Brasil

Abstract

Given a complete and undirected graph $G$, the Adjacent Only Minimum Spanning Tree Problem (AQMSTP) consists of finding a spanning tree that minimizes a quadratic function of its adjacent edges. The strongest AQMSTP linear integer programming formulation in the literature works in an extended variable space, using exponentially many decision variables assigned to the stars of $G$. In this paper, we characterize two families of facet defining inequalities by investigating the projection of that formulation onto the space of the canonical linearization variables. On the algorithmic side, we introduce four new branch-and-bound (BB) algorithms. Three of them are branch-and-cut algorithms based on the
inequalities characterized by projection. The fourth is based on a Lagrangian relaxation scheme devised for the star reformulation. Two of the branch-and-cut algorithms provide very good results, almost always dominating the previously best algorithm for the problem. The Lagrangian relaxation based BB provides even better results. It manages to solve all previously solved AQMSTP instances in the literature in about one tenth of the time needed by its competitors.

**Keywords:** adjacent only minimum spanning tree problem, projection, Branch-and-cut algorithms, Lagrangian Relaxation

## 1 Introduction

Given a complete and undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, and costs $\{c_e \in \mathbb{R} : e \in E\}$ associated to the edges of $E$, the minimum spanning tree problem (MSTP) consists of finding a spanning tree $T = (V, E_T)$ of $G$ with minimum cost $C(T) = \sum_{e \in E_T} c_e$. MSTP can be solved in polynomial time, for example, by the algorithms of Prim [19] or Kruskal [8].

Given costs $\{q_{ef} \in \mathbb{R} : e \in E, f \in E\}$ assigned to pairs of edges of $E$, the quadratic minimum spanning tree problem (QMSTP) is an NP-Hard MSTP variation [2] that seeks for a spanning tree $T$ of $G$ that minimizes the cost function $Q(T) = \sum_{e \in E_T} \sum_{f \in E_T} q_{ef}$. Note that if $q_{ef} = 0$ for all pairs of distinct edges, QMSTP reduces to MSTP. Applications of QMSTP arise, for example, in wireless telecommunication networks, where data might be transmitted by radio waves over a spanning tree topology. In such an application, quadratic costs model the communication interference between edges sharing the same radio frequency. The goal is then to find a spanning tree that minimizes the total interference. Formulations, exact, and heuristic solution algorithms for QMSTP were studied in [2, 3, 12, 18, 20].

We consider a particular case of QMSTP, named the adjacent only QMSTP (AQMSTP), which is also NP-Hard [2]. In the AQMSTP case, $q_{ef} = 0$ holds for all pairs of non-adjacent edges. One example of an AQMSTP application appears in the design of telecommunication spanning trees. Besides the cost incurred by installing each tree edge, additional costs may arise if adjacent edges involve different communication technologies. These extra costs account for the additional equipment that needs to be in place in order to convert data going from one edge to the other. AQMSTP was investigated in [2, 11, 16, 17].

AQMSTP is related to the symmetric quadratic traveling salesman problem [5] (SQTSP). Differently from the widely known TSP, in the SQTSP case, the cost of a cycle does not depend exclusively on pairs of successive nodes traversed by the salesman. Instead, it depends on each three successive nodes in the tour. Therefore, SQTSP consists of finding a Hamiltonian cycle of $G$ that minimizes a quadratic function of the adjacent edges in the cycle. As in the AQMSTP case, quadratic costs for pairs of non-adjacent edges are zero.

The star reformulation introduced by Pereira et al. [16, 17] is the strongest known linear integer programming formulation for AQMSTP. Besides binary variables assigned to the edges in $E$, it uses exponentially many decision vari-
ables assigned to the stars of $G$; a star being any subset of edges incident to a given vertex. Two AQMSTP exact solution approaches based on the star reformulation were introduced in [17]: a branch-and-cut-and-price and a branch-and-cut method. They differ in the way of handling these exponentially many variables. The branch-and-cut-and-price algorithm solves a column generation problem, where the stars are dynamically generated. The branch-and-cut algorithm employs a cutting plane method that operates under a compact variable space and dynamically separates projection cuts, obtained after projecting out the variables assigned to the stars of $G$. These two algorithms cannot systematically solve AQMSTP instances with more than 50 vertices within a reasonable amount of time.

In this paper, we investigate the projection cone associated to the stars of $G$ and characterize two families of valid inequalities. One of them is proven to be facet defining for the convex hull of AQMSTP feasible solutions. We also lift the other family into facet defining inequalities. Although the inequalities we characterized do not fully describe the whole set of projection cuts (some cuts associated to some extreme rays are missing), a very strong formulation, defined in a compact variable space, can be obtained with their use. On the algorithmic side, we introduce four new exact solution approaches for AQMSTP. Three of them are branch-and-cut methods that separate the valid inequalities characterized by projection. The fourth method is a branch-and-bound (BB) algorithm that uses a new Lagrangian relaxation scheme, also based on the star reformulation. These algorithms significantly improve on previous AQMSTP exact solution approaches. Thanks to the lower computational effort needed to separate the facet defining valid inequalities, two of the branch-and-cut algorithms provide very good computational results, almost always dominating the best algorithm in [17]. The Lagrangian relaxation based BB algorithm provides even better results, standing out as the best known exact algorithm for the problem. It solves all existing AQMSTP instances in the literature in nearly one tenth of the time needed by its competitors.

The remainder of the paper is organized as follows. In Section 2, we review the existing AQMSTP formulations and algorithms in the literature. In Section 3, we provide the main theoretical results of the paper. In Sections 4 and 5, we respectively discuss the branch-and-cut and the Lagrangian based BB methods. Computational experiments are presented next, in Section 6. We close the paper in Section 7, offering some conclusions.

2 Literature Review

In what follows, let $g : E \to \{1, 2, \ldots, m\}$ be any bijective function. We write $e < f$ if and only if $g(e) < g(f)$. Given a set $V' \subseteq V$, let $E(V') = \{\{u, v\} \in E : u \in V', v \in V'\}$ be the subset of edges of $E$ with both endpoints in $V'$ and $\delta(V') = \{\{u, v\} \in E : u \in V', e \notin V'\}$ be the subset of edges with exactly one endpoint in $V'$. Given $v \in V$, we write $\delta(v)$ instead of $\delta(\{v\})$. For any linear integer formulation $P$ for AQMSTP, $z(P)$ denotes its linear programming (LP)
relaxation bound.

2.1 AQMSTP Formulations and Solution Approaches

Consider a set \( x = \{ x_e \in \{0, 1\} : e \in E \} \) of binary decision variables, such that \( x_e = 1 \) if and only if edge \( e \) is selected to be in the tree we are looking for. A binary quadratic formulation for QMSTP is:

\[
\min \sum_{e \in E} \sum_{f \in E} q_{ef} x_e x_f, \quad (1)
\]

subject to

\[
\sum_{f \in E} x_f = n - 1, \quad (2)
\]

\[
\sum_{f \in E(W)} x_f \leq |W| - 1, \quad W \subset V, |W| \geq 2, \quad (3)
\]

\[
x_f \in \{0, 1\}, \quad f \in E. \quad (4)
\]

The replacement of (4) by

\[
0 \leq x_f \leq 1, \quad f \in E, \quad (5)
\]

in (2)-(4) results in the convex hull of the incidence vectors of the spanning trees of \( G \) [4], which we denote by \( X \).

A linear integer programming formulation can be obtained from (1)-(4), by introducing linearization variables \( y = \{ y_{ef} \in \{0, 1\} : e \in E, f \in E \} \). Each variable \( y_{ef} \) replaces the product of \( x_e \) by \( x_f \). As such, \( y_{ef} \) must assume value 1 if and only if \( x_e = x_f = 1 \). To enforce that, additional linear constraints have to be added to the formulation. One possible set of constraints with that aim can be obtained by applying the reformulation-linearization technique (RLT) [1]. That is the basis for the QMSTP formulation below, introduced in [18]:

\[
(F^{RLT}) \quad \min \sum_{e \in E} \sum_{f \in E} q_{ef} y_{ef}, \quad (6)
\]

subject to

\[
x \in X, \quad (7)
\]

\[
\sum_{f \in E} y_{ef} = (n - 1)x_e, \quad e \in E, \quad (8)
\]

\[
\sum_{f \in E(W)} y_{ef} \leq (|W| - 1)x_e, \quad e \in E, W \subset V, |W| \geq 2, \quad (9)
\]

\[
y_{ef} \leq x_e, \quad e \in E, f \in E, \quad (10)
\]

\[
y_{ef} \geq 0, \quad e \in E, f \in E, \quad (11)
\]

\[
y_{ee} = x_e, \quad e \in E, \quad (12)
\]
Constraints (8)-(11) are obtained from (2), (3), and (5) by the application of a two-step procedure. The first step is the left multiplication of both sides of each constraint (2), (3), and (5) by $x_e$, for each $e \in E$. The second step is a linearization step, the replacement of each product $x_ex_f$ by a linearization variable $y_{ef}$. Formulation $F^{RLT}$ is not the full first level RLT reformulation, since the latter would also involve the multiplication of constraints (2), (3), and (5) by $1-x_e$, for each edge $e$. Nevertheless, computational experiments conducted in Pereira et al. [18] indicated that, for the AQMSTP case, the lower bound improvements obtained by adding the missing constraints do not pay off the additional computational cost they imply.

Constraints (13) are needed only if one chooses to explicitly distinguish $y_{ef}$ from $y_{fe}$. The distinction is useful, for example, when applying the Lagrangian relaxation scheme in [18]. Once (13) are relaxed in $F^{RLT}$, the resulting problem is polynomially solvable by an adaptation of the Gilmore-Lawler procedure [6, 9]. Pereira et al. [18] used this fact to develop a Lagrangian relaxation scheme where (13) are relaxed and dualized in a Lagrangian fashion and QMSTP lower bounds were obtained by subgradient optimization.

A similar Lagrangian relaxation approach was studied by Rostami and Malucelli [20]. However, they went further in the application of RLT and also investigated level 2 RLT bounds for QMSTP. Pereira et al. [18] also proposed a generalization of $F^{RLT}$ that leads to a hierarchy of QMSTP lower bounds of increasing strength. Instead of using only edges as modelling entities for QMSTP, the model combines larger subgraphs of $G$ into spanning trees. The larger these subgraphs, the stronger the bounds. A Lagrangian relaxation scheme was developed for one of its particular cases, that of combining adjacent edges of $G$ to build spanning trees. Using such an idea, Pereira et al. [18] obtained stronger QMSTP lower bounds than the level 2 RLT bounds in [20].

QMSMT solution approaches like those in [18, 20] could obviously be applied to AQMSTP. RLT bounds in [18], however, tend to be weak. QMSTP RLT bounds could be improved, for example, by the hierarchy of formulations in Pereira et al. [18] or by higher order RLT reformulations. However, the computational effort needed to evaluate these bounds would not pay off the relatively poor bound improvements they would likely imply. Bearing that in mind, Pereira et al. [16, 17] introduced the star reformulation. Being specific for the adjacent only case, the reformulation overcomes these issues to a certain extent.

To define the star reformulation, denote by $S^v = \{E' \subseteq \delta(v)\}$ the set of all stars centred at vertex $v$. Let $S = \cup_{v \in V} S^v$ be the union of all such sets. The reformulation employs the variables $x = \{x_e \in \{0,1\} : e \in E\}$ defined before and new binary variables $t = \{t_H \in \{0,1\} : H \in S\}$. A variable $t_H$ with $H \in S^v$ assumes value 1 if and only if $H$ is the set of edges incident to $v$ in the solution.

\begin{align}
  y_{ef} &= y_{fe}, \quad e,f \in E, e < f, \\
  x_e &\in \{0,1\}, \quad e \in E.
\end{align}
The cost of a star \( H \) is defined as:

\[
q_H = \sum_{e \in H} \left( \sum_{f \in H, e < f} (q_{ef} + q_{fe}) + \sum_{e \in H} \frac{1}{2} q_{ee} \right).
\]

(15)

The star reformulation we present below also makes use of the \( y \) variables, defined for pairs of adjacent edges. However, it makes no distinction between \( y_{ef} \) and \( y_{fe} \). Thus, unless stated otherwise, from now on we let \( y = \{ y_{ef} \in \{0, 1\} : e \in \delta(v), f \in \delta(v), e < f, v \in V \} \). The AQMSTP star formulation is:

\[
(F_{\text{STAR}}) \quad \min \sum_{H \in S} q_H t_H,
\]

subject to:

\[
\sum_{H \in S_v} t_H = 1, \quad v \in V,
\]

(17)

\[
\sum_{H \in S_v, e \in H} t_H = x_e, \quad e \in \delta(v), v \in V,
\]

(18)

\[
\sum_{H \in S_v, e \in H, f \in H} t_H = y_{ef}, \quad e, f \in \delta(v), e < f, v \in V,
\]

(19)

\[
x \in X,
\]

(20)

\[
x_e \in \{0, 1\}, \quad e \in E,
\]

(21)

\[
0 \leq t_H \leq 1, \quad H \in S,
\]

(22)

\[
y_{ef} \text{ unrestricted}, \quad e, f \in \delta(v), e < f, v \in V.
\]

(23)

Note that the objective function (16) is a linear function of the costs of the stars of \( G \). Since only adjacent edges give rise to quadratic costs, the stars capture all the quadratic cost involved in spanning trees of \( G \).

Constraints (17) guarantee that exactly one star will be assigned to each vertex. Whenever an edge \( e = \{u, v\} \) is included in a spanning tree, \( e \) must be included in the stars centred at \( u \) and \( v \). If \( e \) does not belong to the tree, the two stars are not allowed to include that edge. Such a condition is enforced by constraints (18). Likewise, if \( y_{ef} = 1 \) for a pair of adjacent edges \( e \) and \( f \), constraints (19) guarantee that there must be a star including both edges.

Neither linearization variables \( y \) nor constraints (19) were used in the original star reformulation in [16, 17]. We present the reformulation over an extended \((x, y, t)\) variable space, in order to project it onto the usual \((x, y)\) linearization space. In what follows, we allow ourselves the lack of formalism, when we state that the branch-and-cut-and-price and the branch-and-cut algorithms in [17] were based on \( F_{\text{STAR}} \), when, in reality, they are based on a star reformulation that does not include neither \( y \) nor (19).
The branch-and-cut-and-price algorithm in [17], from now on denoted by BCP[17], separates SECs (3) and solves a column generation pricing problem associated with variables \( t \). That pricing problem was formulated as a unconstrained quadratic binary problem (UQBP) [21], widely known to be NP-Hard. A specialized branch-and-cut method was implemented to solve UBQP. BCP[17] was capable of solving AQMSTP instances with up to 40 vertices.

An \( F^{STAR} \) polytope-wise equivalent formulation can be obtained by projecting out variables \( t \). To that aim, let \( p^v = \{ p^v \in \mathbb{R} : v \in V \} \) denote a set of unrestricted real variables. Note that the objective function in \( F^{STAR} \) can be replaced by \( \min \sum_{v \in V} p^v \), provided that constraints

\[
p_v \geq \sum_{H \in S^v} q_H t_H, \quad v \in V,
\]

are appended to the formulation. Consider then the following sets of Farkas multipliers:

\[
\alpha = \{ \alpha^v \in \mathbb{R}_+ : v \in V \}, \quad \text{associated with (24)},
\]

\[
\beta = \{ \beta^v \in \mathbb{R} : v \in V \}, \quad \text{associated with (17)},
\]

\[
\gamma = \{ \gamma_e^v \in \mathbb{R} : e \in \delta(v), v \in V \}, \quad \text{associated with (18), and finally}
\]

\[
\theta = \{ \theta_{ef}^v \in \mathbb{R} : e \in \delta(v), f \in \delta(v), e < f, v \in V \}, \quad \text{associated with (19)}.
\]

Projecting out variables \( t \), we obtain:

\[
(F^{STAR}_{PROJ}) \quad \min \sum_{v \in V} p^v
\]

subject to:

\[
\alpha^v p^v \geq \beta^v + \sum_{e \in \delta(v)} \gamma_e^v x_e + \sum_{e \in \delta(v)} \sum_{f \in \delta(v), e < f} \theta_{ef}^v y_{ef}, \quad (\alpha, \beta, \gamma, \theta) \in R(C^v), \quad v \in V,
\]

\[
x \in X,
\]

\[
x_e \in \{0, 1\}, \quad e \in E,
\]

\[
p_v \text{ unrestricted}, \quad v \in V.
\]

In \( F^{STAR}_{PROJ} \), \( R(C^v) \) denotes the set of extreme rays of the projection cone \( C^v \):

\[
(C^v) \quad - q_H \alpha^v + \beta^v + \sum_{e \in H} \gamma_e^v + \sum_{e \in H, f \in H, e < f} \theta_{ef}^v \leq 0, \quad H \in S^v,
\]

\[
\alpha^v \geq 0,
\]

\[
\beta^v \text{ unrestricted},
\]

7
\[ \gamma^v_e \text{ unrestricted, } e \in \delta(v), \quad (33) \]
\[ \theta^v_{ef} \text{ unrestricted, } e, f \in \delta(v), e < f. \quad (34) \]

Pereira et al. [17] proposed a cutting plane algorithm for evaluating the LP relaxation bounds from \( F_{\text{STAR}}^{\text{PROJ}} \). That algorithm separates constraints (3) and (26). In order to separate the latter set, the algorithm solves an optimization problem defined over the dual of the projection cone \( C^v \). That dual problem actually corresponds to finding the best setting for variables \( t \), given a feasible solution \( \bar{x} \) for the relaxation of \( F_{\text{STAR}}^{\text{PROJ}} \). Therefore, the dual problem can be solved by a column generation approach whose pricing problem can be solved by the same UBQP algorithm used to solve the pricing problems in BCP[17]. That cutting plane algorithm was embedded in an AQMSTP branch-and-cut method, providing the best previously available computational results for the problem. The branch-and-cut algorithm in Pereira et al. [17], from now on denoted by BC[17], solved AQMSTP instances with up to 50 vertices to proven optimality.

3 Polyhedral study

The aim of this section is to characterize particular classes of AQMSTP valid inequalities enclosed in (26), in order to obtain an AQMSTP formulation as strong as \( F_{\text{STAR}}^{\text{PROJ}} \) as possible. Separating such inequalities instead of (26) might help in reducing the computational effort needed to evaluate (or to approximate) bounds \( z(F_{\text{STAR}}^{\text{PROJ}}) \). That may be particularly true provided (i) these specific inequalities are strong, hopefully facet defining, and (ii) in practice, their separation can be carried out faster than (26). These issues are discussed next.

3.1 Valid inequalities

Two particular cases of (26) are
\[ y_{ef} \leq x_h, \quad e, f \in \delta(v), e < f, h \in \{e, f\}, v \in V, \quad (35) \]
and
\[ \sum_{e \in E'} \sum_{f \in E', e < f} y_{ef} \geq \sum_{e \in E'} x_e - 1 \quad E' \subseteq \delta(v), v \in V. \quad (36) \]

Inequalities (36) are obtained when, for any \( E' \subseteq \delta(v) \), we set \( \beta^v = -1 \), \( \gamma^v_e = 1 \) for all \( e \in E' \) and \( \theta^v_{ef} = -1 \) for all \( f \in E' \) with \( e < f \). They are a particular case of clique inequalities [13], proven to be facet defining for the boolean quadric forest polytope [10]. Note that this fact does not necessarily imply they are also facet defining for the AQMSTP polytope, as we do show later on.
Inequalities (35) can be obtained by setting \( \gamma_e^v = -1 \) and \( \theta_{ef}^v = 1 \) for edges \( e \) and \( f \) in \( \delta(v) \) with \( e < f \). They are not facet defining, since they are dominated by the lifted QMSTP valid inequality

\[
y_{ef} + y_{fg} \leq x_h, \quad e = \{u, v\}, f = \{v, w\}, g = \{u, w\},
\]

\[
e \in \delta(v), f \in \delta(v), e < f, h \in \{e, f\}, v \in V. \quad (37)
\]

Note that (37) are valid, since \( y_{ef} = y_{fg} = 1 \) would imply \( x_e = x_f = x_g = 1 \) and the solution would not be cycle free. Indeed, (37) are a particular case of the RLT inequalities (9), defined for \( W = \{u, v, w\} \). They also appear in the context of SQSTP, and were proven to be facet defining for the convex hull of SQSTP feasible solutions in [5].

### 3.2 Facet Proofs

Recall that \( G \) was assumed to be complete. Denote by \( A \) the convex hull of all \((x, y) \in B^{\frac{n(n-1)}{2}} \times R^{\frac{n(n-1)(n-2)}{2}}\) such that \( x \) is the incidence vector of a spanning tree of \( G \) and \( y_{ef} = x_ex_f \) for all \( e \) and \( f \) in \( \delta(v) \) with \( e < f \) and \( v \in V \). Given a valid inequality

\[
\sum_{e \in E} a_e x_e + \sum_{v \in V} \sum_{e \in \delta(v)} \sum_{f \in \delta(v), e < f} b_{ef} y_{ef} \leq c \quad (38)
\]

for \( A \), two additional equivalent compact forms are used here, to make reference to (38): \( ax + by \leq c \) and \((a, b, c) \in R^n \times R^{\frac{n(n-1)(n-2)}{2}} \times R\).

The face of \( A \) induced by an AQMSTP valid inequality \((a, b, c)\) is defined as the polyhedral set \( F(a, b, c) = \{(x, y) \in A : ax + bx = c\} \). Due to (2), \( A \) is not a full dimensional polytope, which implies that any linear combination between (2) and \( ax + by = c \) yields \( F(a, b, c) \).

In order to prove that a facet \( F(a, b, c) \) is a facet, we consider a facet \( F(a, b, c) \) of \( A \) containing \( F(a^*, b^*, c^*) \), whose coefficients \( a, b, \) and \( c \) are unknown. Then, we select points in \( F(a^*, b^*, c^*) \), which are also in \( F(a, b, c) \), and use them to deduce \( a, b, \) and \( c \). If \( F(a^*, b^*, c^*) \) is a facet, we can show that \( ax + by = c \) is a linear combination of \( a^*x + b^*y = c^* \) and (2), which implies \( F(a^*, b^*, c^*) = F(a, b, c) \). Further details on this proof technique can be found in [22], Theorem 3.6, pp 91.

The following polyhedral results hold true for AQMSTP.

**Theorem 1.** Let \( G \) be a complete graph, with \( n \geq 6 \) vertices. Inequalities (37) are facet defining for \( A \).

**Proof.** Consider (37) defined for \( e = e^*, f = f^* \), and \( g = g^* \). Assume \( h = e^* \).

We thus consider the face \( F(a^*, b^*, 0) = \{(x, y) \in A : y_{e^*f^*} + y_{e^*g^*} - x_{e^*} = 0\} \)

induced by (37).

The proof will make use of the spanning trees of \( G \) indicated in Figure 1. For all of them, edges drawn with a continuous line belong to the corresponding tree, while dashed edges do not.
We first show that $F(a^*, b^*, 0)$ is a proper face of $A$. It suffices to note that the inequality is slack for any spanning tree that includes edge $e^*$ but does not include edges $f^*$ and $g^*$.

Let $ax + by = c$ denote a hyperplane that defines a facet $F(a, b, c)$ of $A$, such that $F(a^*, b^*, 0) \subseteq F(a, b, c)$. In what follows, for simplicity of notation, we might refer to $b_{e\neq f}$, $e < f$, as either $b_{ef}$ or $b_{fe}$.

We first show that $b_{ef} = 0$ whenever $\{e, f\} \neq \{e^*, f^*\}$ and $\{e, f\} \neq \{e^*, g^*\}$.

To that aim, we consider 3 cases:

1: Neither $e$ nor $f$ is in $\{e^*, f^*, g^*\}$. We consider three subcases:

1.1: Edges $e$ and $f$ are not adjacent to any edge in $\{e^*, f^*, g^*\}$.

Pick the spanning tree $T_1$ indicated in Figure 1-(a). Note that edges $e$ and $f$ are not adjacent to any edge in $\{e^*, f^*, g^*\}$. Let $T_2$ be obtained from $T_1$ by inserting $e$ and removing $e'$. Note that both $T_1$ and $T_2$ satisfy (37) at equality and thus both imply points in $F(a, b, c)$. Comparing (39) defined for $T_1$ and $T_2$, we obtain

$$a_e + b_{eg'} = a_{e'} + b_{e'g'}.$$  \hspace{1cm} (40)

Let $T_3$ be a spanning tree obtained from $T_1$ by inserting $f$ and removing $f'$. Obtain $T_4$ from $T_3$ by inserting $e$ and removing $e'$. Both $T_3$ and $T_4$ satisfy (39) at equality. Comparing (39) defined for $T_3$ and $T_4$, we obtain

$$a_e + b_{eg'} + b_{ef} = a_{e'} + b_{e'g'}.$$  \hspace{1cm} (41)

Subtracting (40) from (41), we obtain $b_{ef} = 0$.

For all the remaining cases that follow, we apply the same edge exchange argument, to obtain expressions akin to (40) and (41) that allow to show that the desired coefficient $b_{ef}$ is zero valued. In each subcase, we start with a different initial spanning tree $T_1$ and construct $T_2$, $T_3$, and $T_4$ as indicated above: $T_2$ is obtained from $T_1$ by the insertion of $e$ and removal of $e'$, $T_3$ is obtained from $T_1$ by the insertion of $f$ and removal of $f'$, and $T_4$ is obtained from $T_3$ by the insertion of $e$ and removal of $e'$. All spanning trees $T_1$, ..., $T_4$ constructed in such a way will satisfy (37) at equality. Aiming to abbreviate the exposition, in the subcases that follow, we only indicate $T_1$.

1.2: Either $e$ or $f$, but not both, is adjacent to two edges in $\{e^*, f^*, g^*\}$.

We assume without loss of generality that $e$ is the edge that is adjacent to the two edges in $\{e^*, f^*, g^*\}$. We have two subcases to consider:

1.2.1: Edge $e$ is adjacent to $g^*$ and $f^*$. Let $T_1$ be the spanning tree indicated in Figure 1-(b).
1.2.2: Edge \( e \) is adjacent to either \( e^* \) and \( f^* \) or to \( e^* \) and \( g^* \). Without loss of generality, we assume the first case applies. Let \( T_1 \) be given by the solution indicated in Figure 1-(c).

1.3: Both \( e \) and \( f \) are adjacent to two edges in \( \{e^*, f^*, g^*\} \). We have two subcases to consider:

1.3.1: Both \( e \) and \( f \) are adjacent to \( f^* \) and \( g^* \). \( T_1 \) is indicated in Figure 1-(d).

1.3.2: Both \( e \) and \( f \) are adjacent to either \( e^* \) and \( f^* \) or to \( e^* \) and \( g^* \). We assume that the former applies. \( T_1 \) is indicated in Figure 1-(e).

2: Either \( e \) or \( f \), but not both, is in \( \{e^*, f^*, g^*\} \). Without loss of generality, we assume that \( e \in \{e^*, f^*, g^*\} \). We have two subcases:

2.1: \( e = e^* \). \( T_1 \) is indicated in Figure 1-(f).

2.2: Either \( e = f^* \) or \( e = g^* \). We assume that \( e = f^* \). Pick \( T_1 \) as the tree indicated in Figure 1-(g).

3: \( \{e, f\} = \{f^*, g^*\} \). \( T_1 \) is given in Figure 1-(h).

We now show that \( a_e = a_f \) for all \( e \) and \( f \), both different of \( e^* \). To that aim, pick a spanning tree \( T_1 \) with both \( e \) and \( f \), but without \( e^* \). Also, let \( T_1 \) be such that one of the endpoints of \( e \), say \( v_e \), is a leaf. We can connect \( v_e \) to one of the endpoints of \( f \), say \( v_f \), to create a cycle containing \( f \) and \( e \). Let \( g = \{v_e, v_f\} \) be such an edge. We can also assume that \( T_1 \) is such that \( e^* \neq g \). Obtain \( T_2 \) from \( T_1 \) by inserting \( g \) and removing \( f \). Obtain \( T_3 \) from \( T_1 \) by inserting \( g \) and removing \( e \). Note that \( T_1, T_2, \) and \( T_3 \) all satisfy (37) at equality, and thus imply points in \( F(a, b, c) \). Writing (39) for \( T_1, T_2, \) and \( T_3 \) and using the fact that \( b_{\{e^*, f\}} = 0 \) whenever \( \{\overline{e^*}, \overline{f}\} \neq \{e^*, f^*\} \) and \( \{\overline{e^*}, \overline{g}\} \neq \{e^*, g^*\} \), we conclude that \( a_e = a_f \).

To show that \( b_{e^*, f^*} = b_{e^*, g^*} \), we can apply similar edge exchange operations by taking two spanning trees. One contains both \( e^* \) and \( f^* \) and the other contains both \( e^* \) and \( g^* \). It is just a matter of writing (39) for them to get the desired result.

Using the coefficients we obtained above, we can write (39) as

\[
\sum_{e \in E} a'_e x_e + a''_e x_e^* + b_{e^*, f^*} y_{e^*, f^*} + b_{e^*, g^*} y_{e^*, g^*} = c, \tag{42}
\]

where \( a'_e = a_e \) for all \( e \neq e^* \) and \( a''_e = a_e^* \). Picking any spanning tree without \( e^* \), we obtain \( c = (n - 1)a' \). Now picking a tree that includes both \( e^* \) and \( f^* \), we get \( a''_e = -b_{e^*, f^*} \). Similarly, we also obtain \( a''_f = -b_{e^*, g^*} \). Thus, (39) can be written as

\[
\sum_{e \in E} a'_e x_e + a''_e x_e^* = a'(n - 1) + a''_e f^* y_{e^*, f^*} + a''_e g^* y_{e^*, g^*}, \tag{43}
\]

which is \( a' \) times (2) plus \( -a'' \) times \( y_{e^*, f^*} + y_{e^*, g^*} - x_{e^*} = 0 \). Thus, \( F(a^*, b^*, 0) = F(a, b, c) \). \( \square \)
Figure 1: Auxiliary figures for the proof of Theorem 1.
Corollary 1. The polytope \( A \) has dimension \( \frac{n(n-1)}{2} + \frac{n(n-1)(n-2)}{2} - 1 \).

Proof. Given a facet defined by an inequality (37), the proof of Theorem 1 shows that the facet can be represented only by combinations of (37) and (2). This shows that (2) is the only equality satisfied by all points in \( A \). \( \square \)

Inequalities (36) are also facet defining, as the next result states.

Theorem 2. Let \( G \) be a complete graph, with \( n \geq 6 \) vertices. Inequalities (36) are facet defining for \( A \).

Proof. Consider a particular case of (36) defined for \( E' = E^* \) and \( v = v^* \). The implied face of \( A \) is then \( F(a^*,b^*,c^*) = \{(x,y) \in A : \sum_{e \in E^*} \sum_{f \in E^*}, e < f \ y_{ef} = \sum_{e \in E^*} x_e - 1\} \). The proof will make use of the spanning trees of \( G \) presented in Figure 2. In each figure, continuous edges are in the tree, dashed edges are not, thick edges are in \( E^* \), thin edges are not. We assume that \( E^* \neq \emptyset \) since the case \( E^* = \emptyset \) implies a face that does not define a proper subset of \( A \).

![Figure 2: Auxiliary figures for the proof of Theorem 2.](image-url)
any spanning tree that does not include edges in $E^*$. In case $E^* = \delta(v)$ the inequality is slack for any spanning tree that includes three or more edges from $E^*$. Thus, it follows that $F(a^*, b^*: c^*) \neq \mathcal{A}$.

Now, assume that

$$ax + by = c$$

defines a facet $F(a, b, c)$ of $\mathcal{A}$ that is a superset of $F(a^*, b^*, c^*)$. In what follows, for simplicity of notation, we might refer to $b_{ef}, e < f$, as either $b_{ef}$ or $b_{fe}$.

We will first show that $b_{ef} = 0$ for all adjacent $e$ and $f$ not both in $E^*$. This is divided into two cases:

1: Neither $e$ or $f$ is in $E^*$. This case is further divided into three cases:

1.1: Vertex $v^*$ is not an endpoint of either $e$ or $f$. Let $e$ and $f$ be adjacent edges not incident to $v^*$. We can construct a spanning tree $T_1$ such as the one indicated in Figure 2-(a). Now, we insert $e$ and remove $e'$ from $T_1$ to obtain $T_2$. Both $T_1$ and $T_2$ imply points in $F(a^*, b^*, c^*)$. Subtracting (44) implied by $T_2$ from (44) implied by $T_1$, we obtain

$$a_{e'} + b_{e'h} = a_e + b_{eg}. \quad (45)$$

Let $T_3$ be the tree obtained from $T_1$ by inserting $f$ and removing $f'$. Obtain $T_4$ from $T_3$ by inserting $e$ and removing $e'$. Both $T_3$ and $T_4$ imply points in $F(a^*, b^*, c^*)$. Subtracting (44) implied by $T_4$ from (44) implied by $T_3$, we obtain

$$a_{e'} + b_{e'h} = a_e + b_{eg} + b_{ef}. \quad (46)$$

Subtracting (45) from (46) we obtain $b_{ef} = 0$. Note that in $T_1$, $g$ is in $E^*$, but this argument will also work in case it is not.

For all $e$ and $f$ in subsequent cases in 1 and 2, we will show how to obtain trees $T_1, \ldots, T_4$, all satisfying (36) at equality, i.e., implying points in $F(a^*, b^*, c^*)$, over which we can apply the same argument above to conclude that $b_{ef} = 0$. For each case, we will show how to obtain the initial spanning tree $T_1$. The spanning tree $T_2$ we will be constructed by inserting $e$ and removing $e'$ in $T_1$, $T_3$ will be obtained by inserting $f$ and removing $f'$ in $T_1$, and $T_4$ will be obtained from $T_3$ by inserting $e$ and removing $e'$.

1.2: Vertex $v^*$ is an endpoint of either $e$ or $f$, but not both. We assume without loss of generality that $v^*$ is an endpoint of $f$. Let $T_1$ be the tree indicated Figure 2-(b).

1.3: Vertex $v^*$ is an endpoint of both $e$ and $f$. Let $T_1$ be the tree given in Figure 2-(c).

2: One of them is in $E^*$, we assume without loss of generality that $f \in E^*$. We further divide this case in two subcases:

2.1 Edge $e$ is not incident to $v^*$. Let $T_1$ be the tree drawn in Figure 2-(d).
2.2 Edge \( e \) is incident to \( v^* \). Let \( T_1 \) be the tree in Figure 2-(e).

We now show that \( a_e = a_f \) for all \( e \) and \( f \) not in \( E^* \). To that aim, find a spanning tree \( T_1 \) satisfying (36) at equality, in which one endpoint of \( e \), say \( v_e \), and one endpoint of \( f \), say \( v_f \), are leaves, none of these two leaves being \( v^* \). Let \( T_2 \) be the tree obtained from \( T_1 \) by inserting \( g = \{v_e, v_f\} \) and removing \( f \) and let \( T_3 \) be the tree obtained from \( T_1 \) by inserting \( g \) and removing \( e \). Comparing (44) for \( T_1, T_2, \) and \( T_3 \), and using the fact that \( b_{ef} = 0 \) whenever \( e \) and \( f \) are not both in \( E^* \), we get \( a_e = a_f \).

To show that \( a_e = a_f \) whenever both \( e \) and \( f \) are in \( E^* \), let \( T_1 \) be a spanning tree with edge \( e = \{u, v^*\} \in E^* \) and all other vertices connected to \( u \). Insert \( f = \{v^*, w\} \in E^* \) and remove \( e \) to obtain \( T_2 \). Comparing (44) for \( T_1, T_2, \) and using the fact that \( b_{ef} = 0 \) whenever \( e \) and \( f \) are not both in \( E^* \), we obtain \( a_e = a_f \).

We now show that \( b_{ef} = b_{fg} \) for all distinct \( e, f, \) and \( g \) in \( E^* \). Let \( e = \{u_e, v^*\}, f = \{u_f, v^*\}, \) and \( g = \{u_g, v^*\} \) be distinct edges in \( E^* \). Let \( T_1 \) be a spanning tree containing both \( e \) and \( f \), where all other vertices different from \( u_e \) and \( u_f \) are connected to \( u_f \). Let \( T_2 \) be a spanning tree that includes \( f, g \), where all other vertices different from \( u_e \) and \( u_f \) are connected to \( u_f \). Comparing (44) for \( T_1, T_2, \) and using the fact that \( a_e = a_f \) for all \( e \notin E^* \), \( a_e = a_f \) for all \( e \in E^* \), \( b_{ef} = 0 \) for all pairs of edges \( e, f \) not both included in \( E^* \), we conclude that \( b_{ef} = b_{fg} \). Then, it is not difficult to check that \( b_{ef} = b_{gh} \) for all \( e, f, g, \) and \( h \) in \( E^* \), such that \( e \neq f \), and \( g \neq h \).

Equation (44) can now be re-written as

\[
\sum_{e \in E} a'_e x_e + \sum_{e \in E'} a''_e x_e + \sum_{e \in E', f \in E'} b''_{ef} y_{ef} = c, \tag{47}
\]

where \( a'_e = a_e \) for all \( e \in E \setminus E' \) and \( a'_e + a''_e = a_e \) for all \( e \in E' \).

Let \( T \) be an AQMSTP solution with only one edge in \( E^* \). Analyzing (47) for \( T \), we obtain \( a'' = (n - 1)a' = c \). Now, (47) can be re-written as

\[
\sum_{e \in E} a'_e x_e + \sum_{e \in E'} a''_e x_e + \sum_{e \in E', f \in E'} b''_{ef} y_{ef} = a'' \cdot (n - 1)a'. \tag{48}
\]

Let \( T \) be an AQMSTP solution with exactly two edges in \( E^* \). Again, analyzing (48) for \( T \), we obtain \( b'' = -a'' \), and (48) becomes

\[
\sum_{e \in E} a'_e x_e + \sum_{e \in E'} a''_e x_e = \sum_{e \in E', f \in E'} a'' y_{ef} + a'' \cdot (n - 1)a', \tag{49}
\]

which is \( a' \) times (2) plus \( a'' \) times (36) (stated as an equality). Thus, \( F(a^*, b^*, c^*) = F(a, b, c) \). \( \square \)
4 Branch-and-cut algorithms

The inequality (36) formulated for a given set \( E' = \{e, f\} \subseteq \delta(v) \) reads as 
\[ y_{ef} \geq x_e + x_f - 1. \]

Such an observation implies that, despite the fact that some projection cuts in (26) are not included in (36) and (35) (neither in (37)), AQMSTP can be formulated by replacing (26) by (35) and (36). Thus, a linear integer programming formulation for AQMSTP is given by:

\[
(F_{xy}) \quad \min \left\{ \sum_{v \in V} \sum_{e \in \delta(v)} \sum_{f \in \delta(v)} q_{ef} y_{ef} + \sum_{e \in E} q_{ee} x_e : (x, y) \text{satisfies (7), (35), (36) and } y \geq 0 \right\}. \tag{50}
\]

Define \( F_{xy}^+ \) as the formulation obtained when inequalities (37) replace (35) in \( F_{xy} \), and \( F_{RLT}^+ \) as the intersection of \( F_{RLT} \) with constraints (36). Note that \( F_{RLT}^+ \) involves at least \( m + m(m - 1)/2 = O(n^4) \) variables, much more than the \( m + n(n - 1)(n - 2)/2 = O(n^3) \) variables in \( F_{xy} \) and \( F_{xy}^+ \).

In this section, we introduce three branch-and-cut algorithms: BC, BC\(^+\) and BC\(\text{RLT}^+\). They are respectively based on formulations \( F_{xy} \), \( F_{xy}^+ \) and \( F_{RLT}^+ \). The motivation for introducing these methods comes from the fact that \( z(F_{xy}) \) provides very good approximations for \( z(F_{STAR}) \). We first discuss how the LP bounds provided by these formulations are evaluated by cutting plane algorithms. Then, we provide further implementation details for the branch-and-cut algorithms.

4.1 Cutting plane algorithms and separation procedures

The evaluation of \( z(F_{xy}) \) and \( z(F_{xy}^+) \) is conducted by LP cutting plane algorithms that separate the same sets of valid inequalities, namely SECs (3) and clique inequalities (36). The only difference between BC and BC\(^+\) is thus their initial LP relaxation: inequalities (35) are replaced by (37) in BC\(^+\). Thus, implementation details given here apply for both cutting plane methods.

SECs are separated in \( O(n^4) \) worst case time complexity, with the exact algorithm of Padberg and Wolsey [14]. Constraints (36), formulated for sets \( E' \) with \(|E'| = 2\) are explicitly added to the initial LP relaxation. We are not aware of polynomial time separation algorithms for (36) for general \( E' \). Thus, to separate these inequalities, we devised a specific enumerative exact separation procedure, that operates as follows.

For a given \( v \in V \), the separation of inequalities (36) can be carried out by formulating and solving a conveniently defined UQBP instance. Before indicating how that instance is obtained, we first formally define UQBP as an integer program. Let \( I \) be a ground set and let \( \{w_i \in \{0, 1\} : i \in I\} \) be a set of binary
decision variables defined over $I$. Let $A$ be a $|I| \times |I|$ real valued (cost) matrix. UQBP can be formulated as the following 0-1 quadratic program:

$$\min \left\{ \sum_{i \in I} \sum_{j \in I} A_{ij} w_i w_j : w_i \in \{0, 1\}, i \in I \right\}.$$  \hfill (51)

Assume that an LP relaxation for $F_{xy}$ (or for $F_{xy}^+$), possibly involving no inequalities (36) other than those defined by sets $E'$ of size two, was solved to proven optimality. Assume as well that $(x, y)$ denotes the optimal solution to such a relaxation. For each $v \in V$, the separation of (36) can be formulated as the UQBP instance given by setting $I = \delta(v)$, $A_{ee} = -\pi e$ for all $e \in \delta(v)$, and $A_{ef} = A_{fe} = \pi_{ef}/2$ for all pairs of edges $e$ and $f$ in $\delta(v)$ with $e < f$. Whenever the objective function value for the optimal solution to such an UQBP instance is less than $-1$, a violated inequality (36) is found.

The algorithm we devised for solving UQBP in order to separate (36) is given in Figure 3. It is suitable to solve UQBP instances where off-diagonal entries of $A$ are non-negative. In the algorithm, we denote the quadratic cost of $J \subseteq I$ by $A(J) = \sum_{i \in J} A_{ii} + \sum_{i \in J} \sum_{j \in J \setminus \{i\}} A_{ij}$. Input parameters for the algorithm are $I = \emptyset$ and $I^* = \emptyset$. The first, $I$, stands for a set of edges incident to $v$ that will be grown, in the hope of finding violated inequalities (36). The second, $I^*$, stores the subset of edges that implies the most violated inequality. Note that the set $I$ is expanded only if the inclusion of an edge $i \in I \setminus I$ to $I$ contributes to the violation of the corresponding inequality (36), i.e., only if $\sum_{j \in I}(A_{ij} + A_{ji}) + A_{ii} < 0$. The correctness of the algorithm follows from the fact that, since $A_{ij} + A_{ji} > 0$, if $\sum_{j \in I}(A_{ij} + A_{ji}) + A_{ii} \geq 0$ for some $i \notin I$ and if there is a violated inequality defined for $I^*$ with $i \in I^*$, there must also be an inequality defined for $I^* \setminus \{i\}$, at least as violated as the one defined for $I^*$.

For a given $(\pi, \pi)$ to be separated, the algorithm is called once for each vertex $v$. At the end of each of these calls, REC_ENUM outputs a set $I^*$ such that $w_i = 1$

\begin{figure}[h]
\begin{verbatim}
REC_ENUM(\pi, I^*)
{
    for all $i \in I \setminus \pi$ such that $i > j$ for all $j \in \pi$
    {
        if $\sum_{j \in I}(A_{ij} + A_{ji}) + A_{ii} < 0$
        {
            let $I' = \pi \cup \{i\}$
            if $A(I') < A(I^*)$ let $I^* = I'$
            REC_ENUM(I', I^*)
        }
    }
}
\end{verbatim}
\end{figure}

Figure 3: Exact separation algorithm for clique inequalities (36).
for all $i \in I^*$ and $w_i = 0$ for all $i \in I \setminus I^*$, corresponding to the optimal solution for that UQBP instance. Provided $I^* \neq \emptyset$, the inequality formulated in terms of $E' = I^*$ is appended to the LP relaxation.

Despite the fact that its running time grows exponentially with $n$, the separation procedure given in Figure 3 works very fast in practice. The reasons are the following. Since for its application off-diagonal costs need to be non-negative, the cost $q_H$ of a star $H$ increases rapidly with $|H|$, due to its quadratic nature. Therefore, one should expect that the optimal solutions for AQMSTP do not involve vertices with large degrees. We observed that the same applies for the subgraphs implied by the LP relaxations $(\overrightarrow{F}, \overrightarrow{G})$ to be separated. To further substantiate our claim about the practical performance of the separation procedure, note that edges $e$ with $\overrightarrow{F}_e = 0$ are never included in the growing set $T$. That applies since they never satisfy the condition $\sum_{j \in T}(A_{ij} + A_{ji}) + A_{ii} < 0$. Edges whose corresponding values $\overrightarrow{F}_e$ are small are also unlikely to grant such a condition. Overall, for the reasons outlined above, sets $T$ are actually kept very small in the course of the algorithm.

The cutting plane algorithm that evaluates $z(FRLT^+)$ separates SECs (3), (36) and (9). The first two classes of inequalities are separated as described before. Separation of constraints (9), which can also be seen as a particular case of SECs (3), is carried out in polynomial time, by means of a minor modification in the SEC separation algorithm of Padberg and Wolsey [14]. Details on that separation procedure can be found in [15].

**4.2 Further implementation details**

The three branch-and-cut algorithms were implemented by embedding the separation procedures described earlier as callback separation procedures in the IBM CPLEX branch-and-cut framework. Separation is conducted precisely the same way, throughout the entire enumeration search tree, making no distinction between root and non-root nodes. At each node, at the end of each separation round, all violated SECs (3) found by the algorithm in [14] are added to the LP relaxation at hand. For inequalities (9), the separation algorithm is called once for each edge $e$. The most violated inequality (9) found in each call is added to the relaxation. The separation algorithm of (36) is called once for each vertex $v$. In each of them, the most violated inequality is added to the model.

Apart from heuristics and cuts that were turned off, CPLEX’s default settings were used to manage the enumeration search tree. Thus, BC, BC+, and BCRLT+ branch on variables $x$, according to CPLEX default settings. There is no need to branch on variables $y$, since an integer vector $x$ implies an integer $y$.

Valid AQMSTP upper bounds, obtained with the multi-start QMSTP heuristic introduced in [17, 18], are provided for the three branch-and-cut algorithms, before the very first LP relaxation is solved.
5 A Lagrangian Relaxation Based BB Algorithm

In this section, we introduce an AQMSTP BB algorithm based on a Lagrangian relaxation scheme. This Lagrangian relaxation scheme relies on the original star formulation in [17], i.e., neither variables $y$ nor constraints (19) are present.

5.1 Lagrangian Relaxation bounds

The relaxation of constraints (18) decouples the problem into independent optimization problems, defined over $x$ and $t$. Thus, by relaxing (18) in a Lagrangian fashion and attaching Lagrangian multipliers $\gamma = \{\gamma_v^e \in \mathbb{R} : v \in V, e \in \delta(v)\}$ to them, the best attainable bounds are given by the Lagrangian Dual Problem

$$z_{LD} = \max_\gamma L(\gamma),$$

where

$$L(\gamma) = L'(\gamma) + \sum_{v \in V} L_v^u(\gamma),$$

$$L'(\gamma) = \min \left\{ \sum_{e=(u,v) \in E} (\gamma_u^e + \gamma_v^e)x_e : x \in X \right\}$$

and

$$L_v^u(\gamma) = \min \left\{ \sum_{H \in S^v} (q_H - \sum_{e \in H} \gamma_v^e)t_H : t_H \in \{0, 1\} \text{ satisfies (17), } H \in S^v \right\}.$$  (55)

For a given vector $\gamma$ of Lagrangian multipliers, the evaluation of AQMSTP Lagrangian lower bounds $L(\gamma)$ involves solving $n + 1$ independent subproblems. The first corresponds to compute $L'(\gamma)$, by solving an MSTP, under edge weights $\{(\gamma_u^e + \gamma_v^e) : e = \{u, v\} \in E\}$. The other $n$ subproblems consist of evaluating $\{L_v^u(\gamma) : v \in V\}$. For each $v \in V$, computing $L_v^u(\gamma)$ amounts to finding the star centred at $v$, of least Lagrangian modified cost.

Assuming $q_{ef} \geq 0$ for all adjacent pairs of distinct edges $e$ and $f$, as it is the case of all AQMSTP instances in the literature, problems $\{L_v^u(\gamma) : v \in V\}$ can be solved by algorithm $\text{RECENUM}$. Each corresponding UQBP instance is defined by taking $I = \delta(v)$, $A_{ee} = q_{ee} - \gamma_v^e$ for all $e \in \delta(v)$, and $A_{ef} = q_{ef}$ for all pairs of distinct edges $e, f \in \delta(v)$. Again, we expect $\text{RECENUM}$ to have a reasonable practical performance. If off-diagonal costs $\{q_{ef} : e \neq f\}$ were allowed to assume negative values, $\text{RECENUM}$ would be replaced by another algorithm and, possibly, the good practical performance of BB would deteriorate.

Regarding the strength of $z_{LD}$, we have the following result:

**Theorem 3.** $z_{LD} = z(F^{\text{STAR}})$.

**Proof.** Let $P^I$ denote the convex hull of the integer $(x, t)$ points satisfying (17), (20) and (22), i.e., $P^I$ is the convex hull of the incidence vector of all solutions
to the Lagrangian subproblem $L(\gamma)$. Let $P^2$ be the set of all real $(x,t)$ points satisfying (17), (20) and (22). Lagrangian duality theory states that $z_{LD}$ is equal to the optimal value of (16) over real $(x,t)$ in $P^1 \cap \{(x,t) : (x,t) satisfies (18)\}$. Thus, we can prove the theorem by showing that $P^1 = P^2$.

It is clear that $P^1 \subseteq P^2$. We now show the converse, $P^2 \subseteq P^1$. Assume $V = \{v_1, v_2, \ldots, v_n\}$ and let $X$ be the set of all spanning trees of $G$. Let $(\bar{x}, \bar{t})$ be a point in $P_2$. Since $\bar{x} \in X$, we can write $\bar{x}$ as a convex combination of incidence vectors of spanning trees of $G$, let $\pi_T$ be the multiplier of $T \in X$ in such a combination. Let $Z = X \times S^{v_1} \times S^{v_2} \times \cdots \times S^{v_n}$ be the set of all possible combinations consisting of one spanning tree and one star for each vertex. Given one such combination $C = \{T^C, H_1^C, H_2^C, \ldots, H_n^C\}$, let $\psi_C = \pi_T \times \bar{t}_{H_1^C} \times \bar{t}_{H_2^C} \times \cdots \times \bar{t}_{H_n^C}$ and observe that $\sum_{C \in Z} \psi_C = 1$ since $\sum_{T \in X} \pi_T = 1$ and $\sum_{H \in S^v} \bar{t}_H = 1$ for all $v \in V$. Note that $C$ implies a feasible solution $(x^C, t^C)$ in $P_1$, where $x^C$ is the incidence vector of $T^C$, $t_H^C = 1$ if $H \in C$, and $t_H^C = 0$ otherwise, for all stars $H \in S$. Now, let $(\bar{x}, \bar{t}) = \sum_{C \in Z} \psi_C (x^C, t^C)$. We have

$$\bar{x} = \sum_{C \in Z} \psi_C x^C = \sum_{T \in X} \pi_T \sum_{H_1 \in S^{v_1}} \bar{t}_{H_1} \sum_{H_2 \in S^{v_2}} \bar{t}_{H_2} \cdots \sum_{H_n \in S^{v_n}} \bar{t}_{H_n} x^T = \sum_{T \in X} \pi_T x^T = \bar{x}.$$

In an analogous way, we can prove $\bar{t} = \bar{t}$. Thus, $(\bar{x}, \bar{t}) = \sum_{C \in Z} \psi_C (x^C, t^C)$, which shows $P_2 \subseteq P_1$.

5.2 Implementation details

BB solves the Lagrangian Dual problem (52) by applying the Subgradient Method (SM) [7]. At the root node of the enumeration tree, an approximation of the bound $z_{LD}$ was obtained by implementing 5000 SM iterations. The SM parameter that controls the step size is initialized with value 2 and is halved after 100 SM iterations are past without increasing the best root node lower bound. For non-root BB nodes, only 100 SM iterations were implemented. The step size control parameter, also initialized with 2, is halved after every 10 consecutive iterations without improving the lower bound.

BB branches according to the following strategy. Let $\gamma^*$ be the best set of Lagrangian multipliers found by SM, at a given enumeration tree node. Assume that $(\bar{x}, \bar{t})$ denotes the optimal solution to the Lagrangian subproblem $L(\gamma^*)$ formulated and solved at that node. Let $\bar{T} = (V, E_{\bar{T}})$ denote the spanning tree of $G$ implied by $\bar{x}$, i.e., $E_{\bar{T}} = \{e \in E : \bar{x}_e = 1\}$. Likewise, define $\bar{H} = \cup_{v \in V} \bar{H}_v$, where $\bar{H}_v$ denotes the star centred at $v$ that corresponds to the optimal solution
to $L_u''(\gamma^*)$. BB branches on variable $x_\tau$, where

$$
\tau \in \arg \min \left\{ q_{ee} + \sum_{f \neq e} (q_{ef} + q_{fe}) : e = \{u, v\} \in C_1 \cup C_2 \right\}, \quad (56)
$$

$C_1 = \{e = \{u, v\} \in E_{\mathcal{T}} : e \notin \mathcal{H}^u \cup \mathcal{H}^v\}$ and $C_2 = \{e = \{u, v\} \notin E_{\mathcal{T}} : e \in \mathcal{H}^u \cup \mathcal{H}^v\}$ are candidate sets of branching edges. Ties are broken arbitrarily.

Once the branching edge $\tau = \{u, v\}$ is chosen, two new nodes are created. For one of them, we enforce that $e \in E_{\mathcal{T}}$, $e \in \mathcal{H}^u$ and $e \in \mathcal{H}^v$. BB easily handles these branching constraints, without destroying the structure of the Lagrangian scheme outlined above. More precisely, to enforce $e \in E_{\mathcal{T}}$, $e \notin \mathcal{H}^u$ and $e \notin \mathcal{H}^v$, it suffices to define $I = \{\tau\}$ when $\text{RECENUM}$ is called to calculate $L_u''(\gamma)$ and $L_u''(\gamma)$. To enforce $e \notin \mathcal{H}^u$ and $e \notin \mathcal{H}^v$, it suffices to define $I = \delta(u) \setminus \{\tau\}$ and $I = \delta(u) \setminus \{\tau\}$.

In order to save computer memory, BB implements a depth-first search policy. The QMSTP multi-start heuristic in [17, 18] is also called to provide initial upper bounds for SM and BB.

6 Computational experiments

All algorithms introduced here were implemented in C++ and compiled with g++ release 4.6.3, optimization flag -O3 turned on. Our computational experiments were carried out with an Intel XEON E5645 machine with 32GB of RAM and two six-core processors, each core running at 2.4GHz, under the Linux Ubuntu 12.04 LTS operating system. The IBM ILOG CPLEX 12.5 optimization package was used to implement the branch-and-cut methods. No multi-threading was used.

We initially provide computational results for test instances available in the literature [17] and, in Section 6.3, we introduce new ones. AQMSTP instances introduced in [17] comprise complete graphs with $n \in \{15, 20, 30, 40, 50\}$. For each value of $n$, 10 instances are available. Each one was generated by randomly choosing diagonal $\{q_{ee} : e \in E\}$ and off-diagonal $\{q_{ef} : e \in E, f \in E\}$ costs respectively from sets $\{0, 1, \ldots, 100\}$ and $\{0, 1, \ldots, 20\}$, with uniform probability. For each instance, a time limit of 10 CPU hours was imposed on the execution of each algorithm. Computational experiments reported in [17] were carried out under the same computational environment used here and the same time limit was used. Therefore, computational results obtained here and those reported in [17] are directly comparable.

6.1 AQMSTP lower bounds

In this section, we compare various AQMSTP formulations in terms of the strength of their lower bounds and the CPU times needed to evaluate them.
In order to shorten the presentation, we only provide results for AQMSTP instances with \( n = 20 \) and \( n = 50 \) vertices, in Tables 1 and 2. The \( n = 50 \) case corresponds to the largest size tested in [17], while \( n = 20 \) is the largest value of \( n \) for which all instances could be solved by all algorithms compared here.

In the first column of Table 1, we identify each of the \( n = 20 \) instances by an integer between 1 and 10. In the next two subsequent columns, under the heading “FRLT [18]”, we replicate the results reported in [17], for formulation FRLT. They were obtained by applying the QMSTP Lagrangian relaxation algorithm in [18] to AQMSTP. Two entries are given: the approximation for the LP lower bounds \( z(FRLT) \) as evaluated by that algorithm, and \( t \), the corresponding CPU times. The next three columns, under the heading “FST AR [17]”, present results for the AQMSTP cutting plane algorithm of [17]: the exact (Ext) LP lower bounds \( z(FST AR) \), as evaluated by that algorithm, an approximation (Apx) of \( z(FST AR) \), obtained by adding only cuts violated by at least one percent of the linear relaxation value at the moment of separation, and the CPU times \( t \) needed to provide that approximation for the bound. These approximated bounds are precisely the root node lower bounds evaluated by algorithm BC [17]. In the next six columns, similar information is provided for formulations \( Fxy \), \( Fxy^+ \), and \( FRLT^+ \). We provide the LP lower bounds implied by each formulation, and the CPU times needed to evaluate them by the cutting plane algorithms described in Section 4.1. In the last two columns of the table, we present \( z_{LD} \) and the CPU time, \( t \), taken to run 5000 SM iterations and to obtain (to approximate) that bound. All CPU times are reported in seconds.

Table 2 provides similar information for \( n = 50 \). However, results for \( FRLT^+ \) are not given since \( z(FRLT^+) \) could not be evaluated within the imposed time limit.

Results in Tables 1 and 2 suggest that bounds \( z(Fxy) \) are very close to \( z(FST AR) \). For 17 out of 20 instances, we found that \( \lceil z(Fxy) \rceil = \lceil z(FST AR) \rceil \) holds. Since the AQMSTP instances tested here involve integer costs, in practice, the two formulations were equally strong for these 17 cases. For the 3 others, the difference in favor of \( z(FST AR) \) was of one unity only. A comparison between \( z(Fxy^+) \) and \( z(Fxy) \) indicates that inequalities (37) slightly improve \( Fxy \) without any significant impact on the computational times.

Our computational experiments suggest that \( FRLT^+ \) is significantly stronger than \( FRLT \). For the \( n = 20 \) instances, values for \( z(FRLT^+) \) exceed \( z(FRLT) \) by 65.7\%, on the average. Since \( z(Fxy) \) and \( z(FST AR) \) are very close and inequalities (35) are already included in \( FRLT \), we conclude that clique inequalities (36) are of crucial importance for the strength of \( FST AR \). Their inclusion in \( FRLT \) allowed bounds \( z(FRLT^+) \) to slightly surpass \( z(Fxy) \) and \( z(FST AR) \). Note that a huge computational effort is required to compute \( z(FRLT^+) \), mainly due to the large number \( O(n^4) \) of variables explicitly kept in the formulation. Indeed, for the \( n = 20 \) case, the CPU times needed to evaluate \( z(Fxy) \) and \( z(Fxy) \) are two (sometimes three) orders of magnitude smaller than the counterparts for \( z(FRLT^+) \). Overall, as far as CPU times are concerned, formulation \( FRLT^+ \) is not competitive with the others.
The Lagrangian relaxation scheme proposed here is quite effective. Note that bounds $z_{LD}$ reported in the tables are very close to $z(F_{STAR})$, the true theoretical bound attainable by our Lagrangian relaxation scheme. In addition to that, SM works very fast. Take the $n = 50$ case as a baseline. The bounds $z_{LD}$ provided by our SM implementation are very close to, sometimes stronger than, $z(F_{xy})$ and $z(F_{yxy})$. However, CPU times needed by SM to evaluate them are, on the average, two orders of magnitude smaller than the CPU times needed by the LP cutting plane algorithms responsible for computing $z(F_{xy})$ and $z(F_{yxy})$. Furthermore, this Lagrangian relaxation scheme is also two orders of magnitude faster than the Lagrangian relaxation algorithm in [18] that evaluates $z(F_{RLT})$.

Table 1: AQMSTP lower bounds - $n = 20$.

<table>
<thead>
<tr>
<th>Inst</th>
<th>$z(F_{RLT})$</th>
<th>$F_{STAR}$ [17]</th>
<th>$F_{xy}$</th>
<th>$F_{yxy}$</th>
<th>$F_{RLT}$</th>
<th>Lag. Rel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>239.2 3</td>
<td>354 354 14</td>
<td>354 0</td>
<td>354 0</td>
<td>355.2 219</td>
<td>354 0</td>
</tr>
<tr>
<td>2</td>
<td>207.6 3</td>
<td>342.9 342.8 12</td>
<td>342.9 0</td>
<td>342.9 0</td>
<td>344.8 825</td>
<td>342.2 0</td>
</tr>
<tr>
<td>3</td>
<td>237.4 3</td>
<td>378.8 378.8 12</td>
<td>378.8 0</td>
<td>380.8 0</td>
<td>380.9 55</td>
<td>378.8 0</td>
</tr>
<tr>
<td>4</td>
<td>216.6 3</td>
<td>323 322.8 15</td>
<td>323 1</td>
<td>323.7 1</td>
<td>323.8 280</td>
<td>322.6 0</td>
</tr>
<tr>
<td>5</td>
<td>170.7 3</td>
<td>332 332 12</td>
<td>332 0</td>
<td>332 0</td>
<td>332 78.5</td>
<td>332 0</td>
</tr>
<tr>
<td>6</td>
<td>211 3</td>
<td>357.7 357.7 14</td>
<td>357.7 1</td>
<td>358.7 0</td>
<td>359.3 221</td>
<td>357.7 0</td>
</tr>
<tr>
<td>7</td>
<td>258.8 3</td>
<td>384.1 384.1 13</td>
<td>384.1 0</td>
<td>384.1 0</td>
<td>385.5 104</td>
<td>384.0 0</td>
</tr>
<tr>
<td>8</td>
<td>193.5 3</td>
<td>315.6 315.6 12</td>
<td>315.6 0</td>
<td>317.0 0</td>
<td>317.9 466</td>
<td>315.6 0</td>
</tr>
<tr>
<td>9</td>
<td>165.4 3</td>
<td>304 304 12</td>
<td>304 1</td>
<td>310.8 1</td>
<td>313.1 288</td>
<td>303.8 0</td>
</tr>
<tr>
<td>10</td>
<td>237.5 3</td>
<td>390.2 390.2 18</td>
<td>390.0 1</td>
<td>393 0</td>
<td>393.8 1274</td>
<td>390.2 0</td>
</tr>
<tr>
<td>avg</td>
<td>213.8 3</td>
<td>348.2 348.2 13</td>
<td>348.2 0</td>
<td>349.7 0</td>
<td>350.6 381</td>
<td>348.1 0</td>
</tr>
</tbody>
</table>

Table 2: AQMSTP lower bounds - $n = 50$.

<table>
<thead>
<tr>
<th>Inst</th>
<th>$z(F_{RLT})$</th>
<th>$F_{STAR}$ [17]</th>
<th>$F_{xy}$</th>
<th>$F_{yxy}$</th>
<th>$z_{LD}$</th>
<th>Lag. Rel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>203.4 187</td>
<td>513.3 513.3 3684</td>
<td>513 187</td>
<td>514.9 122</td>
<td>513 2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>233.2 189</td>
<td>516.8 516.7 4708</td>
<td>516.8 192</td>
<td>516.8 126</td>
<td>516.4 2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>262.8 187</td>
<td>550.1 550.1 5026</td>
<td>549.6 157</td>
<td>550.5 97</td>
<td>549.7 2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>249.9 187</td>
<td>544.3 544.3 4121</td>
<td>544.2 173</td>
<td>544.9 252</td>
<td>544.1 2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>244.9 186</td>
<td>525.8 525.8 3703</td>
<td>525.5 181</td>
<td>527.4 141</td>
<td>525.5 2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>285.9 187</td>
<td>581 581 4429</td>
<td>581 126</td>
<td>581.6 87</td>
<td>580.8 2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>236.4 186</td>
<td>533.8 533.7 4173</td>
<td>532.8 177</td>
<td>533.8 106</td>
<td>533.4 2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>209 186</td>
<td>489.6 489.6 3456</td>
<td>489.5 164</td>
<td>492.5 126</td>
<td>489.3 2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>214.1 185</td>
<td>485.7 485.7 3502</td>
<td>485.7 168</td>
<td>486.9 105</td>
<td>485.3 2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>200.3 188</td>
<td>473.3 473.3 3387</td>
<td>473.3 124</td>
<td>473.6 79</td>
<td>473 2</td>
<td></td>
</tr>
<tr>
<td>avg</td>
<td>234 187</td>
<td>521.4 521.3 4019</td>
<td>518.6 164</td>
<td>519.8 110</td>
<td>521.1 2</td>
<td></td>
</tr>
</tbody>
</table>

23
6.2 A comparison of exact solution approaches

Computational results for BC\(^{(17)}\), BC, BC\(^+\) and BC\(^{RLT+}\) are reported in Tables 3 and 4, respectively for \(n = 20\) and \(n = 50\). For each algorithm, we report the number of nodes investigated in the corresponding enumeration search tree and the CPU time (\(t\)) needed to solve each instance. At the bottom of the tables, we provide average values, considering only the instances solved to optimality by all methods. Results for the \(z(F^{RLT})\) based Lagrangian Relaxation branch-and-bound algorithm of [18] and for BCP\(^{(17)}\) were left out of the comparisons since they were dominated by BC\(^{(17)}\), in terms of CPU times. Results for BC\(^{RLT+}\) are also not presented for the \(n = 50\) case, since it was not able of solving instances with more than 20 vertices within the imposed time limit.

<table>
<thead>
<tr>
<th>Instance</th>
<th>BC(^{(17)}) nodes</th>
<th>BC(^+) nodes</th>
<th>BC(^{RLT+}) nodes</th>
<th>BB nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>399</td>
</tr>
<tr>
<td>2</td>
<td>14 15</td>
<td>12 2</td>
<td>15 2</td>
<td>9 1174</td>
</tr>
<tr>
<td>3</td>
<td>9 14</td>
<td>5 2</td>
<td>4 1</td>
<td>7 272</td>
</tr>
<tr>
<td>4</td>
<td>13 18</td>
<td>23 4</td>
<td>7 2</td>
<td>17 5782</td>
</tr>
<tr>
<td>5</td>
<td>0 13</td>
<td>0 0</td>
<td>0 0</td>
<td>0 78</td>
</tr>
<tr>
<td>6</td>
<td>0 15</td>
<td>12 3</td>
<td>2 1</td>
<td>2 329</td>
</tr>
<tr>
<td>7</td>
<td>11 15</td>
<td>5 1</td>
<td>4 0</td>
<td>13 480</td>
</tr>
<tr>
<td>8</td>
<td>11 15</td>
<td>6 2</td>
<td>6 2</td>
<td>7 895</td>
</tr>
<tr>
<td>9</td>
<td>40 16</td>
<td>21 3</td>
<td>13 2</td>
<td>13 5380</td>
</tr>
<tr>
<td>10</td>
<td>30 20</td>
<td>53 6</td>
<td>32 4</td>
<td>28 31196</td>
</tr>
<tr>
<td><strong>avg</strong></td>
<td>12.8 16</td>
<td>13.7 2</td>
<td>8.3 1</td>
<td>9.6 4599</td>
</tr>
<tr>
<td><strong>nodes</strong></td>
<td></td>
<td></td>
<td><strong>nodes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>t</strong></td>
<td></td>
<td></td>
<td><strong>t</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Comparison of exact solution approaches - \(n = 20\).

All \(n = 20\) instances were solved to proven optimality by the five methods being compared. For that size, the fastest algorithm was the Lagrangian based BB, followed by BC\(^+\), BC, BC\(^{(17)}\), and finally, by BC\(^{RLT+}\). As a result of the large number of variables explicitly considered by BC\(^{RLT+}\), it lagged behind all other methods, despite the fact that it is based on the strongest formulation considered here, \(F^{RLT+}\).

The lack of computer memory was the reason why BC and BC\(^+\) left one \(n = 50\) instance unsolved. In contrast, BB and BC\(^{(17)}\) solved all 10 instances to proven optimality. Considering the 9 instances solved by all of them, a clear advantage in terms of CPU times exists for BC and BC\(^+\) over BC\(^{(17)}\).

However, by far, the fastest algorithm was the Lagrangian based BB method. On the average, for the \(n = 50\) set of instances, BB was one order of magnitude faster than BC and BC\(^+\) and, in some cases, it was nearly two orders of magnitude faster than BC\(^{(17)}\). Nevertheless, BB investigates substantially more nodes than BC\(^{(17)}\), which also relies on the star reformulation. One likely explanation is that few SM iterations are carried out at non-root BB nodes and, in general, SM does not provide the exact Lagrangian Dual bound. The im-
Table 4: Comparison of exact solution approaches - \( n = 50 \). An entry “-” indicates that the instance was not solved within the time limit of 10 CPU hours.

Aiming to clearly understand the reasons why BC, BC\( ^+ \) and BB significantly improved on the results given by BC\( ^{[17]} \), we provide additional computational results for the \( n = 50 \) instances, in Table 5. Besides the total CPU time \( t \), the table presents one additional entry for each algorithm. For BC\( ^{[17]} \), it presents the total time taken to separate projection cuts (26) in column \( t_{\text{sep}} \). For algorithms \( BC \) and \( BC^+ \), the table provides, under columns \( t_{\text{sep}} \), the total time taken to separate clique cuts (36). Finally, for BB, the table presents \( t_{\text{sub}} \), the total CPU time needed to solve the Lagrangian subproblems \( \{ L^*_v(\gamma) : v \in V \} \). These times were measured over the entire enumeration search trees.

Results in Table 5 show that the separation of inequalities (36) accounts for about 1% of the total CPU time taken by \( BC \) and \( BC^+ \). On the contrary, BC\( ^{[17]} \) spent, on the average, about 91% of its CPU time to separate projection cuts. Note however that the difference \( t - t_{\text{sep}} \) is always smaller for BC\( ^{[17]} \). That applies because BC and \( BC^+ \) rely on the extended variable space \((x,y)\), while the BC\( ^{[17]} \) uses only variables \( x \). Thus LP re-optimization is more time consuming for \( BC \) and \( BC^+ \). Except for one \( n = 50 \) instance, \( BC \) and \( BC^+ \) dominated BC\( ^{[17]} \). Indeed, the replacement of (26) by (35) and (36) left the lower bounds nearly unchanged, with significant CPU time savings in the separation of valid inequalities. Overall, these savings paid off the increase in the LP re-optimization time.

Table 5 also shows that the time needed by BB to solve \( \{ L^*_v(\gamma) : v \in V \} \) is always much larger than the time taken by \( BC \) and \( BC^+ \) to separate (36).
The reason is that BB calls \texttt{REC_ENUM} many more times: on the average, for the \( n = 50 \) instances, \( B^+ \) and BB respectively called \texttt{REC_ENUM} \( 6.52 \times 10^4 \) and \( 2.22 \times 10^7 \) times. Each call to \texttt{REC_ENUM} made by \( B^+ \) and BB thus takes, on the average, \( 4 \times 10^{-4} \) and \( 6 \times 10^{-6} \) seconds respectively. The reason for the huge differences between these averages CPU times has to do with the different cost structures involved for the UQBP instances being solved within BB and \( B^+ \).

It seems that the cost structure under the Lagrangian Relaxation scheme helps in speeding up each \texttt{REC_ENUM} call. BB calls \texttt{REC_ENUM} more frequently than \( B^+ \) because the number of separation rounds in \( B^+ \) is not fixed in advance, while for BB, a fixed number of SM iterations is typically carried out at each node of the enumeration tree. In addition, BB typically explores much more nodes than \( B^+ \).

<table>
<thead>
<tr>
<th>Instance</th>
<th>( B^+ ) ( F_{\text{STAR}} )</th>
<th>( B ) ( F_{xy} )</th>
<th>( B^+ ) ( F_{xy} )</th>
<th>( BB ) ( F_{\text{STAR}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( t_{\text{sep}} )</td>
<td>( t )</td>
<td>( t_{\text{sep}} )</td>
<td>( t )</td>
</tr>
<tr>
<td>1</td>
<td>7635</td>
<td>8313.6</td>
<td>3.8</td>
<td>2044</td>
</tr>
<tr>
<td>2</td>
<td>5300.9</td>
<td>5815.9</td>
<td>3.3</td>
<td>1149.4</td>
</tr>
<tr>
<td>3</td>
<td>12574.1</td>
<td>13445.1</td>
<td>6.9</td>
<td>3508.2</td>
</tr>
<tr>
<td>4</td>
<td>14398.6</td>
<td>15503.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>5429.1</td>
<td>5993.2</td>
<td>4.1</td>
<td>2067.3</td>
</tr>
<tr>
<td>6</td>
<td>6381.5</td>
<td>6983.2</td>
<td>3.8</td>
<td>1700.5</td>
</tr>
<tr>
<td>7</td>
<td>5032.6</td>
<td>5594.1</td>
<td>4</td>
<td>1614.1</td>
</tr>
<tr>
<td>8</td>
<td>5332.5</td>
<td>5974.9</td>
<td>6.5</td>
<td>3458.9</td>
</tr>
<tr>
<td>9</td>
<td>3474.9</td>
<td>4025.5</td>
<td>2</td>
<td>566.9</td>
</tr>
<tr>
<td>10</td>
<td>6860.7</td>
<td>7579.6</td>
<td>3.7</td>
<td>1515.2</td>
</tr>
<tr>
<td>avg</td>
<td>7242</td>
<td>7922.9</td>
<td>4.2</td>
<td>1958.3</td>
</tr>
</tbody>
</table>

Table 5: Computational time profile for the exact algorithms.

### 6.3 Computational experiments with new AQMSTP instances

In order to further evaluate the Lagrangian relaxation based BB algorithm, we generated two new sets of test instances. We first present results for the first new set, that was generated as indicated in [17], for larger values of \( n \in \{60, 70, 80\} \).

For each value of \( n \), 10 additional test instances were generated. As in [17], diagonal and off-diagonal costs for these instances were randomly chosen from integers in the intervals \([0, 100]\) and \([0, 20]\), respectively. Computational results for these new instances are presented in Tables 6-8. Since they are out of reach for the other algorithms discussed here, the table only presents results for BB. The first two columns respectively provide the initial AQMSTP upper bound (\(ub\)), and the CPU time taken by the multi-start heuristic to find the corresponding feasible solution. The next two columns respectively indicate the bound \( z_{LD} \) and the CPU time SM took to evaluate that bound. The last four columns provide results for the enumeration search tree. The first two of them
are the best upper (ubb) and lower (blb) bounds found during the search. The subsequent two columns give the number of nodes investigated by BB, nodes, followed by the CPU time needed to solve each instance, t.

The Lagrangian based BB algorithm solved all problems with up to 70 vertices within the specified time limit. Average CPU times BB needed to solve these \( n = 70 \) instances are about one half of the average CPU times needed by BC\textsuperscript{17} to solve similarly generated \( n = 50 \) instances. BB also solved 8 out of 10 instances with 80 vertices within the time limit. For the two remaining ones, BB could not reduce the root node duality gap, even after exploring over 250000 nodes.

Table 6: Lagrangian Relaxation based BB algorithm: results for instances with \( n = 60 \), generated by randomly sampling \( q_{ee} \in \{0,1,\ldots,100\} \) and \( q_{ef} \in \{0,1,\ldots,20\}, e \neq f \), as described in [17].

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial UB</th>
<th>Root Node</th>
<th>BB Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( ub ) t</td>
<td>( z_{LD} ) t</td>
<td>( ub ) ( blb ) nodes t</td>
</tr>
<tr>
<td>1</td>
<td>587 5</td>
<td>464.7 3</td>
<td>491 491 6469 457</td>
</tr>
<tr>
<td>2</td>
<td>635 5</td>
<td>527.7 3</td>
<td>559 559 18119 1339</td>
</tr>
<tr>
<td>3</td>
<td>599 5</td>
<td>490 3</td>
<td>515 515 5349 398</td>
</tr>
<tr>
<td>4</td>
<td>566 6</td>
<td>453.1 3</td>
<td>478 478 3389 200</td>
</tr>
<tr>
<td>5</td>
<td>610 4</td>
<td>522.7 3</td>
<td>548 548 6751 483</td>
</tr>
<tr>
<td>6</td>
<td>596 5</td>
<td>494.5 3</td>
<td>514 514 4637 351</td>
</tr>
<tr>
<td>7</td>
<td>690 5</td>
<td>573.6 3</td>
<td>613 613 49607 3803</td>
</tr>
<tr>
<td>8</td>
<td>592 5</td>
<td>487.7 3</td>
<td>510 510 5163 380</td>
</tr>
<tr>
<td>9</td>
<td>677 6</td>
<td>525.3 3</td>
<td>554 554 12123 910</td>
</tr>
<tr>
<td>10</td>
<td>647 5</td>
<td>518.3 3</td>
<td>544 544 5233 392</td>
</tr>
<tr>
<td>avg</td>
<td>619.9 5</td>
<td>505.7 3</td>
<td>532.6 532.6 11684 877</td>
</tr>
</tbody>
</table>

Table 7: Lagrangian Relaxation based BB algorithm: results for instances with \( n = 70 \), generated by randomly sampling \( q_{ee} \in \{0,1,\ldots,100\} \) and \( q_{ef} \in \{0,1,\ldots,20\}, e \neq f \), as described in [17].

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial UB</th>
<th>Root Node</th>
<th>BB Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( ub ) t</td>
<td>( z_{LD} ) t</td>
<td>( ub ) ( blb ) nodes t</td>
</tr>
<tr>
<td>1</td>
<td>727 11</td>
<td>589.5 6</td>
<td>617 617 9627 1132</td>
</tr>
<tr>
<td>2</td>
<td>673 9</td>
<td>516.1 5</td>
<td>541 541 5171 557</td>
</tr>
<tr>
<td>3</td>
<td>710 11</td>
<td>546.1 5</td>
<td>572 572 15437 1706</td>
</tr>
<tr>
<td>4</td>
<td>668 9</td>
<td>497.8 5</td>
<td>534 534 32197 3411</td>
</tr>
<tr>
<td>5</td>
<td>674 11</td>
<td>520.9 5</td>
<td>560 560 38961 4392</td>
</tr>
<tr>
<td>6</td>
<td>639 11</td>
<td>527.1 5</td>
<td>556 556 25373 2652</td>
</tr>
<tr>
<td>7</td>
<td>747 10</td>
<td>566.1 5</td>
<td>606 606 113819 12266</td>
</tr>
<tr>
<td>8</td>
<td>788 10</td>
<td>649.9 6</td>
<td>685 685 76647 9114</td>
</tr>
<tr>
<td>9</td>
<td>674 11</td>
<td>525.1 5</td>
<td>554 554 15391 1610</td>
</tr>
<tr>
<td>10</td>
<td>727 9</td>
<td>545.9 5</td>
<td>580 580 28191 3102</td>
</tr>
<tr>
<td>avg</td>
<td>702.7 10</td>
<td>548.4 5</td>
<td>580.5 580.5 36081.4 3994</td>
</tr>
</tbody>
</table>
Table 8: Lagrangian Relaxation based BB algorithm: results for instances with \( n = 80 \), generated by randomly sampling \( q_{ee} \in \{0, 1, \ldots, 100\} \) and \( q_{ef} \in \{0, 1, \ldots, 20\}, e \neq f \), as described in [17].

All instances introduced in [17] and [2] have diagonal and off-diagonal costs randomly chosen from \( \{0, 1, \ldots, 100\} \) and \( \{0, 1, \ldots, 20\} \), respectively. We generated a second new set of instances, by also choosing the off-diagonal costs from \( \{0, 1, \ldots, 100\} \), with uniform probability. Again, ten instances were generated for each size of \( n \in \{30, 40, 50\} \). The two best algorithms introduced here, \( BC^+ \) and BB, were tested with these new instances. \( BC^+ \) was largely outperformed by BB. BB solved all instances solved by \( BC^+ \) taking, on the average, 30% less CPU time. BB also solved some \( n = 50 \) instances, none of them being solved by \( BC^+ \). Therefore, in Tables 9-11, we only present results for BB. Instances with more than \( n = 50 \) vertices were not tested, since most of these new \( n = 50 \) instances were left unsolved when the time limit was hit.

Results in these tables suggest that these instances are harder to solve than the previous ones. On the average, BB takes longer to solve the new \( n = 30 \) instances than the previously generated \( n = 50 \) instances. A summary of BB results for the new instances is the following. For \( n = 30 \), all ten instances were solved within the time limit; for \( n = 40 \), 9 out of the 10 instances were solved, while for \( n = 50 \), only 4 were solved.

7 Conclusions

In this paper, we investigated the star reformulation [17] for the Adjacent Only Minimum Spanning Tree Problem. We investigated the projection of the variables associated to the stars into the usual \((x, y)\) variable space and characterized two sets of valid inequalities. One of these two families, clique cuts, was proven to define facets of the polytope of feasible solutions. We also lifted the other family into facet defining inequalities for AQMSTP.

On the algorithmic side, we suggested three branch-and-cut algorithms that, in different ways, explore the inequalities characterized by projection. Two of
Table 9: Lagrangian Relaxation based BB algorithm: results for instances with $n = 30$, generated by randomly sampling $q_{ee} \in \{0, 1, \ldots, 100\}$ and $q_{ef} \in \{0, 1, \ldots, 100\}$, $e \neq f$.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial UB $ub$</th>
<th>Root Node $z_{LD} t$</th>
<th>BB Tree $bub bb$ $nodes t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1170</td>
<td>858.7 1</td>
<td>929 929 2239 45</td>
</tr>
<tr>
<td>2</td>
<td>950</td>
<td>749.5 1</td>
<td>828 828 24217 440</td>
</tr>
<tr>
<td>3</td>
<td>1149</td>
<td>770.2 1</td>
<td>848 848 2859 58</td>
</tr>
<tr>
<td>4</td>
<td>1218</td>
<td>742.3 1</td>
<td>826 826 11919 231</td>
</tr>
<tr>
<td>5</td>
<td>1146</td>
<td>791.4 1</td>
<td>855 855 5131 96</td>
</tr>
<tr>
<td>6</td>
<td>1201</td>
<td>799.2 1</td>
<td>887 887 33671 621</td>
</tr>
<tr>
<td>7</td>
<td>1168</td>
<td>815.6 1</td>
<td>894 894 8853 175</td>
</tr>
<tr>
<td>8</td>
<td>1160</td>
<td>857.3 1</td>
<td>930 930 24145 435</td>
</tr>
<tr>
<td>9</td>
<td>1211</td>
<td>770.5 1</td>
<td>818 818 1089 24</td>
</tr>
<tr>
<td>10</td>
<td>1097</td>
<td>806.4 1</td>
<td>893 893 22485 427</td>
</tr>
</tbody>
</table>

$avg$ 1147 0 795.5 1 870.8 870.8 13660.8 255

Table 10: Lagrangian Relaxation based BB algorithm: results for instances with $n = 40$, generated by randomly sampling $q_{ee} \in \{0, 1, \ldots, 100\}$ and $q_{ef} \in \{0, 1, \ldots, 100\}$, $e \neq f$.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial UB $ub$</th>
<th>Root Node $z_{LD} t$</th>
<th>BB Tree $bub bb$ $nodes t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1387</td>
<td>918.1 3</td>
<td>1001 1001 18371 825</td>
</tr>
<tr>
<td>2</td>
<td>1266</td>
<td>916.4 3</td>
<td>957 957 11659 524</td>
</tr>
<tr>
<td>3</td>
<td>1374</td>
<td>935.6 3</td>
<td>1012 1012 21943 973</td>
</tr>
<tr>
<td>4</td>
<td>1316</td>
<td>946.8 2</td>
<td>1019 1019 19925 846</td>
</tr>
<tr>
<td>5</td>
<td>1379</td>
<td>886.2 3</td>
<td>990 990 268907 11511</td>
</tr>
<tr>
<td>6</td>
<td>1449</td>
<td>881.1 2</td>
<td>965 965 18535 797</td>
</tr>
<tr>
<td>7</td>
<td>1229</td>
<td>881 3</td>
<td>920 920 5175 218</td>
</tr>
<tr>
<td>8</td>
<td>1408</td>
<td>914.7 3</td>
<td>1000 1000 77329 3302</td>
</tr>
<tr>
<td>9</td>
<td>1313</td>
<td>849.2 2</td>
<td>895 895 5871 219</td>
</tr>
<tr>
<td>10</td>
<td>1278</td>
<td>937.6 3</td>
<td>1053 939.3 870007 36000*</td>
</tr>
</tbody>
</table>

$avg$ 1339.9 0 906.7 2 981.2 969.8 131772.2 2135
these new branch-and-cut algorithms provided very good computational results, almost always dominating the branch-and-cut algorithm in Pereira et al. [17].

A fourth branch-and-bound algorithm was introduced here. It is based on a Lagrangian relaxation scheme for the star reformulation. It managed to solve all AQMSTP instances in the literature, in about one tenth of the time needed by its competitors. Considering instances with cost structure similar to those previously found in the literature, the algorithm increased from 50 to 70 vertices the size of the instances that are now systematically solved to proven optimality. Harder test instances, with larger off-diagonal costs, were also tested. For them, the algorithm also outperforms its competitors and consistently solves instances with 30 vertices.

Acknowledgments This research was partially conducted when Alexandre Salles da Cunha was visiting LIMOS - Laboratoire d’Informatique, de Modélisation et d’Optimisation des Systèmes, in Clermont-Ferrand, France. Alexandre Salles da Cunha was partially funded by FAPEMIG grant CEX - PPM-00187-15 and CNPq grants 305423/2012-6, 471464/2013-9 and 200493/2014-0.

References


---

<table>
<thead>
<tr>
<th>Instance</th>
<th>Initial UB</th>
<th>Root Node</th>
<th>BB Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ub</td>
<td>t</td>
<td>$z_{LD}$</td>
</tr>
<tr>
<td>1</td>
<td>1661 1</td>
<td>988 4</td>
<td>1128 988</td>
</tr>
<tr>
<td>2</td>
<td>1527 1</td>
<td>950.8 4</td>
<td>1033 1033</td>
</tr>
<tr>
<td>3</td>
<td>1718 1</td>
<td>1055.2 4</td>
<td>1150 1077.8</td>
</tr>
<tr>
<td>4</td>
<td>1812 1</td>
<td>1056.8 4</td>
<td>1218 1056.8</td>
</tr>
<tr>
<td>5</td>
<td>1582 1</td>
<td>1026.7 5</td>
<td>1172 1026.7</td>
</tr>
<tr>
<td>6</td>
<td>1676 1</td>
<td>1042.9 5</td>
<td>1169 1169</td>
</tr>
<tr>
<td>7</td>
<td>1715 1</td>
<td>1044.3 5</td>
<td>1136 1136</td>
</tr>
<tr>
<td>8</td>
<td>1673 1</td>
<td>1006.4 5</td>
<td>1141 1014.1</td>
</tr>
<tr>
<td>9</td>
<td>1491 1</td>
<td>1010.1 5</td>
<td>1084 1084</td>
</tr>
<tr>
<td>10</td>
<td>1588 1</td>
<td>986.9 4</td>
<td>1118 1018.1</td>
</tr>
<tr>
<td>avg</td>
<td>1644.3 1</td>
<td>1016.8 4</td>
<td>1134.9 1060.3</td>
</tr>
</tbody>
</table>

Table 11: Lagrangian Relaxation based BB algorithm: results for instances with $n = 50$, generated by randomly sampling $q_{ee} \in \{0, 1, \ldots, 100\}$ and $q_{ef} \in \{0, 1, \ldots, 100\}$, $e \neq f$. 


