Computers & Operations Research 63 (2015) 149–160

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Lower bounds and exact algorithms for the quadratic minimum spanning tree problem

Dilson Lucas Pereira [a,1], Michel Gendreau [b], Alexandre Salles da Cunha [c,*,2]

[a] CAPES Foundation, Ministry of Education of Brazil, Brasília - DF 70.040-020, Brazil
[b] École Polytechnique de Montréal, Montréal - Canada
[c] Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte - MG, Brazil

ARTICLE INFO

Available online 14 May 2015

Keywords:
Quadratic 0-1 programming
Lagrangian relaxation
Spanning trees

ABSTRACT

We address the quadratic minimum spanning tree problem (QMSTP), the problem of finding a spanning tree of a connected and undirected graph such that a quadratic cost function is minimized. We first propose an integer programming formulation based on the reformulation–linearization technique (RLT). We then use the idea of partitioning spanning trees into forests of a given fixed size and obtain a QMSTP reformulation that generalizes the RLT model. The reformulation is such that the larger the size of the forests, the stronger lower bounds provided. Thus, a hierarchy of formulations is obtained. At the lowest hierarchy level, one has precisely the RLT formulation, which is already stronger than previous formulations in the literature. The highest hierarchy level provides the convex hull of integer feasible solutions for the problem. The formulations introduced here are not compact, so the direct evaluation of their linear programming relaxation bounds is not practical. To overcome that, we introduce two lower bounding procedures based on Lagrangian relaxation. These procedures are embedded into two parallel branch-and-bound algorithms. As a result of our study, several instances in the literature were solved to optimality for the first time.

1. Introduction

We are given an undirected graph $G=(V,E)$ and a matrix $Q=(q_{ef})_{e,f\in E}$ of *interaction costs* between the edges of G. We assume that G is a connected graph, with $n=|V|$ vertices and $m=|E|$ edges. The quadratic minimum spanning tree problem (QMSTP) consists of finding a spanning tree of G whose incidence vector $\mathbf{x}\in\mathbb{B}^m$ minimizes the function

$$\sum_{e,f\in E} q_{ef}x_e x_f.$$

QMSTP was proven to be NP-Hard by Assad and Xu [1]. However, if the objective function is linear (i.e., if Q is diagonal), QMSTP reduces to the minimum spanning tree problem (MSTP), for which several polynomial time algorithms are known [2,3]. Another interesting particular case of QMSTP is the adjacent-only QMSTP (AQMSTP), where $q_{ef}=0$ holds for all edges e and j that do not share endpoints. AQMSTP is also NP-Hard [1].

1.1. Literature review

In this section, we present a review of exact solution approaches for QMSTP. For heuristic algorithms, we suggest references [1,4–9].

A common procedure for computing lower bounds for constrained quadratic 0-1 problems is that of Gilmore [10] and Lawler [11]. The procedure was introduced for the quadratic assignment problem (QAP) and later adapted for other problems, e.g., the quadratic 0–1 knapsack problem [12]. Quite often, the Gilmore–Lawler procedure is also used in the resolution of subproblems in Lagrangian relaxation schemes.

The QMSTP was first addressed by Assad and Xu [1]. The authors introduced a lower bounding procedure that can be thought as a dual ascent algorithm for obtaining near optimal multipliers in a Lagrangian relaxation scheme. The structure of that Lagrangian relaxation was chosen so that the Gilmore–Lawler procedure could be used to solve the Lagrangian subproblems. Assad and Xu [1] implemented a branch-and-bound (BB) algorithm based on the Lagrangian lower bounds and managed to solve instances defined over complete graphs with up to 12 vertices.

Öncan and Punnen [6] also introduced a procedure based on Lagrangian relaxation. Their strategy relies on the QMSTP formulation introduced in [1], strengthened with new valid inequalities.

* Corresponding author.
E-mail addresses: dilson@dcc.ufmg.br (D.L. Pereira), michel.gendreau@cirrelt.ca (M. Gendreau), acunha@dcc.ufmg.br (A.S. da Cunha).
[1] Dilson Lucas Pereira was partially funded by CAPES, BEX 2418/11-8.
[2] Alexandre Salles da Cunha is partially funded by CNPq Grant nos. 305423/2012-6, 471464/2013-9 and FAPEMIG CEX PPM 00164-13.

http://dx.doi.org/10.1016/j.cor.2015.04.020
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

The structure of their relaxation is similar to that in [1]; they relax the same set of constraints relaxed in [1] plus the additional reinforcing inequalities. Lagrangian relaxation subproblems are thus solved by the Gilmore–Lawler procedure and multipliers are adjusted by subgradient optimization. However, a BB algorithm was not implemented. Inconsistencies on their computational results were reported in [9,13].

A modified version of the exact algorithm in [1] was introduced by Cordone and Passeri [9]. In particular, the new algorithm is based on a relaxed version of the Lagrangian subproblem in [1] that sacrifices lower bounds quality but, overall, improves CPU times. The exact algorithm in [1] 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.

Assad and Xu [1] also addressed the AQMSTP particular case. Their algorithm is similar to the one for the general case, but it exploits the special QMSTP cost structure in order to speed-up the resolution of the Lagrangian subproblems. Instances defined over complete graphs with up to 15 vertices were solved.

A bi-objective version of AQMSTP was investigated by Maia et al. [14]. The linear ($\sum_{e \in E} q_{ee} x_e$) and the quadratic ($\sum_{e,f \in E, e \neq f} q_{ef} x_e x_f$) parts of the AQMSTP cost function were considered as two separate objective functions. They proposed approaches for generating solutions in the Pareto front. One of those approaches is a local search heuristic. The other is an adaptation of the BB algorithm in [1]. The adapted BB algorithm managed to generate the Pareto front for instances with up to 20 vertices.

Pereira et al. [15] proposed a stronger reformulation for AQMSTP that is naturally linear. To achieve that, the formulation employs an extended variable space, given by $\mathbf{x}$ and also by binary variables assigned to the stars of G. Pereira et al. [15] also investigated the formulation obtained after projecting the extended variable space on the $\mathbf{x}$ space. Two BB algorithms were implemented and tested: a Branch-and-cut-and-price algorithm for the extended reformulation and a Branch-and-cut algorithm based on its projection. Instances defined over complete graphs with up to 50 vertices were solved.

A common approach to solve quadratic 0-1 programs consists of linearizing the non-linear terms in the objective function in order to obtain a (mixed) integer linear program. In general, the process is the following. Assuming that a set $\mathbf{x} = (x_e)_{e \in M}$ of variables is used to formulate the problem, the first step in the reformulation is the introduction of additional variables $\mathbf{y} = (y_{ef})_{e,f \in M}$. The quadratic terms $x_e x_f$ are then replaced by the corresponding variables y_{ef}. Finally, linear constraints enforcing the condition $y_{ef} = x_e x_f$ are added to the formulation.

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 and $\mathbf{y}$ satisfies $y_{ef} = x_e x_f$ for all $e, f \in M$. In the QMSTP case, for instance, feasible values for $\mathbf{x}$ are the incidence vectors of spanning trees of G. In the case where $\mathbf{x}$ might be any binary vector, the resulting linearized polytope is known as the boolean quadric polytope (BQP). Valid inequalities for BQP are valid for all constrained quadratic 0–1 problems, QMSTP included. BQP has been widely studied, see Padberg [16], for example. 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 [17]. However, to our knowledge, no computational experiments with those inequalities were ever reported.

1.2. *Outline of the paper and main contributions*

Despite the attention it received in the literature, QMSTP remains very hard to solve to proven optimality, since lower bounds provided by known formulations are often still far away from optimal objective values. We propose stronger QMSTP formulations as well as exact solution approaches based on them. Thanks to new lower bounds that are significantly stronger than previous ones and to the use of highly efficient parallel computing strategies to search the branch-and-bound trees, we now solve instances with up to 50 vertices to proven optimality.

The paper is organized as follows. In Section 2, we introduce a new integer programming (IP) formulation for QMSTP, based on a partial application of the reformulation–linearization technique (RLT) [18,19]. An effective Lagrangian relaxation scheme is developed to evaluate its linear programming (LP) relaxation bounds. It is also shown that the LP lower bounds provided by that formulation dominate previous QMSTP lower bounds in the literature.

In Section 3, we develop a generalization of the formulation presented in Section 2. The generalization is based on the idea of decomposing spanning trees into forests of a fixed size. A hierarchy of formulations of increasing strength results from such an idea. The larger the size of the forests, the stronger the formulations. However, the number of variables and constraints also grow with that parameter. Aiming at developing lower bounding procedures of practical value, we propose strategies to reduce the number of variables in the formulation and study two relaxations for it. Although these relaxations are hard to solve in general, we identify a particular case that can be solved in polynomial time, and develop a Lagrangian relaxation scheme based on it.

The two Lagrangian relaxation schemes we propose are embedded into two BB algorithms in Section 4. Given the difficulty to solve QMSTP in practice, these BB algorithms employ parallel programming techniques. As a result of new features suggested here for load balancing, our parallel algorithms obtain good rates of parallel efficiency (around 80%).

Finally, in Section 5, we present computational experiments conducted on two sets of instances from the literature. Our computational results attest the quality of the new lower bounds introduced here. Moreover, many new optimality certificates are provided with the algorithms we propose.

1.3. *Notation*

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'. We denote by $\delta(V') = \{\{u,v\} \in E : u \in V', v \notin V'\}$ the set of edges with exactly one endpoint in V'. We use $\mathbb{B} = \{0,1\}$ to refer to the set of binary values.

Given a formulation F for an optimization problem, we denote by $Z(F)$ its optimal solution value. If F is an IP formulation, we denote by $Z_{LP}(F)$ the optimal solution value of its LP relaxation.

2. RLT-1 based bounds

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

$$\min \left\{ \sum_{e,f \in E} q_{ef} x_e x_f : \mathbf{x} \in X \cap \mathbb{B}^m \right\},$$

where X denotes the convex hull of the incidence vectors of the spanning trees of G [20], i.e., the set of vectors in $\mathbb{R}^m$ that satisfy

$$\sum_{f \in E} x_f = n - 1, \tag{1}$$

$$\sum_{f \in E(S)} x_f \le |S|-1, \quad S \subset V, \quad |S| \ge 2, \tag{2}$$

$$0 \le x_f \le 1, \quad f \in E. \tag{3}$$

For each edge e, define $X_e = X \cap \{\mathbf{x} \in \mathbb{R}^m : x_e = 1\}$, i.e., X_e is the convex hull of the incidence vectors of the spanning trees of G that contain e (see [21] for details on the spanning tree polytope).

To obtain a 0–1 IP formulation for QMSTP, we apply a scheme based on the reformulation–linearization technique (RLT) (Caprara [22] gives an exposition of this approach for general constrained quadratic 0–1 programs). The scheme consists of two steps:

Reformulation step: Multiply each constraint in (1)–(3) by each variable x_e, obtaining new, non-linear, constraints.

Linearization step: Introduce linearization variables $\mathbf{y} = (y_{ef})_{e,f \in E}$. For each constraint obtained after the multiplication by x_e, replace the product $x_e x_f$ by y_{ef}. Do the same for the objective function, using y_{ef} to replace the product $x_e x_f$. Note that we explicitly distinguish between $y_{ef} = x_e x_f$ and $y_{fe} = x_f x_e$. We also replace the powers x_e^2 by y_{ee} instead of x_e. In doing so, we will be able to exploit a special structure in the resulting formulation. Because of this, we will need to add constraints $y_{ef} = y_{fe}$, for all distinct edges e and f, and $y_{ee} = x_e$, for every edge e.

For each variable x_e, the application of the reformulation–linearization procedure above yields the constraints:

$$\sum_{f \in E} y_{ef} = (n-1)x_e, \tag{4}$$

$$\sum_{f \in E(S)} y_{ef} \le (|S|-1)x_e, \quad S \subset V, \quad |S| \ge 2, \tag{5}$$

$$y_{ee} = x_e, \tag{6}$$

$$0 \le y_{ef} \le x_e, \quad f \in E. \tag{7}$$

Notice that (4)–(7) is simply X_e multiplied by x_e. Thus, we refer to (4)–(7) as $X_e \times x_e$. We also denote by $\mathbf{y}_e = (y_{ef})_{f \in E}$ the row of $\mathbf{y}$ indexed by e. Once the linearization scheme has been applied, the following 0–1 IP formulation is obtained

$$F_{RLT}: \quad \min\left\{ \sum_{e,f \in E} q_{ef} y_{ef} : (\mathbf{x},\mathbf{y}) \in P_{RLT} \cap \mathbb{B}^{m+m^2} \right\},$$

where P_{RLT} refers to the polyhedral region defined by

$$\mathbf{x} \in X, \tag{8}$$

$$\mathbf{y}_e \in X_e \times x_e, \quad e \in E, \tag{9}$$

$$y_{ef} = y_{fe}, \quad e < f \in E. \tag{10}$$

Constraints (8) imply that $\mathbf{x}$ must be the incidence vector of a spanning tree. Constraints (9) guarantee that, for each edge e satisfying $x_e = 1$, $\mathbf{y}_e$ must be the incidence vector of a spanning tree containing e. However, for each edge e such that $x_e = 0$, constraints (9) enforce that $\mathbf{y}_e$ must be the null vector. Finally, constraints (10) guarantee that the spanning trees implied by the vectors $\mathbf{y}_e$ and $\mathbf{x}$ must match.

Observe that the tree implied by $\mathbf{y}_e$ determines the edges that interact with e. For this reason, we refer to it as the *interaction tree* for edge e. Formulation F_{RLT} requires that all interaction trees are identical. These are hard constraints and are going to be relaxed in our solution strategy.

The reformulation process just discussed consists of a partial application of the first RLT level (RLT-1) [23,19]. The complete RLT scheme would also involve the multiplication of (1)–(3) by $(1-x_e)$, followed by the linearization step. As such, the full application of the first RLT level would provide the following additional valid inequalities for QMSTP:

$$(\mathbf{x}-\mathbf{y}_e)(E(S)) \le (|S|-1)(1-x_e), \quad e \in E, \quad S \subset V, \quad |S| \ge 2, \tag{11}$$

which are obtained from (2). In the full RLT reformulation, the multiplication of constraints (1) and (3) by $(1-x_e)$ leads to redundant inequalities. Therefore, only the inclusion of (2) is needed to obtain the complete RLT-1 reformulation.

It was shown by Lee and Leung [17] that (5), (7), and (11) define facets of BQFP. In that study, it was also shown that the clique and the cut inequalities for BQP [16] also define facets for BQFP. The authors in [17] also provide polynomial time separation algorithms for inequalities (5) and (11), to be used within LP relaxation algorithms. To the best of our knowledge, no polynomial time separation algorithms were provided before, for the remaining inequalities presented in [17].

In the 0–1 quadratic programming literature, another linearization scheme has been used for quite a long time. Such an approach uses constraints (7), (10) and also

$$y_{ef} \ge x_e + x_f - 1, \quad e < f \in E, \tag{12}$$

which is a particular case of the clique inequalities for BQP [16]. Our preliminary computational experiments indicated that neither (11) nor (12) significantly improves on bounds $Z_{LP}(F_{RLT})$. However, the inclusion of these inequalities adds a significant computational overhead for the evaluation of the associated lower bounds. For this reason, our formulation and solution techniques do not consider them.

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

$$\sum_{e \in E} y_{ef} = (n-1)x_f, \quad f \in E, \tag{13}$$

$$\sum_{e \in \delta(v)} y_{ef} \ge x_f, \quad f \in E, v \in V. \tag{14}$$

Assad and Xu [1] introduced the QMSTP formulation

$$F_{AX92}: \quad \min\left\{ \sum_{e,f \in E} q_{ef} y_{ef} : (\mathbf{x},\mathbf{y}) \in P_{AX92} \cap \mathbb{B}^{m+m^2} \right\},$$

where $P_{AX92} = \left\{ (\mathbf{x},\mathbf{y}) \in \mathbb{R}^{m+m^2} : (\mathbf{x},\mathbf{y}) \text{ satisfies (8)–(9) and (13)} \right\}$. Öncan and Punnen [6] added the valid inequalities (14) to P_{AX92} and formulated QMSTP as

$$F_{OP10}: \quad \min\left\{ \sum_{e,f \in E} q_{ef} y_{ef} : (\mathbf{x},\mathbf{y}) \in P_{OP10} \cap \mathbb{B}^{m+m^2} \right\},$$

where $P_{OP10} = P_{AX92} \cap \left\{ (\mathbf{x},\mathbf{y}) \in \mathbb{R}^{m+m^2} : (\mathbf{x},\mathbf{y}) \text{ satisfies (4)} \right\}$. The following results hold true.

Proposition 1. $P_{AX92} \supseteq P_{OP10} \supseteq P_{RLT}$

Proof. Constraints (13) are clearly implied by (4) and (10). To check that constraints (14) are also implied by P_{RLT}, formulate (5) in terms of an edge f and $S = V/\{v\}$, for any vertex v. Then subtract the resulting inequality from (4) (also formulated for f), to obtain

$$\sum_{e \in \delta(v)} y_{fe} \ge x_f,$$

which, together with (10), implies (14).□

Corollary 1. $Z_{LP}(F_{AX92}) \le Z_{LP}(F_{OP10}) \le Z_{LP}(F_{RLT})$.

Due to the large number of variables and constraints in F_{RLT}, computing $Z_{LP}(F_{RLT})$ by directly solving the LP relaxation of F_{RLT}, even with a cutting plane algorithm where (2) and (5) are dynamically separated, is not practical. Therefore, we implement an alternative strategy. We relax and dualize constraints (10) by attaching to them unconstrained Lagrangian multipliers $\theta = (\theta_{ef})_{e<f\in E}$ (assume $\theta_{ef} = -\theta_{fe}$ in case $e > f \in E$), to obtain the problem

$$LRP_{RLT}(\theta): \quad \min\left\{\sum_{e,f\in E} q'_{ef} y_{ef} : (\mathbf{x},\mathbf{y}) \in P^{LRP}_{RLT} \cap \mathbb{B}^{m+m^2}\right\},$$

where polytope P^{LRP}_{RLT} is obtained by relaxing (10) in the definition of P_{RLT}, i.e., P^{LRP}_{RLT} is given by (8) and (9). Lagrangian modified costs are defined as $q'_{ef} = q_{ef} + \theta_{ef}$, for distinct edges e and f, and $q'_{ee} = q_{ee}$, for every edge e. The corresponding Lagrangian dual is

$$LDP_{RLT}: \quad \max\left\{Z(LRP_{RLT}(\theta)) : \theta \in \mathbb{R}^{\frac{m(m-1)}{2}}\right\}.$$

It is not hard to see that, because X and X_e are integral polytopes, P^{LRP}_{RLT} is also an integral polytope. This, in turn, implies that $Z(LDP_{RLT}) = Z_{LP}(F_{RLT})$ holds.

Observe that after relaxing (10), all $\mathbf{y}_e$ become independent of one another. As a consequence, it is possible to develop a specialized procedure for solving $LRP_{RLT}(\theta)$. Let $(\overline{\mathbf{x}},\overline{\mathbf{y}})$ be an optimal solution for $LRP_{RLT}(\theta)$. If $\overline{x}_e = 0$ for an edge e, then $\overline{\mathbf{y}}_e = \mathbf{0}$. On the other hand, if $\overline{x}_e = 1$, $(\overline{\mathbf{x}},\overline{\mathbf{y}})$ must be so that $\overline{\mathbf{y}}_e$ is the incidence vector of a spanning tree that solves

$$p_e = \min\left\{\sum_{f\in E} q'_{ef} y_{ef} : \mathbf{y}_e \in X_e \cap \mathbb{B}^m\right\}. \tag{15}$$

This means that the selection of edge e implies a cost given by p_e. Consequently, one can solve $LRP_{RLT}(\theta)$ by computing the spanning tree $\overline{\mathbf{x}}$ that minimizes:

$$\overline{p} = \min\left\{\sum_{e\in E} p_e x_e : \mathbf{x} \in X \cap \mathbb{B}^m\right\}, \tag{16}$$

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

Algorithm 1. Solution for $LRP_{RLT}(\theta)$.

```
Input: A QMSTP instance given by $G=(V,E)$ and $Q \in \mathbb{R}^{m^2}_{+}$. A set
    of Lagrangian multipliers $\theta \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: A solution $(\overline{\mathbf{x}},\overline{\mathbf{y}})$ for $LRP_{RLT}(\theta)$ .
1. For each edge $e$, solve (15) and denote the solution vector by
   $\tilde{\mathbf{y}}_e$.
2. Solve (16), denote by $\tilde{\mathbf{x}}$ the solution vector.
3. An optimal solution $(\overline{\mathbf{x}},\overline{\mathbf{y}})$ of cost $Z(LRP_{RLT}(\theta)) = \overline{p}$ for
   $LRP_{RLT}(\theta)$ is given by $\overline{\mathbf{x}} = \tilde{\mathbf{x}}$ and $\overline{\mathbf{y}}_e = \tilde{x}_e\tilde{\mathbf{y}}_e$ for all $e \in E$.
```

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

An approximate solution for the Lagrangian dual LDP_{RLT} can be obtained by means of the subgradient method [24]. We denote by LAG_{RLT} the algorithm that uses the subgradient method to solve LDP_{RLT} and uses Algorithm 1 to solve the Lagrangian subproblems. Based on the observations above, for the practical evaluation of $Z_{LP}(F_{RLT})$, it is reasonable to expect LAG_{RLT} to perform better than directly solving the LP relaxation of F_{RLT}.

In Section 4, a BB algorithm based on lower bounding procedure LAG_{RLT} is introduced. Computational results for that algorithm are provided later, in Section 5.

Although formulation F_{RLT} is at least as strong as F_{AX92} and F_{OP10}, duality gaps implied by $Z_{LP}(F_{RLT})$ may still be large. This observation motivates the study of stronger lower bounding approaches for QMSTP.

3. A generalization of the RLT-1 based bounds

Let K be a positive integer that divides $n-1$. Then, any partition of the edge set of any spanning tree T of G results in $(n-1)/K$ acyclic subsets, with K edges each. In other words, T can be decomposed into $(n-1)/K$ forests of K edges each. We refer to such forests as K-forests. A single edge defines a 1-forest. Thus, Algorithm 1 finds the minimum cost that can be implied by all 1-forests in any solution. It then uses these bounds to select the best possible set of $(n-1)$ 1-forests. A generalization of this idea consists of finding the minimum cost that can be implied by all K-forests in any solution, and then using these values to select the best $(n-1)/K$ K-forests. This idea is presented in detail in this section.

We first show that the bounds obtained with this generalization can be computed by applying Lagrangian relaxation to a conveniently defined QMSTP reformulation. Some additional notation is needed.

Denote by $E^K = \{H \subseteq E : H \text{ defines a } K\text{−forest }\}$ the set of all subsets of edges of E that induce K-forests and let $o = |E^K|$. Define $E^K_e = \{H \in E^K : e \in H\}$ as the set of the elements of E^K that contain edge e. Finally, for all $H \in E^K$, let $X_H = X \cap \{\mathbf{x} \in \mathbb{R}^m : x_e = 1, \forall e \in H\}$. It is not hard to see 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 is selected to be part of the solution. To formulate the interaction trees, binary variables $\mathbf{t} = (t_{He})_{H\in E^K, e\in E}$ are used. We denote the row of $\mathbf{t}$ indexed by H by $\mathbf{t}_H = (t_{He})_{e\in E}$. Note that $\mathbf{t}_H$ defines the interaction tree for the K-forest induced by H. QMSTP can be formulated as

$$F^K_{GEN}: \quad \min\left\{\sum_{e,f\in E} q_{ef} y_{ef} : (\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{t}) \in P^K_{GEN} \cap \mathbb{B}^{m+m^2+o+om}\right\},$$

where polytope P^K_{GEN} is given by

$$\mathbf{x} \in X, \tag{17}$$

$$x_e = \sum_{H\in E^K_e} s_H, \quad e \in E, \tag{18}$$

$$\mathbf{t}_H \in X_H \times s_H, \quad H \in E^K, \tag{19}$$

$$\mathbf{y}_e = \sum_{H\in E^K_e} \mathbf{t}_H, \quad e \in E, \tag{20}$$

$$y_{ef} = y_{fe}, \quad e < f \in E. \tag{21}$$

The notation $X_H \times s_H$ is used to represent the set of points in X_H multiplied by s_H, i.e.:

$$\sum_{f\in E} t_{Hf} = (n-1)s_H, \tag{22}$$

$$\sum_{f\in S} t_{Hf} \le (|S|-1)s_H, \quad S \subset V, \quad |S| \ge 2, \tag{23}$$

$$t_{Hf} = s_H, \quad e \in H, \tag{24}$$

$$0 \le t_{Hf} \le s_H, \quad e \in E. \tag{25}$$

Constraints (17) require that $\mathbf{x}$ defines a spanning tree of G. The decomposition of the spanning tree into K-forests is enforced by constraints (18). By (19), if a set H that induces a K-forest is used in the decomposition ($\mathbf{t}_H = 1$), then an interaction tree must be defined for it. Otherwise, $\mathbf{t}_H = \mathbf{0}$. Constraints (20) state that, in case edge e appears in a K-forest that belongs to the solution, then the interaction tree $\mathbf{y}_e$ for e must be equal to the interaction tree for that K-forest, otherwise, $\mathbf{y}_e = \mathbf{0}$. Finally, by (21), 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 fixed. Thus, F^K_{GEN} is only polynomially larger than F_{RLT}. Regarding the strength of F^K_{GEN}, when $K=1$, formulations F_{RLT} and F^K_{GEN} are equivalent. If $K = n-1$, E^K will be the set of all spanning trees of G and, consequently, $Z_{LP}(F^K_{GEN})$ will be the optimal solution value of QMSTP. For general values of K, we have the following result.

Proposition 2. *Let* $\text{Proj}_{\mathbf{xy}}(P^K_{GEN})$ *be the projection of* P^K_{GEN} *onto the vector space of* $(\mathbf{x},\mathbf{y})$. *For two factors* K *and* L *of* $n-1$, $L > K$, *the following holds*

$$\text{Proj}_{\mathbf{xy}}(P^L_{GEN}) \subseteq \text{Proj}_{\mathbf{xy}}(P^K_{GEN}).$$

Proof. Consider a vector $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}}) \in P^L_{GEN}$. We are going to show that there is a vector $(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{s}},\tilde{\mathbf{t}}) \in P^K_{GEN}$ such that $\tilde{\mathbf{x}} = \overline{\mathbf{x}}$ and $\tilde{\mathbf{y}} = \overline{\mathbf{y}}$.

For every $H \in E^L$ define the set $E^K(H) = \{e \in E^K : H \cap I = K\}$, 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 $e \in E^K$ let

$$\tilde{s}_e = \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} \overline{s}_H, \tag{26}$$

and

$$\tilde{\mathbf{t}}_e = \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} \overline{\mathbf{t}}_H. \tag{27}$$

Thus, from (26), for any edge e, we have

$$\overline{x}_e = \sum_{H \in E^L_e} \overline{s}_H = \sum_{H \in E^L_e} \frac{1}{c} \sum_{e \in E^K(H) : e \in I} \overline{s}_H = \sum_{e \in E^K_e} \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} \overline{s}_H = \sum_{e \in E^K_e} \tilde{s}_e = \tilde{x}_e,$$

which shows that (17) and (18) are satisfied in P^K_{GEN}. From (27) we have

$$\overline{\mathbf{y}}_e = \sum_{H \in E^L_e} \overline{\mathbf{t}}_H = \sum_{H \in E^L_e} \frac{1}{c} \sum_{e \in E^K(H) : e \in I} \overline{\mathbf{t}}_H = \sum_{e \in E^K_e} \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} \overline{\mathbf{t}}_H = \sum_{e \in E^K_e} \tilde{\mathbf{t}}_e = \tilde{\mathbf{y}}_e,$$

which shows that (20) and (21) are also satisfied in P^K_{GEN}. Finally, we need to show that (19) is also satisfied. We will only show that, for any edge e, the cardinality constraint $\sum_{e \in E} t_{Ie} = (n-1)s_e$ is satisfied. The satisfaction of the remaining constraints can be proved in a similar fashion. Using both (26) and (27),

$$\sum_{e \in E} \tilde{t}_{Ie} = \sum_{e \in E} \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} \overline{t}_{He} = \frac{1}{c} \sum_{H \in E^L : e \in E^K(H)} (n-1)\overline{s}_H = (n-1)\tilde{s}_e,$$

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 to grant that at least one of such decomposition does exist for any spanning tree. By eliminating redundant possibilities from E^K, we can reduce the number of variables of F^K_{GEN} and eventually improve its LP relaxation. The next result shows how that can be accomplished for $K \le 4$.

Proposition 3. *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 $\{u,v\}$ and $\{v,w\}$ such that u and w are leaves, or u is a leaf and v is not connected to any vertex other than u or w. 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 $\{u,v\}$ of T, one at a time, under the condition that u 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 3, computing $Z_{LP}(F^K_{GEN})$ by explicitly solving LPs is impractical, due to the large number of variables and constraints included in F^K_{GEN}. To provide a more practical approach to approximate such bounds, once again, we consider relaxing (21) in a Lagrangian fashion. In doing this, we obtain a Lagrangian relaxation subproblem whose solution is given by the generalization of Algorithm 1, discussed at the beginning of this section and detailed next.

Consider again that unconstrained multipliers $\theta = (\theta_{ef})_{e<f \in E}$, defined as before, are assigned to (21). The Lagrangian relaxation of constraints (21) yields the following Lagrangian subproblem:

$$LRP_{GEN}(\theta): \quad \min\left\{ \sum_{e,f \in E} q'_{ef} y_{ef} : (\mathbf{x},\mathbf{y},\mathbf{s},\mathbf{t}) \in P^{LRP}_{GEN} \cap \mathbb{B}^{m+m^2+o+om} \right\},$$

where P^{LRP}_{GEN} is obtained by relaxing (21) in P^K_{GEN}, i.e., P^{LRP}_{GEN} is represented by (17) and (20). Lagrangian modified costs q'_{ef} are defined as before.

Observe that, by (20), the objective function of $LRP_{GEN}(\theta)$ can be written as

$$\sum_{e,f \in E} q'_{ef} y_{ef} = \sum_{e,f \in E} \sum_{H \in E^K_e} q'_{ef} t_{Hf} = \sum_{H \in E^K} \sum_{e \in H} \sum_{f \in E} q'_{ef} t_{Hf}.$$

Therefore, using the fact that in $LRP_{GEN}(\theta)$ the choice of the spanning tree $\mathbf{t}_H$ depends only on s_H, it follows that in an optimal solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ for $LRP_{GEN}(\theta)$, if $\overline{s}_H = 1$, $\overline{\mathbf{t}}_H$ will be the incidence vector of a spanning tree that minimizes:

$$p_H = \min\left\{ \sum_{e \in H} \sum_{f \in E} q'_{ef} t_{Hf} : \mathbf{t}_H \in X_H \cap \mathbb{B}^m \right\}. \tag{28}$$

Thus, problem $LRP_{GEN}(\theta)$ can be solved with the resolution of

$$\overline{p} = \min\left\{ \sum_{H \in E^K} p_H s_H : \mathbf{x} \in X; x_e = \sum_{H \in E^K_e} s_H, \forall e \in E; (\mathbf{x},\mathbf{s}) \in \mathbb{B}^{m+o} \right\}, \tag{29}$$

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 $\theta \in \mathbb{R}^{\frac{m(m-1)}{2}}$.
Output: A Solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ for $LRP_{GEN}(\theta)$.
1. For every $H \in E^K$, solve (28) and denote by $\tilde{\mathbf{t}}_H$ its solution
   vector.
2. Solve problem (29) to obtain a solution $(\tilde{\mathbf{x}},\tilde{\mathbf{s}})$.
3. Obtain a solution $(\overline{\mathbf{x}},\overline{\mathbf{y}},\overline{\mathbf{s}},\overline{\mathbf{t}})$ of cost $Z(LRP_{GEN}(\theta)) = \overline{p}$ for
   $LRP_{GEN}(\theta)$ by setting $\overline{\mathbf{x}} = \tilde{\mathbf{x}}$, $\overline{\mathbf{s}} = \tilde{\mathbf{s}}$, $\overline{\mathbf{t}}_H = \overline{s}_H \tilde{\mathbf{t}}_H$ for all $H \in E^K$, and
   $\overline{\mathbf{y}}_e = \sum_{H \in E^K_e} \overline{\mathbf{t}}_H$ for all $e \in E$.
```

While Algorithm 2 actually solves $LRP_{GEN}(\theta)$, the problem is in fact NP-Hard for $K \ge 3$ [13]. The problem of deciding whether a

$(K+1)$-uniform hypergraph has a spanning tree is an NP-Hard problem for $K \geq 3$ that can be polynomially reduced to $LRP_{GEN}(\theta)$. Consequently, it is unlikely that one can come up with an efficient algorithm to solve step 2, in particular. Thus, given the complexity of solving $LRP_{GEN}(\theta)$, we study an alternative approach to derive lower bounds from F^K_{GEN}.

3.1. Selecting Edge-disjoint K-Forests

Consider the subtour elimination constraints (SECs) (2). For integer solutions of F^K_{GEN}, these constraints are already implied by other constraints in the formulation. To check this, observe that

$$y_{ef} = y_{fe} \leq x_f, \quad e, f \in E,$$

and as $\mathbf{y}_e$ defines a spanning tree, $\mathbf{x}$ also defines a spanning tree. Moreover, by further relaxing and dualizing (2) in $LRP_{GEN}(\theta)$, the resulting problem consists of selecting $(n-1)/K$ non-overlapping K-forests, i.e., their union does 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). As before, let $\theta = (\theta_{ef})_{e < f \in E}$ be unconstrained multipliers associated to (21). After the two sets of constraints are relaxed and dualized, the following Lagrangian subproblem results

$$LRP_{SEC}(\theta, \mu): \quad C + \min\left\{ \sum_{e,f \in E} q'_{ef} y_{ef} : (\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}) \in P^{LRP}_{SEC} \cap \mathbb{B}^{m+m^2+o+om} \right\},$$

where P^{LRP}_{SEC} is obtained by relaxing (21) and (2) in P^K_{GEN}, i.e., P^{LRP}_{SEC} is defined by (1), (3), (18)–(20). Lagrangian costs q'_{ef} are defined as $q'_{ef} = q_{ef} + \theta_{ef}$, for distinct edges e and f, $q'_{ee} = q_{ee} + \sum_{S \subset V, |S| \geq 2, e \in E(S)} \mu_S$, for each edge e, and $C = -\sum_{S \subset V, |S| \geq 2} \mu_S (|S|-1)$.

The associated Lagrangian dual is

$$LDP_{SEC}: \quad \max\left\{ Z(LRP_{SEC}(\theta, \mu)) : (\theta, \mu) \in \mathbb{R}^{\frac{m(m-1)}{2}} \times \mathbb{R}_+^{|\{S \subset V, |S| \geq 2\}|} \right\}.$$

Let $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ be an optimal solution for $LRP_{SEC}(\theta, \mu)$. 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 (28). Also, after the relaxation of (2), variables $\mathbf{x}$ can be eliminated from the formulation. In this way, $LRP_{SEC}(\theta, \mu)$ can be solved with the resolution of

$$\overline{p} = C + \min\left\{ \sum_{H \in E^K} p_H s_H : \sum_{H \in E^K_e} \mathbf{s}_H \leq 1, \forall e \in E; \sum_{H \in E^K} \mathbf{s}_H = \frac{n-1}{K}; \mathbf{s} \in \mathbb{B}^o \right\}, \tag{30}$$

and the appropriate adjustment of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{t}$. Problem (30) asks for the optimal selection of $(n-1)/K$ K-forests of G such that no edge e appears in more than one of them. In other words, (30) is a set packing problem with an additional cardinality constraint. This problem can be efficiently solved for $K=2$ but is NP-Hard for $K \geq 3$ [13]. The problem of deciding whether a K-uniform hypergraph has a perfect matching is a NP-Hard problem for $K \geq 3$ that can be polynomially reduced to (30).

If $K=2$, problem (30) 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 $\overline{E} = E^K$. Finding a matching with fixed cardinality of a graph can be done in polynomial time; an algorithm to that aim is given in [25]. That algorithm is used as the basis of Algorithm 3 outlined below, for the resolution of (30).

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 (30).
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 $p_H=0$ 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 $e \in E$, if there is an $H \in E^K_e$ such that
    $s_H=1$, set $\overline{x}_e = 1$. Otherwise, set $\overline{x}_e = 0$.
```

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 (30). In order to solve $LRP_{SEC}(\theta, \mu)$ for $K=2$, we can proceed by computing the costs (28), followed by the resolution of (30) 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 $(\theta, \mu) \in \mathbb{R}^{\frac{m(m-1)}{2}} \times \mathbb{R}_+^{|\{S \subset V, |S| \geq 2\}|}$.
Output: Solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ for $LRP_{SEC}(\theta, \mu)$ for $K=2$.
1. Solve problem (28) for each $H \in E^K$ and let $\tilde{\mathbf{t}}_H$ be the
    minimizing vector.
2. Solve (30) by Algorithm 3. Let $(\tilde{\mathbf{x}}, \tilde{\mathbf{s}})$ denote a solution.
3. Obtain a solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}}, \overline{\mathbf{s}}, \overline{\mathbf{t}})$ of cost $Z(LRP_{SEC}(\theta, \mu)) = \overline{p}$ for
    $LRP_{SEC}(\theta, \mu)$ by setting $\overline{\mathbf{s}} = \tilde{\mathbf{s}}$ and $\overline{\mathbf{x}} = \tilde{\mathbf{x}}$. Let $\overline{\mathbf{t}}_H = \overline{s}_H \tilde{\mathbf{t}}_H$ for all
    $H \in E^K$, and $\overline{\mathbf{y}}_e = \sum_{H \in E^K_e} \overline{\mathbf{t}}_e$ for all $e \in E$.
```

The first step of Algorithm 4 can be implemented to run in $O(om \log n) = O(m^3 \log n)$ time complexity, while Algorithm 3 can be implemented to run in $O(|\overline{V}|^2 |\overline{E}|) = O(m^4)$ [26]. This gives the complexity of step 2, which determines the overall worst case time complexity of Algorithm 4.

From Lagrangian relaxation theory we know that

Proposition 4. $Z(LDP_{SEC}) \geq Z_{LP}(F^K_{GEN})$.

In fact, as discussed above for $K=2$, the solutions for the Lagrangian subproblem $LRP_{SEC}(\theta, \mu)$ implicitly satisfy the blossom inequalities [27] (facet defining inequalities for the matching polytope), which are missing from P^{LRP}_{SEC}.

The evaluation of $Z(LDP_{SEC})$ requires finding optimal multipliers for an exponential number of constraints (2). One of the known algorithmic alternatives to deal with exponentially many inequalities candidates to Lagrangian dualization is the relax-and-cut approach [28]. Due to the already excessive (though polynomial in n and m) number of other dualized constraints, the benefits of implementing a relax-and-cut algorithm for the evaluation of

$Z(LDP_{SEC})$ 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 LDP_{SEC} 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_{SEC}. A BB algorithm based on LAG_{SEC} is presented in Section 4. We report computational results for LAG_{SEC} in Section 5.

4. Branch-and-bound algorithms

In this section, we describe the main implementation details of two BB algorithms, BB_{RLT} and BB_{SEC}, respectively based on the Lagrangian relaxation lower bounding procedures LAG_{RLT} and LAG_{SEC}. BB_{RLT} and BB_{SEC} are quite similar, differing only when explicitly mentioned in the exposition that follows.

4.1. Initial upper bounds

The following multi-start heuristic is used by our exact solution algorithms to obtain an initial upper bound.

The first phase of the heuristic consists of finding a minimum cost spanning tree $T=(V,E_T)$ of G under randomly chosen weights. To that aim, each edge of E is assigned a randomly chosen weight in the set $\{1,\dots,100\}$. Then, to obtain T, we compute the minimum spanning tree of G under these weights, using Prim's algorithm.

Given the current spanning tree $T=(V,E_T)$, we check the inclusion of each edge $e \in E \setminus E_T$, one by one. For each edge, we identify the best spanning tree T', under the original quadratic cost function, that can be obtained by replacing an edge $f \in E_T$ by e. This is easily accomplished by letting T' be the spanning tree with the smallest quadratic cost that can be obtained by replacing some edge in the cycle of $(V, E_T \cup \{e\})$ by e. If T' is better than T, then T is replaced by T'. The process is repeated until all edges $e \in E \setminus E_T$ have been investigated and no improving spanning tree could be found.

The two BB algorithms call the heuristic above 100 times. The best solution found is then used as an initial QMSTP upper bound.

4.2. Lower bounds and node selection

Algorithm BB_{RLT} makes use of LAG_{RLT} as its bounding procedure, while algorithm BB_{SEC} employs LAG_{SEC}. 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. In 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.

4.3. Branching and variable selection

Assume that $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ denotes the solution to the Lagrangian Relaxation problem ($LRP_{RLT}(\theta)$ for BB_{RLT} and $LRP_{SEC}(\theta,\mu)$ for BB_{SEC}) that provided the best Lagrangian lower bound at a given BB node.

If $\overline{y}_{ef} = \overline{y}_{fe}$ for all pairs of distinct edges e and f, the subproblem at the current node has been solved to optimality. Otherwise, there is either (i) an edge e such that $\overline{x}_e = 1$ and $\overline{y}_{fe} = 0$ for an edge f with $\overline{x}_f = 1$ (edge e was selected but was not used in some interaction tree) or (ii) an edge e such that $\overline{x}_e = 0$ and $\overline{y}_{fe} = 1$ for an edge f with $\overline{x}_f = 1$ (edge e was not selected but was used in some interaction tree). Any edge e meeting one of these two cases 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, we forbid the selection of the edge.

The way branching constraints are enforced is one of the few differences between BB_{RLT} and BB_{SEC}. For BB_{RLT}, 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_{SEC}, if an edge e is imposed (resp. forbidden) it is necessary to grant that, in Algorithm 3, exactly one (resp. none) of the 2-forests containing e is (resp. are) selected. To guarantee such a condition, we remove from the auxiliary graph $\overline{G}$ any edge that connects e to a vertex of U (resp. $\overline{V}$).

In order to select the branching variable, we do the following. For each variable candidate to branching, we solve the corresponding Lagrangian subproblems that result when the variable is imposed and forbidden. When solving these problems, the multipliers of the current node are used, but no multiplier adjustment is further implemented. If any of the resulting lower bounds is larger than the best known upper bound, the variable is then fixed accordingly. If any variable is fixed when these bounds are computed, we postpone branching and apply another round of subgradient optimization to the current node, starting with the best multipliers obtained during the previous round. Otherwise, we select the branching variable according to the strong branching strategy [29], i.e., we select the variable for which the minimum between the two bounds is maximum.

4.4. Redistributing the costs of fixed variables

Assume that at a given BB node, a nonempty set F of edges is known to be included in the (integer) solution for that node. The objective function can be written as

$$\begin{aligned}
&\sum_{e \in E\setminus F}\sum_{f \in E} q_{ef}x_ex_f + \sum_{e \in F}\sum_{f \in E} q_{ef}x_ex_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E} q_{ef}x_ex_f + \frac{n-1}{n-1}\sum_{e \in F}\sum_{f \in E} q_{ef}x_ex_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E} q_{ef}x_ex_f + \frac{\sum_{g \in E} x_g}{n-1}\sum_{e \in F}\sum_{f \in E} q_{ef}x_ex_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E} q_{ef}x_ex_f + \frac{\sum_{e \in E} x_e}{n-1}\sum_{g \in F}\sum_{f \in E} q_{gf}x_kx_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E} q_{ef}x_ex_f + \frac{\sum_{e \in E\setminus F} x_e}{n-1}\sum_{g \in F}\sum_{f \in E} q_{gf}x_kx_f \\
&\qquad + \frac{\sum_{e \in F} x_e}{n-1}\sum_{g \in F}\sum_{f \in E} q_{gf}x_kx_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E}\left(q_{ef} + \frac{\sum_{g \in F} q_{gf}x_g}{n-1}\right)x_ex_f + \sum_{e \in F}\sum_{f \in E}\frac{\sum_{g \in F} q_{gf}x_g}{n-1}x_ex_f.
\end{aligned}$$

Recall that $x_e = 1$ for $e \in F$ and, thus:

$$\begin{aligned}
&\sum_{e \in E\setminus F}\sum_{f \in E}\left(q_{ef} + \frac{\sum_{g \in F} q_{gf}}{n-1}\right)x_ex_f + \sum_{e \in F}\sum_{f \in E}\frac{\sum_{g \in F} q_{gf}}{n-1}x_ex_f \\
&\quad = \sum_{e \in E\setminus F}\sum_{f \in E}\left(q_{ef} + \frac{\sum_{g \in F} q_{gf}}{n-1}\right)y_{ef} + \sum_{e \in F}\sum_{g \in E}\frac{\sum_{f \in F} q_{fg}}{n-1}y_{ef}.
\end{aligned}$$

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

The best Lagrangian lower bounds are not affected by the reformulation indicated above. However, the multipliers obtained with the subgradient method are usually not the optimal ones but (hopefully good) approximations of them. Besides, convergence to the optimal set of multipliers is often an issue when a large number of inequalities are dualized. Thus, the reformulation above might help to obtain sharper approximations of the true lower bounds, earlier in the application of the subgradient method.

4.5. Parallelization

Our computational experiments are conducted in a multi-processor shared memory system. In order to take advantage of such a hardware, our BB implementations use parallel programming techniques, following the guidelines proposed in [30], for a parallel QAP BB algorithm. We give a brief description of the parallelization scheme in [30] and show how we improved that implementation, by introducing an effective load balancing mechanism.

As in [30], 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 [30], a processor stops whenever it detects 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 or not 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.

As a result of new features suggested here for load balancing, our parallel algorithms obtain rates of parallel efficiency around 80%.

5. Computational experiments

In this section, we report our computational experience with the lower bounding procedures and with the BB algorithms introduced in this paper. We compare our methods to other QMSTP approaches in the literature.

5.1. 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 [9]. These instances comprise graphs with $n \in \{10, 15, \ldots, 50\}$ and densities $d \in \{33\%, 67\%, 100\%\}$. Depending on how the diagonal (q_{ee}) and the off-diagonal (q_{ef}) 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_{ee} and q_{ef} correspond to integers randomly chosen from $\{1, \ldots, 10\}$. Similarly, for CP2, $q_{ee} \in \{1, \ldots, 10\}$ and $q_{ef} \in \{1, \ldots, 100\}$. For CP3, $q_{ee} \in \{1, \ldots, 100\}$ and $q_{ef} \in \{1, \ldots, 10\}$. Finally, for CP4, q_{ee} and q_{ef} are integers in $\{1, \ldots, 100\}$.

The second set, denoted by OP, was introduced by Öncan and Punnen [6]. 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_{ee} and q_{ef} 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. Given two different edges $e = \{a, b\}$ and $f = \{c, d\}$, $q_{ef} = w_a w_b w_c w_d$. For an edge e, q_{ee} is an integer randomly chosen from $\{1, \ldots, 10,000\}$. For OP3, each vertex represents a 2-dimensional point with coordinates randomly chosen in the interval [0,100]. The value of q_{ee} is given by the Euclidean distance between the endpoints of e, while q_{ef}, for distinct e and f, is the distance between the midpoints of e and f.

5.2. Computational results

Computational experiments were performed on a machine with the following configuration: Two Intel Xeon processors, each one with six cores running at 2.4 GHz, 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.

The implementation of our subgradient method follows the particular settings described next. In every iteration, the normalized subgradient is the direction used in the search for the next set of multipliers. At the root node, 5000 iterations of the subgradient method are performed. The usual parameter that controls the step size in subgradient methods is initialized at value 2 and then halved whenever 500 iterations have passed without improving the best Lagrangian bound. For non-root nodes, the subgradient method runs for at most 100 iterations and the step size control parameter is halved every 10 iterations without improvement in the best bound.

In Table 1, we present a comparison of lower bounds for some selected instances in our test bed. We report lower bounds for the Lagrangian relaxation schemes of Assad and Xu [1], which we denote LAG_{AX92}, Öncan and Punnen [6], which we denote LAG_{OP10}, LAG_{RLT}, and LAG_{SEC}. The lower bounds we report for LAG_{AX92} and LAG_{OP10} were evaluated by ourselves. We implemented the Lagrangian relaxation algorithms as described in those references. We found differences between the bounds we evaluated and those reported by Öncan and Punnen [6]. In fact, the bounds reported for LAG_{OP10} in [6] quite often exceed valid QMSTP upper bounds, which is impossible. For an in-depth discussion of this matter, we refer the reader to [13] and also to [9].

The first four columns of Table 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 lower bounding procedure.

In Table 2, we compare algorithms BB_{RLT}, BB_{SEC}, and BB_{CP}, the BB algorithm in [9]. For BB_{RLT} and BB_{SEC}, a time limit of 100 h was imposed. The stopping criteria for BB_{CP}, however, were not the same for all instances. For CP instances, that algorithm was stopped after a time limit of 3600 s was reached. For OP instances with $n \le 15$, that algorithm was stopped after the investigation of 10^8 nodes. For instances with $n > 15$, that algorithm was stopped after the investigation of 10^6 nodes. Differently from Table 1, where computational results indicated in each row refer to a particular instance, rows in Table 2 present results aggregated for several instances, in the ranges of n and m indicated in the table.

The first three columns of Table 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

Table 1
Lower bound comparisons.

Instance				LAG_{AX92}		LAG_{OP10}		LAG_{RLT}		LAG_{SEC}	
n	m	type	ub	lb	t	lb	t	lb	t	lb	t
25	100	CP1	2185	1115.5	0	1193.2	3	1285.1	2	1718.7	40
25	100	CP2	19,976	8170.6	0	8988.8	3	10,061.1	2	14,860.8	52
25	100	CP3	2976	2069.1	0	1961.3	3	2289.2	2	2652.5	41
25	100	CP4	21,176	9296.8	0	10,089.2	3	11,190	2	15,977	55
25	200	CP1	2023	755.1	0	801.3	12	828	3	1316.7	305
25	200	CP2	18,251	4154.4	0	4564.5	12	5028.6	3	10,497.4	362
25	200	CP3	2546	1468.1	0	1385.3	12	1626.8	3	2071	261
25	200	CP4	19,207	5183.6	0	5560.5	12	6065.4	3	11,522	360
25	300	CP1	1943	668.5	0	705.2	23	715.4	6	1143.2	1012
25	300	CP2	17,411	2879.4	0	3161.8	24	3443.2	6	8533.5	1118
25	300	CP3	2471	1279.2	0	1213.9	23	1405.6	6	1875.3	918
25	300	CP4	18,370	3865.2	0	4086.6	24	4451.3	6	9563.4	1135
13	78	OP1	1022	513.7	0	475.9	2	606.3	0	842.6	21
13	78	OP1	1089	592.9	0	532.1	2	702	0	900.1	22
13	78	OP1	1163	609.8	0	576.3	2	697	0	945.2	22
13	78	OP1	1129	703.5	0	659	1	803.1	0	1033.3	21
13	78	OP1	1023	663.2	0	588.4	2	748.8	0	1001.4	21
13	78	OP1	982	586.9	0	546.2	1	715.4	0	933	21
13	78	OP1	1048	520.8	0	466	2	613.3	0	838.3	22
13	78	OP1	1045	611.8	0	571.1	2	712.8	0	929.4	22
13	78	OP1	1065	637.6	0	594.3	2	741.2	0	980.3	22
13	78	OP1	1160	618.2	0	572.1	2	720.3	0	978.6	21
13	78	OP2	45,586	44,885	0	44,693	1	45,586	0	45,586	12
13	78	OP2	49,313	48,747.1	0	45,717	1	49,313	0	49,313	11
13	78	OP2	44,513	44,257.5	0	43,676.5	1	44,513	0	44,513	11
13	78	OP2	37,250	37,250	0	37,054	1	37,250	0	37,250	11
13	78	OP2	50,990	49,908	0	46,969	1	50,990	0	50,990	11
13	78	OP2	43,261	42,380	0	41,140	1	43,261	0	43,261	12
13	78	OP2	36,085	35,809.1	0	35,135	1	36,085	0	36,085	11
13	78	OP2	34,474	34,442.6	0	33,775	1	34,474	0	34,474	10
13	78	OP2	28,566	28,360.2	0	27,653	1	28,566	0	28,566	10
13	78	OP2	34,847	34,493	0	33,909	1	34,847	0	34,847	13
13	78	OP3	1731	1595.6	0	1648.3	1	1731	0	1731	9
13	78	OP3	2484	2341.4	0	2318.7	1	2484	0	2484	10
13	78	OP3	2440	2228.8	0	2297.2	1	2436.6	0	2440	12
13	78	OP3	2489	2307.5	0	2272.5	1	2453.2	0	2483	23
13	78	OP3	2044	1932.8	0	1915	1	2044	0	2044	11
13	78	OP3	1806	1655.7	0	1634	1	1805	0	1806	11
13	78	OP3	2185	2041.9	0	2035	1	2185	0	2185	10
13	78	OP3	2275	2081	0	2134.2	1	2272.8	0	2275	11
13	78	OP3	1968	1741.5	0	1857.6	1	1943.1	0	1957.7	21
13	78	OP3	2331	2241.4	0	2252	1	2331	0	2331	10

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 produced by LAG_{RLT} are much stronger than the previous bounds in the literature. For OP1 instances, for example, the bounds produced by LAG_{RLT} are 16.6% stronger than those produced by LAG_{AX92} and 26.7% stronger than those produced by LAG_{OP10}. The procedure LAG_{RLT} seems to offer a good trade-off between lower bound quality and computational effort. That claim is validated by how BB_{RLT} and BB_{SEC} do compare one another.

The procedure LAG_{SEC}, which is based on formulation F^{K}_{GEN}, provides lower bounds that are significantly stronger than those provided by LAG_{RLT}, that relies on formulation F_{RLT}. The results in Table 1 show that, for CP instances, the bounds produced by LAG_{SEC} are, on average, 65% stronger than those produced by LAG_{RLT}, 90.6% stronger than those produced by LAG_{AX92}, and 81.2% stronger than those produced by LAG_{OP10}. Consequently, the number of nodes investigated by BB_{SEC} is orders of magnitude smaller than BB_{RLT} and BB_{CP} counterparts. However, BB_{SEC} is dominated by BB_{RLT} in terms of computational time, due to the high costs demanded to run LAG_{SEC}.

Our algorithms were able to solve several new instances to proven optimality, namely: all 4 CP instances with $n=20$ vertices and 67% graph density, all 30 OP1 instances with $16 \leq n \leq 18$ (instances with $n=20$ were not available to us), one OP3 instance with $n=17$, one OP3 instance with $n=18$, all 20 OP3 instances with $n=30$ and $n=50$ and all 20 OP2 instances with $n=30$ and $n=50$. Detailed computational results for all these instances are provided in Tables 3–6. For both BB_{RLT} and BB_{SEC}, we present the lower bound at the root node (lb_{root}), the time (in seconds) taken to solve the root node ($t_{root}(s)$), the total number of nodes investigated in the search tree (n_{nodes}) and the total time taken (in seconds) to solve the problem ($t(s)$). An entry "-" indicates that the corresponding algorithm was not able to solve the instance within the specified time limit.

The use of different computational environments and different stopping criteria make a precise CPU time comparisons of our algorithms and those in the literature very hard to conduct. Furthermore, our BB algorithms were executed on a parallel environment, with parallel efficiency around 80%. In our point of view, the fact that the search trees in BB_{RLT} and BB_{SEC} were implemented in parallel certainly improved our results, since the methods could better exploit the strength of the formulations they rely on. However, parallel programming is not the solely responsible for the new optimality certificates we provide here. The quality of the lower bounds we proposed was also a key factor.

Table 2
Comparison of branch-and-bound algorithms.

Instance			BB_{CP}			BB_{RLT}			BB_{SEC}		
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	51,880,837	887	28/28	144,309	946	28/28	24,106	3170
20	127	CP	0/4	–	–	4/4	24,431,331	271,761	0/4	–	–
10–15	45–105	OP1	60/60	7,922,195	184	60/60	19,239	174	60/60	4057	775
16–17	120–136	OP1	0/20	–	–	20/20	449,565	6386	20/20	48,463	20,080
18	153	OP1	0/10	–	–	10/10	5,351,735	93,178	0/10	–	–
10–20	45–190	OP2	100/100	583,379	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	979,125	36	100/100	21	12	100/100	13	218
30	435	OP3	0/10	–	–	10/10	129	491	10/10	81	17,857
50	1225	OP3	0/10	–	–	10/10	735	19,045	0/10	–	–

Table 3
QMSTP BB results for instances of Cordone and Passeri [9] solved for the first time.

Instance				BB_{RLT}				BB_{SEC}			
n	m	*type*	*ub*	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
20	67	CP1	1252	598.3	2	24,431,331	271,760	–	–	–	–
20	67	CP2	10,893	3797.3	1	15,202,397	168,843	–	–	–	–
20	67	CP3	1792	1306.1	2	89,595	1527	–	–	–	–
20	67	CP4	11,893	4671.6	1	21,244,515	238,189	–	–	–	–

Table 4
QMSTP BB results for type 1 instances of Öncan and Punnen [6] solved for the first time.

Instance				BB_{RLT}				BB_{SEC}			
n	m	*type*	*ub*	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	66,679	733	1241.7	73	14,389	4766
16	120	OP1	1667	913	1	87,299	923	1297.6	74	18,641	5928
16	120	OP1	1629	900.2	1	44,877	473	1265.1	84	8993	2891
16	120	OP1	1659	903.3	1	72,119	783	1271.4	78	20,374	6708
16	120	OP1	1695	930.7	1	67,577	733	1311.9	69	15,880	5170
16	120	OP1	1518	833.4	1	27,547	289	1225.2	72	2449	948
16	120	OP1	1652	864.8	1	128,423	1365	1256.1	74	21,287	7029
16	120	OP1	1687	959.2	1	43,201	457	1347.2	73	7975	2626
16	120	OP1	1543	785	1	72,719	777	1162.2	71	19,273	6233
16	120	OP1	1619	887.3	1	42,193	454	1266.4	76	7445	2530
17	136	OP1	1843	973.2	1	191,595	2712	1415.6	104	24,776	10,771
17	136	OP1	1828	984.8	1	101,253	1546	1441.4	95	11,588	5171
17	136	OP1	1859	1021.3	1	127,597	1848	1459.8	98	15,239	6611
17	136	OP1	1839	941.4	1	287,547	4263	1398.2	97	38,661	16,345
17	136	OP1	1795	904.8	1	449,565	6385	1359.8	116	46,463	18,853
17	136	OP1	1817	946.1	1	215,273	3168	1388	98	29,827	13,176
17	136	OP1	1893	980.4	1	382,019	5370	1438.8	89	48,463	20,079
17	136	OP1	1818	987.4	1	109,887	1602	1436.4	104	12,866	5936
17	136	OP1	1734	932.4	1	67,545	1098	1369.1	97	7507	3392
17	136	OP1	1812	922.7	1	242,045	3582	1365.3	103	35,631	15,514
18	153	OP1	2153	1075.6	2	1,516,447	26,497	–	–	–	–
18	153	OP1	2125	1037.5	2	1,890,921	32,555	–	–	–	–
18	153	OP1	2108	1094.6	2	568,609	9659	–	–	–	–
18	153	OP1	2026	1029.8	2	857,377	14,439	–	–	–	–
18	153	OP1	2028	941.8	2	1,238,159	23,234	–	–	–	–
18	153	OP1	2023	976.4	2	1,121,977	20,526	–	–	–	–
18	153	OP1	1951	961.2	2	552,413	9755	–	–	–	–
18	153	OP1	2089	957.3	2	3,191,969	55,061	–	–	–	–
18	153	OP1	2138	995.5	2	5,351,735	93,178	–	–	–	–
18	153	OP1	2169	1141	2	1,010,763	18,211	–	–	–	–

6. Conclusion

In this paper, we investigated formulations and exact solution approaches for the quadratic minimum spanning tree problem.

Initially, we introduced a 0–1 IP formulation based on RLT and we derived a Lagrangian relaxation algorithm based on it. A BB algorithm that uses lower bounds based on that Lagrangian relaxation was implemented. We also introduced a reformulation

Table 5
QMSTP BB results for type 2 instances of Öncan and Punnen [6] solved for the first time.

Instance				BB_{RLT}				BB_{SEC}			
n	m	*type*	*ub*	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
30	435	OP2	82,953	82,953	12	1	85	82,953	1383	1	1454
30	435	OP2	76,977	76,977	12	1	115	76,977	1186	1	1287
30	435	OP2	88,098	88,096	13	1	86	88,096	1826	1	1899
30	435	OP2	90,361	90,361	12	1	99	90,361	1319	1	1410
30	435	OP2	69,976	69,976	12	1	117	69,976	1327	1	1429
30	435	OP2	78,864	78,864	12	1	150	78,864	1245	1	1383
30	435	OP2	73,015	73,015	12	1	111	73,015	1112	1	1209
30	435	OP2	73,619	73,619	12	1	133	73,619	1170	1	1287
30	435	OP2	81,534	81,534	12	1	121	81,534	1393	1	1501
30	435	OP2	74,602	74,602	12	1	86	74,602	1362	1	1437
50	1225	OP2	172,157	172,157	177	1	1338	–	–	–	–
50	1225	OP2	170,915	170,915	175	1	1317	–	–	–	–
50	1225	OP2	160,256	160,256	176	1	1503	–	–	–	–
50	1225	OP2	152,830	152,830	174	1	1389	–	–	–	–
50	1225	OP2	174,926	174,926	176	1	1133	–	–	–	–
50	1225	OP2	154,341	154,341	173	1	1731	–	–	–	–
50	1225	OP2	180,023	180,023	176	1	1214	–	–	–	–
50	1225	OP2	153,578	153,578	175	1	1295	–	–	–	–
50	1225	OP2	179,932	179,932	174	1	1399	–	–	–	–
50	1225	OP2	155,241	155,241	175	1	1456	–	–	–	–

Table 6
QMSTP BB results for type 3 instances of Öncan and Punnen [6] solved for the first time.

Instance				BB_{RLT}				BB_{SEC}			
n	m	*type*	*ub*	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$	lb_{root}	$t_{root}(s)$	n_{nodes}	$t(s)$
17	136	OP3	3734	3724.9	1	3	5	3734	64	1	67
18	153	OP3	4060	4044.7	2	1	10	4058.5	148	1	167
30	435	OP3	6992	6865.3	15	35	214	6964.2	2437	15	5157
30	435	OP3	9057	8700.3	16	129	491	8859.5	2069	81	17,856
30	435	OP3	7823	7823	13	1	179	7823	1233	1	1395
30	435	OP3	7936	7886.1	14	1	200	7936	1214	1	1392
30	435	OP3	8092	8042.1	18	9	272	8092	1380	1	1597
30	435	OP3	8566	8438.3	15	15	223	8533.5	2928	9	5691
30	435	OP3	7525	7364.4	14	41	342	7472.3	2079	31	4958
30	435	OP3	8645	8409.5	15	49	318	8545.9	2211	29	8001
30	435	OP3	8692	8526.1	15	31	296	8653.4	2348	28	5015
30	435	OP3	7239	7209.2	14	1	197	7239	1422	1	1601
50	1225	OP3	17,524	16,933.3	195	735	19,045	–	–	–	–
50	1225	OP3	16,780	16,558.9	192	13	6061	–	–	–	–
50	1225	OP3	13,198	12,940.4	189	71	8244	–	–	–	–
50	1225	OP3	15,137	14,708.5	193	123	11,761	–	–	–	–
50	1225	OP3	16,358	15,677.9	195	405	14,950	–	–	–	–
50	1225	OP3	14,996	14,781.4	190	21	8791	–	–	–	–
50	1225	OP3	17,282	17,222.5	188	1	5219	–	–	–	–
50	1225	OP3	14,975	14,959.9	185	1	4856	–	–	–	–
50	1225	OP3	13,594	13,591.8	185	1	3920	–	–	–	–
50	1225	OP3	18,062	17,736	193	111	8141	–	–	–	–

for the problem, using the idea of partitioning spanning trees into forests of a given fixed size. We ended up with a reformulation that generalizes the RLT model. One particular case of the generalization was used to derive another Lagrangian relaxation lower bounding procedure. A second BB algorithm, based on that procedure, was also implemented.

Although the Lagrangian bounds obtained with the second lower bounding procedure are significantly stronger than those provided by the first, the second BB algorithm was dominated by the first in terms of overall running time. That happens because the evaluation of lower bounds by our second reformulation demands excessive CPU running times. The bounding procedure based on the application of the RLT seems to offer a better trade-off between lower bound quality and computational effort. As a result, the associated BB algorithm managed to solve, for the first time, several instances in the literature, including some with $n=50$ vertices.

References

[1] Assad A, Xu W. The quadratic minimum spanning tree problem. Naval Res Logist 1992;39(3):399–417.
[2] Prim R. Shortest connection networks and some generalizations. Bell Syst Tech J 1957;36:1389–401.
[3] Kruskal J. On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc 1956;7:48–50.
[4] Zhout G, Gen M. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. Comput Oper Res 1998;25(3):229–37.
[5] Soak S-M, Corne D, Ahn B-H. The edge-window-decoder representation for tree-based problems. IEEE Trans Evol Comput 2006;10(2):124–44.
[6] Öncan T, Punnen AP. The quadratic minimum spanning tree problem: a lower bounding procedure and an efficient search algorithm. Comput Oper Res 2010;37(10):1762–73.
[7] Sundar S, Singh A. A swarm intelligence approach to the quadratic minimum spanning tree problem. Inf Sci 2010;180(17):3182–91.
[8] Palubeckis G, Rubliauskas D, Targamadzė A. Metaheuristic approaches for the quadratic minimum spanning tree problem. Inf Technol Control 2010;39:257–68.

[9] Cordone R, Passeri G. Solving the quadratic minimum spanning tree problem. Appl Math Comput 2012;218(23):11597–612.

[10] Gilmore PC. Optimal and suboptimal algorithms for the quadratic assignment problem. J Soc Ind Appl Math 1962;10(2):305–13.

[11] Lawler EL. The quadratic assignment problem. Manag Sci 1963;9(4):586–99.

[12] Pisinger D. The quadratic knapsack problem—a survey. Discrete Appl Math 2007;155(5):623–48.

[13] Pereira DL. Formulations and algorithms based on linear integer programming for the quadratic minimum spanning tree problem [Ph.D. thesis]. Universidade Federal de Minas Gerais, Brazil; 2014.

[14] Maia SMDM, Goldbarg EFG, Goldbarg MC. On the biobjective adjacent only quadratic spanning tree problem. Electron Notes Discrete Math 2013;41:535–42 presented at the 2013 International Network Optimization Conference (INOC).

[15] Pereira DL, Gendreau M, Cunha AS. Branch-and-cut and Branch-and-cut-and-price Algorithms for the Adjacent Only Quadratic Minimum Spanning Tree Problem, Networks; 2015, http://dx.doi.org/10.1002/net.21580. in press.

[16] Padberg M. The boolean quadric polytope: some characteristics, facets and relatives. Math Progr 1989;45(1–3):139–72.

[17] Lee J, Leung J. On the Boolean quadric forest polytope. INFOR 2004;42 (2):125–41.

[18] Adams WP, Sherali HD. A tight linearization and an algorithm for zero-one quadratic programming problems. Manag Sci 1986;32(10):1274–90.

[19] Sherali HD, Adams WP. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Discrete Appl Math 1994;52(1):83–106.

[20] Edmonds J. Matroids and the greedy algorithm. Math Progr 1971;1(1):127–36.

[21] Magnanti TL, Wolsey LA. Optimal trees. In: Ball M, Magnanti T, Monma C, Nemhauser G, editors. Network models, handbooks in operations research and management science, vol. 7. Elsevier; 1995. p. 503–615 Chap. 9.

[22] Caprara A. Constrained 0-1 quadratic programming: basic approaches and extensions. Eur J Oper Res 2008;187(3):1494–503.

[23] Adams WP, Sherali HD. A tight linearization and an algorithm for zero-one quadratic programming problems. Manag Sci 1986;32(10):1274–90.

[24] Held M, Wolfe P, Crowder H. Validation of subgradient optimization. Math Progr 1974;6:62–88.

[25] Plesník J. Constrained weighted matchings and edge coverings in graphs. Discrete Appl Math 1999;92(2–3):229–41.

[26] Gerards AMH. Matching. In: Ball M, Magnanti T, Monma C, Nemhauser G, editors. Network models, hand s in operations research and management science, vol. 7. Elsevier; 1995. p. 135–224 Chap. 3.

[27] Edmonds J. Maximum matching and a polyhedron with 0,1 vertices. J Res Natl Bur Stand 1965;69B:125–30.

[28] Lucena A. Non delayed relax-and-cut algorithms. Ann Oper Res 2005;140 (1):375–410.

[29] Achterberg T, Koch T, Martin A. Branching rules revisited. Oper Res Lett 2005;33(1):42–54.

[30] Mans B, Mautor T, Roucairol C. A parallel depth first search branch and bound algorithm for the quadratic assignment problem. Eur J Oper Res 1995;81 (3):617–28.