Lagrangian and Branch-and-Cut Approaches for Upgrading Spanning Tree Problems

Eduardo Álvarez-Miranda *[1] and Markus Sinnl†[2]

[1]*Department of Industrial Engineering, Universidad de Talca, Curicó, Chile*
[2]*Department of Statistics and Operations Research, Faculty of Business, Economics and Statistics, University of Vienna, Vienna, Austria*

Abstract

Problems aiming at finding budget constrained optimal upgrading schemes to improve network performance have received attention over the last two decades. In their general setting, these problems consist of designing a network and, simultaneously, allocating (limited) upgrading resources in order to enhance the performance of the designed network.

In this paper we address two particular optimal upgrading network design problems; in both cases, the sought layout corresponds to a spanning tree of the input network and upgrading decisions can be taken on nodes. We design Mixed Integer Programming-based algorithmic schemes to solve the considered problems: Lagrangian relaxation approaches and branch-and-cut algorithms. Along with the designed algorithms, different enhancements, including valid inequalities, primal heuristic and variable fixing procedures, are proposed.

Using two set of instances, we experimentally compare the designed algorithms and explore the benefits of the devised enhancements. The results show that the proposed approaches are effective for solving to optimality most of the instances in the testbed, or manage to obtain solutions and bounds giving very small optimality gaps. Besides, the proposed enhancements turn out to be beneficial for improving the performance of the algorithms.

1. Introduction, Motivation and Problem Definition

Different models for finding budget constrained optimal upgrading schemes to improve network performance have been proposed over the last two decades. Roughly speaking, these problems consist of designing a network and, simultaneously, allocating (limited) upgrading resources in order to enhance the network' performance (e.g., its cost efficiency, its connectivity, its stability, etc.). These type of problems, which can be generally referred to as *budget constrained network upgrading problems* [Krumke et al., 1999], can be found in several decision making contexts. For instance in a multicast communication setting, a backbone server broadcasts a signal to many subscribers; the layout of such communication network should be such that delays between the

*ealvarez@utalca.cl
†markus.sinnl@univie.ac.at

server and all subscribers must be minimal or bounded. In such a problem, the decision maker seeks for an arrangement of nodes, technologies, and connections such that a positive function of node delays is minimized or it fulfills a Quality-of-Service (QoS) requirement [Álvarez-Miranda et al., 2016]. Another application appears in electric power grids, where transformers and other costly appliances can be located along the network in order improve its performance by reducing the risks of shortages or frequency instability [Costa et al., 2011].

In this paper we address two particular optimal upgrading network design problems; in both cases, the sought layout corresponds to a spanning tree of the input network and upgrading decisions can be taken on nodes. These problems were originally proposed in [Krumke et al., 1999] and motivated by a telecommunication design problematic. The authors provided approximation algorithms and hardness results for the two variants. Before formally defining the addressed problems, we first present the considered upgrading model.

Node-based Upgrading Model Let $G = (V, E)$ be an undirected graph, where V is the set of nodes and E is the set of edges. For each edge $e \in E$, we are given three integers $d_e^0 \geq d_e^1 \geq d_e^2 \geq 0$; the value d_e^l (the l-upgrade of edge e) represents the *length* or *delay* of e if exactly l of its endpoints are upgraded. This means that upgrading a node $i \in V$ reduces the delay of all the edges that are incident to it. Additionally, for each node $i \in V$, we are given an *upgrade cost* $c_i \geq 0$ which must be paid in case the corresponding node is upgraded. Edge delays as well as edge upgrading costs are encoded by vectors $\mathbf{d}$ and $\mathbf{c}$, respectively.

As pointed out in [Krumke et al., 1999], this model is a generalization of the node upgrade model introduced by [Paik and Sahni, 1995], in which d_e^1 and d_e^2 where such that $d_e^1 = \alpha d_e^0$ and $d_e^2 = \alpha^2 d_e^0$, for all $e \in E$, for a given $\alpha \in (0, 1)$.

We can now present the formal definition of the two problems addressed in this paper.

The Minimum Delay Upgrading MST Problem Let $\mathcal{T}$ be the set of all spanning trees on G. Likewise, let an *upgrading scheme* S to be encoded by a subset $V_S \subseteq V$, so that all nodes in V_S are upgraded, while the remaining ones are not. An upgrading scheme S is feasible if the cost of the upgrading actions induced by S,

$$C(S) = \sum_{i \in V_S} c_i,$$

does not exceed a given *upgrading budget* $B \geq 0$. Let $\mathcal{S}$ denote the family of all upgrading schemes. Let $D(T, S)$ be the total delay of the spanning tree $T \in \mathcal{T}$ under upgrading scheme $S \in \mathcal{S}$. Using this notation, we can formulate the minimum delay upgrading MST problem (MDUMST) as

$$(T^*, S^*) = \arg\min \{D(T, S) \mid C(S) \leq B,\ S \in \mathcal{S} \text{ and } T \in \mathcal{T}\}; \tag{MDUMST}$$

in other words, we look for a pair (T^*, S^*) that minimizes total delay of the corresponding minimum spanning tree. In [Krumke et al., 1999] the authors provided an $(1, (1+\epsilon)^2 \mathcal{O}(\log|V|))$-approximation algorithm. In a bicriteria optimization context, this means that the algorithm (i) either produces a solution for which the value $C(S)$ is most $(1+\epsilon)^2 \mathcal{O}(\log|V|)$ times the specified budget B, and the value of $D(T, S)$ is the minimum value of a solution that satisfies the budget constraint; or (ii) correctly provides the information that there is no subgraph which satisfies the budget constraint $C(S) \leq B$.

The Minimum Cost Upgrading MST Problem Let $W \geq 0$ be a maximum total delay bound; we seek for a spanning tree $T \in \mathcal{T}$ and an upgrading scheme $S \in \mathcal{S}$, so that the total upgrading cost is minimized while the total delay is at most W, i.e.,

$$(T^*, S^*) = \arg\min \{C(S) \mid D(T, S) \leq W,\ S \in \mathcal{S} \text{ and } T \in \mathcal{T}\}. \tag{MCUMST}$$

Similar as for the MDUMST, an $((1+\epsilon)^2\mathcal{O}(\log|V|), 1)$-approximation algorithm was provided in [Krumke et al., 1999].

Previous Work One of the very first references to network upgrading problems based on node-upgrades can be found in [Paik and Sahni, 1995]. In that paper, five variants of network upgrading problems are studied. For these variants delays can be caused both along edges and across nodes, so the upgrade decisions involve both components. The computational complexity of the discussed problems is provided, showing that they range from polynomially solvable problems up to NP-hard problems.

In [Ibaraki et al., 2005], the authors study a problem in which the goal is to find a budget constrained node-upgrading decision so as to minimize the *eccentricity*, i.e., the largest distance from one designated node to the other nodes of a tree. Hardness results as well approximation algorithms are provided for two different node-upgrading models.

Complementary, problems where the designed network corresponds to a path have been also considered before in the literature. In [Dilkina et al., 2011] the upgrading shortest path problem (USP) is introduced. The problem is presented in the context of improving landscape connectivity within natural reserves. Note that in the USP, the shortest-path cost is based on costs of nodes in the path, which can be improved by upgrading, i.e., the node-upgrading model is different to the model considered in this work. A Mixed Integer Programming (MIP) model is provided as well as a basic greedy heuristic. Computational results on a set of randomly generated instances show that the proposed heuristic is competitive when compared to the direct resolution of the proposed MIP model in a commercial solver. Later on, in [Álvarez-Miranda et al., 2016], both a branch-and-cut (B&C) and a Benders decomposition are designed for the USP. Numerical results show that the designed algorithmic strategies outperform those proposed in [Dilkina et al., 2011] on equivalent benchmark instances.

It is interesting to note that network upgrading problems can be seen as two-player cooperative games; one of the players wants to efficiently allocate resources for upgrading the network (e.g., S) so that the performance of the decision of a second player (e.g., T) is enhanced. This differs from the *network interdiction* problems [see, e.g. Hemmecke et al., 2003, Israeli and Wood, 2002]. In these problems, one of the players aims at efficiently allocating resources in order to *downgrade* or *interdict* the network, which leads to worsening the decision of a second player.

Our Contribution and Paper Outline In this paper, we propose MIP models for both MDUMST and MCUMST and design specially tailored Lagrangian relaxations and B&C approaches to solve the proposed formulations. We also present different enhancements for the devised approaches, including valid inequalities, primal heuristics and variable fixing procedures.

The proposed algorithmic schemes are shown to be effective for solving to optimality most of the instances in the testbed, or manage to obtain solutions and bounds yielding very small optimality gaps. Besides, the proposed enhancements turn out to be beneficial for improving the performance of the algorithms. The Lagrangian relaxation approaches allow us to tackle even larger instances, whose sizes represent a burden for the B&C algorithm.

The paper is organized as follows. In Section 2 we present MIP formulations for both problems and Lagrangian approaches for providing both upper and lower bounds. Enhancements for the approaches are also discussed in the Section. Alternative MIP formulations based on a transformation of the input graph and B&C approaches to tackle them are presented in Section 3. In Section 4 we report computational results obtained when solving, with the two proposed approaches, two sets of instances; one comprised by randomly generated instances, and a second one of instances adapted from the well-knwon SteinLib dataset. Conclusions and paths for future work are discussed in Section 5.

2. Lagrangian Relaxation Approaches

For a combinatorial optimization CO problem, which can be formulated as MIP with a set of *easy* constraints $A\mathbf{x} \leq \mathbf{b}$ and *complicating* constraints $H\mathbf{x} \leq \mathbf{h}$, i.e.,

$$\text{CO}: \quad z^* = \min\left\{\mathbf{f}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b},\ H\mathbf{x} \leq \mathbf{h} \text{ and } \mathbf{x} \in \mathbb{Z}^n\right\}, \tag{CO}$$

Lagrangian relaxation is an attractive way to solve it [see, e.g., Wolsey, 1998]. *Easy* and *complicating* constraints means, that the problem CO without constraints $H\mathbf{x} \leq \mathbf{h}$ is easy to solve, e.g., using a combinatorial algorithm. Let $\boldsymbol{\lambda} \geq 0$ be a vector of *dual multipliers* for $H\mathbf{x} \leq \mathbf{h}$. Using this vector, the Lagrangian relaxation of (CO) for a given $\boldsymbol{\lambda}$ is defined as

$$z_R(\boldsymbol{\lambda}) = \min\left\{\left(\mathbf{f}^T + \boldsymbol{\lambda}^T H\right)\mathbf{x} - \boldsymbol{\lambda}^T\mathbf{h} \mid A\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \in \mathbb{Z}^n\right\}; \tag{LR}$$

this relaxation, $z_R(\boldsymbol{\lambda})$, gives a lower bound for the objective of (CO), i.e., $z^* \geq z_R(\boldsymbol{\lambda})$, and to find the best lower bound, a maximization problem in $\boldsymbol{\lambda}$, called the *Lagrangian dual problem*, is solved. We use a subgradient method to solve this maximization problem; details of it are discussed at the end of this section. Within the subgradient method, the Lagrangian relaxation (LR) gets solved for different multipliers $\boldsymbol{\lambda}$.

In the following, we provide a formulation for MDUMST, in which the Lagrangian relaxation decomposes into a *minimum spanning tree problem* and a *knapsack problem*; and a formulation of MCUMST, in which the Lagrangian relaxation decomposes into a *minimum spanning tree problem* and a problem that can be solved by inspection. Primal heuristics and variable fixing based on Lagrangian prices are also described in this section.

2.1 MIP Formulations for the MDUMST and MCUMST

Let $\mathbf{y} \in \{0,1\}^{|V|}$ be a vector of binary variables so that $y_i = 1$ if node $i \in V$ is upgraded, and $y_i = 0$ otherwise; therefore, a realization of vector $\mathbf{y}$ encodes an upgrading scheme. Complementary, let $\mathbf{x} \in \{0,1\}^{3|E|}$ be a vector binary variables so that $x_e^l = 1$ if edge $e \in E$ is taken as part of the spanning tree and its l-upgrade, $l \in \{0,1,2\}$, contributes to the total delay; $x_e^l = 0$, otherwise. The vector $\mathbf{x}$ can also be interpreted as the incidence vector of a multigraph containing three edges e^0, e^1, e^2 with associated distances d_e^0, d_e^1, d_e^2 for $e \in E$. Considering the definition presented above, variables $\mathbf{y}$ and $\mathbf{x}$ are coupled by

$$x_e^1 + 2x_e^2 \leq y_i + y_j,\ \forall e : \{i,j\} \in E, \tag{COUP}$$

which forces that if edge e is taken and its 1-upgrade (resp. 2-upgrade) contributes to the total delay, then exactly one (resp. two) of its endpoints are upgraded.

For the case of the MDUMST, the upgrading scheme must verify a budget constraint for a given budget B; in terms of $\mathbf{y}$ variables, this can be expressed as

$$\sum_{i \in V} c_i y_i \leq B. \tag{BUD}$$

Likewise, for the MCUMST, the total delay bound can be expressed as

$$\sum_{e \in E} \left(d_e^0 x_e^0 + d_e^1 x_e^1 + d_e^2 x_e^2 \right) \leq W. \tag{DEL}$$

Finally, let $SpT(\mathbf{x})$ be a generic notation for the constraint that forces the selected edges indicated by $\mathbf{x}$ to induce a spanning tree in G. Note that for the purpose of the Lagrangian approach, we do not need an explicit description of this constraint, since the resulting problem will be solved in a combinatorial way by calculation of a minimum spanning tree.

Using the elements presented above, for a given cost budget B, the MDUMST can be formulated as the following MIP model

$$\text{(MDUMST)} \qquad \min \Big\{ \mathbf{d}^T \mathbf{x} \mid \text{(COUP), (BUD)}, SpT(\mathbf{x}),\ \mathbf{x} \in \{0,1\}^{3|E|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|} \Big\};$$

similarly, for a given total delay bound W, the following MIP formulation allows to model the MCUMST,

$$\text{(MCUMST)} \qquad \min \Big\{ \mathbf{c}^T \mathbf{y} \mid \text{(COUP), (DEL)}, SpT(\mathbf{x}),\ \mathbf{x} \in \{0,1\}^{3|E|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|} \Big\}.$$

Although the formulations presented above are sufficient for a whole description of the set of feasible solutions, is possible to enhance them by including additional valid inequalities. Note that the following inequalities are valid for both the MDUMST and the MCUMST. They are based on the following result.

Theorem 1. *The subgraph induced by the edges indicated by* $\mathbf{x}^2$ *and nodes indicated by* $\mathbf{y}$ *must be a forest.*

Proof. Follows from the combination of the facts that (i) an edge $e : \{i, j\} \in E$ can only be selected at upgrade-level 2, i.e., $x_{ij}^2 = 1$, if both nodes i and j are upgraded, and that (ii) the solution indicated by $\mathbf{x}$ must be a tree. $\square$

Thus, constraints which induce this fact can be added to the presented models. This can be achieved with the following set of *generalized subtour-elimination constraints (GSECs)* [see, e.g., Goemans, 1994]. Let $S \subseteq V$ and $E(S) = \{\{i,j\} \in E \mid i \in S, j \in S\}$, the GSEC constraints are defined as

$$\sum_{e \in E(S)} x_e^2 - \sum_{k \in S} y_k \leq y_v,\ \forall v \in S,\ \forall S \subseteq V : |S| \geq 2, \tag{GSEC}$$

which in our context ensure, that for every subset S of upgraded nodes, only $|S|-1$ edges of upgrade-level 2 in $E(S)$ are selected.

In our approach, we do not use the complete family of (GSEC), since this family is of exponential size, but only the ones for $|S|=2$, i.e.,

$$x_e^2 - y_i \leq 0 \text{ and } x_e^2 - y_j \leq 0, \; \forall e = \{i,j\} \in E; \tag{GSEC-2}$$

and the one for $|S|=|V|$, i.e.,

$$\sum_{e \in E(V)} x_e^2 - \sum_{i \in V} y_i \leq 1. \tag{GSEC-V}$$

The latter is valid since in every feasible solution *at most* one upgrade must be selected allowing to replace the right-hand-side by 1.

2.2 Lagrangian Relaxation for MDUMST

To construct a Lagrangian relaxation for MDUMST, we relax the coupling constraints (COUP) and the valid inequalities (GSEC-2) and (GSEC-V); these relaxed constraints are encoded as $(H, \mathbf{h})$ in the remainder of this section. Note that due the structure of the relaxed constraints, all coefficients of $\mathbf{x}$ in H are positive and all coefficients of $\mathbf{y}$ are negative. Let $\boldsymbol{\lambda}$ be a vector of dual multipliers of appropriate size. We obtain the following Lagrangian relaxation

$$(\text{MDUMST-LR}(\boldsymbol{\lambda})) \qquad \min \Big\{ \mathbf{d}^T\mathbf{x} + \boldsymbol{\lambda}^T(H(\mathbf{x},\mathbf{y}) - \mathbf{h}) \mid (\text{BUD}), \; SpT(\mathbf{x}), \; \mathbf{x} \in \{0,1\}^{3|E|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|} \Big\};$$

since all constraints involving both $\mathbf{x}$ and $\mathbf{y}$ are relaxed in (MDUMST-LR($\boldsymbol{\lambda}$),) the problem can be decomposed in a problem only involving $\mathbf{x}$ and in another only involving $\mathbf{y}$. Let $\mathbf{d}_{\boldsymbol{\lambda}}$ be the vector of *Lagrangian prizes* for $\mathbf{x}$ for a given $\boldsymbol{\lambda}$, i.e., $\mathbf{d}_{\boldsymbol{\lambda}} = \mathbf{d} + \boldsymbol{\lambda}^T H_{\mathbf{x}}$, where $H_{\mathbf{x}}$ indicates the part of H containing the coefficients of $\mathbf{x}$. The problem in $\mathbf{x}$ reads

$$(\text{MDUMST-LR}(\boldsymbol{\lambda})\text{-}\mathbf{x}) \qquad \min \Big\{ \mathbf{d}_{\boldsymbol{\lambda}}^T\mathbf{x} \mid SpT(\mathbf{x}), \; \mathbf{x} \in \{0,1\}^{3|E|} \Big\},$$

i.e., it is a minimum spanning tree problem in a multigraph. The problem can be easily solved using a minimum spanning tree algorithm by realizing that of the three edges e^0, e^1, and e^2, $e = \{i,j\}$, only the one with minimum Lagrangian prize may occur in the optimal solution.

Complementary, let $\mathbf{c}_{\boldsymbol{\lambda}}$ be the vector of Lagrangian prizes for $\mathbf{y}$ for a given $\boldsymbol{\lambda}$, i.e., $\mathbf{c}_{\boldsymbol{\lambda}} = \boldsymbol{\lambda}^T H_{\mathbf{y}}$, where $H_{\mathbf{y}}$ indicates the part of H containing the coefficients of $\mathbf{y}$. Recall that all coefficients in $H_{\mathbf{y}}$ are negative due to the structure of the relaxed constraints. The problem in $\mathbf{y}$ reads

$$(\text{MDUMST-LR}(\boldsymbol{\lambda})\text{-}\mathbf{y}) \qquad \min \Big\{ \mathbf{c}_{\boldsymbol{\lambda}}^T\mathbf{y} \mid (\text{BUD}), \; \mathbf{y} \in \{0,1\}^{|V|} \Big\};$$

Since all coefficients in $H_{\mathbf{y}}$ are negative and $\boldsymbol{\lambda} \geq 0$, we have that $\mathbf{c}_{\boldsymbol{\lambda}} \leq \mathbf{0}$. Consequently, (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{y}$) is just the 0/1-knapsack problem (stated in minimization form) and we solve it by dynamic programming [see Martello and Toth, 1990].

Constructing Primal Solutions Let $\mathbf{y}^*$ be the optimal solution to (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{y}$) for a given $\boldsymbol{\lambda}$. We use $\mathbf{y}^*$ to construct feasible solutions to MDUMST as follows.

1. Take $\mathbf{y}^*$ as upgrading scheme. Note that $\mathbf{y}^*$ is a feasible upgrading scheme, since it fulfills (BUD).
2. Compute a minimum spanning tree using the delays induced by $\mathbf{y}^*$.

Variable Fixing We use variable fixing based on Lagrangian prizes to reduce the size of the problem during the course of the algorithm [see, e.g. Wolsey, 1998]. Let $(\mathbf{x}^-, \mathbf{y}^-)$ be the variables with value zero in the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to (MDUMST-LR($\boldsymbol{\lambda}$)) for a given $\boldsymbol{\lambda}$. Each binary variable x_e^{l-} and y_i^- is tentatively set to one, and the Lagrangian relaxation is resolved. If the obtained bound is larger than the objective value, say z^I, of the current incumbent, the variable cannot be one in the optimal solution and thus can be fixed to zero for the remainder of the algorithm. For the given $\boldsymbol{\lambda}$, let $z_{\boldsymbol{\lambda}}(\mathbf{x}^*)$ be the optimal solution value of (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{x}$) and $z_{\boldsymbol{\lambda}}(\mathbf{y}^*)$ be the optimal solution value of (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{y}$). The optimal solution value $z_{\boldsymbol{\lambda}}(\mathbf{x}^*, \mathbf{y}^*)$ for (MDUMST-LR($\boldsymbol{\lambda}$)) is then $z_{\boldsymbol{\lambda}}(\mathbf{x}^*) + z_{\boldsymbol{\lambda}}(\mathbf{y}^*) - \boldsymbol{\lambda}^T\mathbf{h}$. To speed-up the variable fixing routine, we resolve a relaxation of the Lagrangian relaxation and again use the decomposability of the problem. Since relaxations give an under-estimation of the value obtained by an exact method, the procedure remains valid.

- Resolving for a x_e^{l-} fixed to one: Clearly, only the optimal value of (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{x}$) is influenced by the variable fixing. This means, we would need to compute the minimum spanning tree containing edge e^l. Let $d^l_{\boldsymbol{\lambda}e}$ be the Lagrangian prize of the edge e^l and $d^*_{\boldsymbol{\lambda}}$ the largest value of $d^l_{\boldsymbol{\lambda}f}$ amongst all edges f^l with $x_f^{*l} = 1$. It is easy to see that $z_{\boldsymbol{\lambda}}(\mathbf{x}^*) + d^l_{\boldsymbol{\lambda}e} - d^*_{\boldsymbol{\lambda}}$ is a lower bound on the cost of the minimum spanning tree containing e^l. Hence, edge x_e^{l-} can be fixed to zero, if $z_{\boldsymbol{\lambda}}(\mathbf{x}^*) + d^l_{\boldsymbol{\lambda}e} - d^*_{\boldsymbol{\lambda}} + z_{\boldsymbol{\lambda}}(\mathbf{y}^*) - \boldsymbol{\lambda}^T h > z^I$. Note that similar approaches have been used for other constrained tree problems [see, e.g., Salles da Cunha et al., 2009, Savelsbergh and Volgenant, 1985].
- Resolving for a y_i^- fixed to one: Clearly, only the optimal value of (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{y}$) is influenced by the variable fixing. We compute the value of the Linear Programming (LP)-relaxation of the knapsack-problem with the fixed variable. This value is also known as Dantzig-bound [see, e.g., Martello and Toth, 1990] and can be obtained by sorting all items with $c_{\boldsymbol{\lambda}i} < 0$ in ascending according to the ratio $c_{\boldsymbol{\lambda}i}/c_i$, $i \in V$ (recall that the problem is stated in minimization form in our case). Items are picked until the budget B is reached; it can happen that only a fraction of the last item can be picked. Let $\bar{\mathbf{y}}$ be the obtained solution; the variable y_i^- can be fixed to zero, if $z_{\boldsymbol{\lambda}}(\mathbf{x}^*) + z_{\boldsymbol{\lambda}}(\bar{\mathbf{y}}) - \boldsymbol{\lambda}^T h > z^I$.

2.3 Lagrangian Relaxation for MCUMST

In the case of the MCUMST, besides relaxing the coupling constraints and the added valid inequalities, we also relax the total delay constraint (DEL). We again use $(H, \mathbf{h})$ to encode the relaxed constraints, and since (DEL) contains only positive coefficients of $\mathbf{x}$, it still holds that all coefficients of $\mathbf{x}$ in H are positive, while all coefficients of $\mathbf{y}$ are negative. For an appropriately sized vector $\boldsymbol{\lambda}$

of dual multipliers, we obtain the following Lagrangian relaxation for MCUMST

$$\text{(MCUMST-LR}(\boldsymbol{\lambda})) \qquad \min\left\{\mathbf{c}^T\mathbf{y} + \boldsymbol{\lambda}^T(H(\mathbf{x},\mathbf{y}) - \mathbf{h}) \mid SpT(\mathbf{x}), \right. \\ \left. \mathbf{x} \in \{0,1\}^{3|E|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|}\right\};$$

the problem (MDUMST-LR($\boldsymbol{\lambda}$)) also decomposes into a problem in $\mathbf{x}$ and another in $\mathbf{y}$. Let $\mathbf{d}_{\boldsymbol{\lambda}}^T$ be the vector of Lagrangian prizes for $\mathbf{x}$ for a given $\boldsymbol{\lambda}$, i.e., $\mathbf{d}_{\boldsymbol{\lambda}}^T = \boldsymbol{\lambda}^T H_x$. The problem in $\mathbf{x}$ reads

$$\text{(MCUMST-LR}(\boldsymbol{\lambda})\text{-}\mathbf{x}) \qquad \min\left\{\mathbf{d}_{\boldsymbol{\lambda}}^T\mathbf{x} \mid SpT(\mathbf{x}),\ \mathbf{x} \in \{0,1\}^{3|E|}\right\};$$

i.e., it is similar to (MDUMST-LR($\boldsymbol{\lambda}$)$-\mathbf{x}$), with the only difference being the definition of the Lagrangian prizes (recall that for MCUMST, they also contained $\mathbf{d}$).

Let $\mathbf{c}_{\boldsymbol{\lambda}}$ be the vector of Lagrangian prizes for $\mathbf{y}$ for a given $\boldsymbol{\lambda}$, i.e., $\mathbf{c}_{\boldsymbol{\lambda}} = \mathbf{c} - \boldsymbol{\lambda}^T H_{\mathbf{y}}$. Note that due to the presence of $\mathbf{c}$, the Lagrangian prizes can be positive, zero, or negative. The problem in $\mathbf{y}$ reads

$$\text{(MCUMST-LR}(\boldsymbol{\lambda})\text{-}\mathbf{y}) \qquad \min\left\{\mathbf{c}_{\boldsymbol{\lambda}}^T\mathbf{y} \mid \mathbf{y} \in \{0,1\}^{|V|}\right\};$$

this problem can be solved by inspection, i.e., by setting all variables y_i with $c_{\boldsymbol{\lambda}i} < 0$ to one, and the remaining ones to zero.

Constructing Primal Solutions We use the Lagrangian prizes $\mathbf{c}_{\boldsymbol{\lambda}}$ for variables $\mathbf{y}$ to guide our search for feasible solutions by using the following primal heuristic:

1. Sort the nodes $i \in V$ in an ascending order using $\mathbf{c}_{\boldsymbol{\lambda}i}$, save this ordering in list L.
2. Set $k = \lceil |V|/2 \rceil$, $k_l = 0, k_u = |V|$.
3. Construct an upgrading scheme $\mathbf{y}^*$ by upgrading nodes 1 to k on list L.
4. Compute a minimum spanning tree using the delays induced by $\mathbf{y}^*$.
5. If the solution is feasible (i.e., the delay of the spanning tree is at most W), set $k_u = k$, $k = \lceil (k + k_l)/2 \rceil$ and update the incumbent if the solution improves it; otherwise set $k_l = k$, $k = \lceil (k + k_u)/2 \rceil$. If k would not change by this new setting of k, then Stop; otherwise go back to step 3. In other words, we do a binary search to determine the best k.

The above scheme is speeded-up using the lower bound z_{LB} and upper bound z^I; in other words, if an upgrading scheme $\mathbf{y}^*$ would cost less than z_{LB} or more than z^I we do not need to compute the spanning tree, since the produced solution cannot be feasible in the first case, and in the second case, it would not improve the incumbent solution value.

Variable Fixing We also use variable fixing for MCUMST by exploiting the decomposability of the problem. The procedure for variables in $\mathbf{x}^-$ is exactly the same as for MDUMST (as the problems (MCUMST-LR($\boldsymbol{\lambda}$)-$\mathbf{x}$) and (MDUMST-LR($\boldsymbol{\lambda}$)-$\mathbf{x}$) are the same). The procedure for variables in $\mathbf{y}^-$ is simpler than for MDUMST, since the problem (MCUMST-LR($\boldsymbol{\lambda}$)-$\mathbf{y}$) can be solved by inspection, in contrast to the knapsack problem for MDUMST: sSetting a variable y_i^- to one does not change the rest of the solution $(\mathbf{x}^*, \mathbf{y}^*)$, hence the optimal value after resolving is $z_{\boldsymbol{\lambda}}(\mathbf{x}^*) + z_{\boldsymbol{\lambda}}(\mathbf{y}^*) + c_{\boldsymbol{\lambda}i} - \boldsymbol{\lambda}^T\mathbf{h}$. If this value is larger than z^I, y_i can be fixed to zero.

2.4 Details of the Subgradient Method

We use the subgradient method [see, e.g., Wolsey, 1998] to solve the Lagrangian dual problem

$$\max_{\boldsymbol{\lambda}\geq 0} z_R(\boldsymbol{\lambda}) \tag{LD}$$

where $z_R(\boldsymbol{\lambda})$ is (MDUMST-LR($\boldsymbol{\lambda}$)) or (MCUMST-LR($\boldsymbol{\lambda}$)) resp., in order to find the best possible lower bound. In the subgradient method, $z_R(\boldsymbol{\lambda})$ gets iteratively resolved for different values of dual multipliers $\boldsymbol{\lambda}$. The value of $\boldsymbol{\lambda}^{t+1}$, at iteration $t+1$, obtained from the current solution $(\mathbf{x}^t, \mathbf{y}^t)$ with the help of a subgradient $\mathbf{g}^t$. Such subgradient is calculated as $\mathbf{g}^t = \mathbf{h} - H(\mathbf{x}^t, \mathbf{y}^t)$. In practice, it often leads to more efficient approaches to use a search direction $\mathbf{s}^t$ calculated with the help of the subgradient to update $\boldsymbol{\lambda}^{t+1}$, instead of just using $\mathbf{g}^t$. In this paper, we use the average direction search strategy [see Haouari et al., 2008, Sherali and Ulular, 1990], which is defined as

$$\mathbf{s}^t = \mathbf{g}^t + \frac{\|\mathbf{g}^t\|}{\|\mathbf{s}^{t-1}\|}\mathbf{s}^{t-1}.$$

The last ingredient necessary to derive $\boldsymbol{\lambda}^{t+1}$ is a step size μ^t, which is calculated as

$$\mu^t = \beta\frac{z^I - z_R\left(\boldsymbol{\lambda}^t\right)}{\|\mathbf{s}^t\|}$$

where β is a given parameter in $(0,2]$. The dual multipliers $\boldsymbol{\lambda}^{t+1}$ are then calculated as $max\{0, \boldsymbol{\lambda}^t - \mu^t\mathbf{s}^t\}$.

In our implementation, we initialize with $s^0 = 0$, and $\beta = 2$. If there are $\gamma = 20$ iterations without improvement of the lower bound z_{LB}, we set $\beta = \beta/2$ and also set $(\mathbf{x}^t, \mathbf{y}^t)$ to the solution that has provided the best lower bound so far; afterwards, we recalculate the subgradient and the search direction [a similar strategy has been used in Haouari et al., 2008]. We call the variable fixing routine and primal heuristic at every iteration. The maximum number of iterations in the subgradient method is set to 400. Finally, it proved computationally advantageous to initialize the entries of $\boldsymbol{\lambda}^0$ with 10^{-5}.

3. Directed MIP Formulation and Branch-and-Cut Algorithm

In this section we propose MIP formulations to solve MDUMST and MCUMST using B&C. The formulations are extended formulations, which are defined on a graph obtained from bi-directing G. This is done, since for tree-based problems, such *directed* formulations are usually more efficient than formulations based on the original undirected graph G [see, e.g., Ljubić et al., 2006, Salles da Cunha et al., 2009].

In particular, an instance is transformed as follows: let $G_A = (V, A)$ so that $A = \{(i,j),\ (j,i) \mid \forall e : \{i,j\} \in E\}$, i.e., each edge is replaced by two directed arcs (one in each direction). Additionally, the delay values are transformed as follows: $d^0_{ij} = d^0_{ji} = d^0_e$, $d^1_{ij} = d^1_{ji} = d^1_e$ and $d^2_{ij} = d^2_{ji} = d^2_e$ for all $e : \{i,j\} \in E$. With a slight abuse of notation, the vector of delay values after the transformation is given the same name as its undirected counterpart, i.e., $\mathbf{d}$.

Let $\mathbf{z} \in \{0,1\}^{3|A|}$ be a vector of binary variables so that $z^l_{ij} = 1$ if arc $(i,j) \in A$ is taken in the spanning arborescence and its l-upgrade, $l \in \{0,1,2\}$, contributes to the total delay; $z^l_{ij} = 0$, otherwise. For a given set of nodes $S \subseteq V$, let $\delta^-(S) = \{(i,j) \in A \mid i \in V \setminus S,\ j \in S\}$ (resp. $\delta^+(S) = \{(i,j) \in A \mid i \in S,\ j \in V \setminus S\}$), i.e., the set of incoming (resp. outgoing) arcs of a given subset of nodes $S \subseteq V$.

The counterpart of coupling constraints (COUP) is expressed in the $(\mathbf{z}, \mathbf{y})$-variables by

$$z^1_{ij} + 2z^2_{ij} \le y_i + y_j,\ \forall (i,j) \in A; \tag{$\mathbf{z}$-COUP}$$

since for any edge $\{i,j\}$, only one arc may be in the solution, these constraints can be replaced by the following stronger ones

$$z^1_{ij} + z^1_{ji} + 2z^2_{ij} + 2z^2_{ji} \le y_i + y_j,\ \forall \{i,j\} \in E. \tag{$\mathbf{z}$-COUP+}$$

The total delay bound can be expressed, in the $\mathbf{z}$-space, as

$$\sum_{(i,j)\in A} \left(d^0_{ij} z^0_{ij} + d^1_{ij} z^1_{ij} + d^2_{ij} z^2_{ij} \right) \le W. \tag{$\mathbf{z}$-DEL}$$

Note that in this directed graph, instead of looking for a spanning tree we actually look for a spanning arborescence on G rooted at some chosen node 1. Such topology can be imposed by the following connectivity constraints,

$$\sum_{(i,j)\in\delta^-(S)} \left(z^0_{ij} + z^1_{ij} + z^2_{ij} \right) \ge 1,\ \forall S \subseteq V \setminus \{1\}; \tag{$\mathbf{z}$-CONN}$$

these are the so-called *directed cut-set inequalities* [see, e.g., Koch and Martin, 1998, Ljubić et al., 2006], and they impose that in the subgraph induced by $\mathbf{z}$ there exists a directed path from 1 to each of the other nodes in the graph. Since there is an exponential number of constraints ($\mathbf{z}$-CONN), we do not add them at the start, but separate them on-the-fly within the B&C. The separation problem for ($\mathbf{z}$-CONN) is a *max-flow problem* and is discussed in more detail later in this section. Note that for $S = \{j\}$ the inequality sign in ($\mathbf{z}$-CONN) can be replaced by equality.

We can now provide the following MIP formulations in the $(\mathbf{z}, \mathbf{y})$-space for the studied problems. For a given budget level B, the MDUMST can formulated as

$$\text{(MDUMST-MIP)} \qquad \min \left\{ \mathbf{d}^T\mathbf{z} \,\middle|\, (\mathbf{z}\text{-COUP+}),\ (\text{BUD}),\ (\mathbf{z}\text{-CONN}),\ \mathbf{z} \in \{0,1\}^{3|A|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|} \right\};$$

likewise, for a given total delay bound W, the MCUMST can be modeled as

$$\text{(MCUMST-MIP)} \qquad \min \left\{ \mathbf{c}^T\mathbf{y} \,\middle|\, (\mathbf{z}\text{-COUP+}),\ (\mathbf{z}\text{-DEL}),\ (\mathbf{z}\text{-CONN})\ \ \mathbf{z} \in \{0,1\}^{3|A|} \text{ and } \mathbf{y} \in \{0,1\}^{|V|} \right\}.$$

Additionally, we can define valid inequalities for both (MDUMST-MIP) and (MCUMST-MIP) similar to the ones used in Lagrange relaxation. In particular, the counterpart of constraints (GSEC-2) is given by

$$z^2_{ij} + z^2_{ji} \le y_j,\ \forall (i,j) \in A; \tag{$\mathbf{z}$-GSEC-2}$$

complementary, the counterpart of constraint (GSEC-V) is given by the following set of constraints

$$\sum_{(i,j)\in\delta^-(\{j\})} z^2_{ij} \leq y_j,\ \forall j \in V \setminus \{1\}. \qquad (\mathbf{z}\text{-GSEC-}j)$$

Separation of Connectivity Constraints (**z**-CONN) Let $(\tilde{\mathbf{z}}, \tilde{\mathbf{y}})$ be the solution of the LP-relaxation of the MDUMST or the MCUMST at a given node of the branch-and-bound tree. The separation works by calculating a maximum flow from $1 \in V$ to each of the other nodes $k \in V \setminus \{1\}$ in G, with capacities of arcs (i,j) set to $\tilde{\kappa}_{ij} = \tilde{z}^0_{ij} + \tilde{z}^1_{ij} + \tilde{z}^2_{ij}$. If the maximum flow between 1 and k has a value $\epsilon < 1$, the corresponding cut-set $\delta^-(S)$ induces violated connectivity cut (**z**-CONN). Note that similar separation routines are used in many constrained tree problems [see, e.g., Koch and Martin, 1998, Ljubić et al., 2006].

We enhanced the separation routine by modifying it to produce *orthogonal* cuts [see, e.g. Lucena and Resende, 2004]. The modification works by setting $\tilde{z}^0_{ij} = \tilde{z}^1_{ij} = \tilde{z}^2_{ij} = 1$ once arc (i,j) appeared in a cut-set $\delta^-(S)$ in the separation for some k. This way, the arc cannot appear in the cut-set when applying the separation routine for node $k' \neq k$. This modification turned out to be very helpful for performance; otherwise, many similar cuts get added, leading to very large LPs that need to get resolved.

Additionally, we added an early termination criterion to the cut-loop in order to avoid adding cuts of type (**z**-CONN) that do not improve the lower bound (but still increase the size of the current LP model). If for three rounds of the separation the lower bound at a given node moves less than 10^{-3} units, the cut-loop is aborted.

Note that for the correctness of the B&C approach, it is enough to call the separation routine only for integer solutions. In such case, we do not need to use a max-flow algorithm; we simply build the connected components of G induced by integer $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ and construct constraints (**z**-CONN) based on the connected components.

Branching Priorities Due to the coupling constraints (COUP) (as well as the valid inequalities (**z**-GSEC-2), and (**z**-GSEC-j)), setting an upgrade variable, say y_i, to 0 has immediate implications on the value of the **z** variables of its adjacent arcs. If $y_i = 0$ then $z^2_{ij} = 0$ for all $(i,j) \in \delta^+(i)$ and $(j,i) \in \delta^-(i)$, and if $y_i = y_j = 0$ for $(i,j) \in A$, $z^1_{ij} = z^1_{ji} = z^2_{ij} = z^2_{ji} = 0$. Hence, giving higher branching priorities to the **y** variables could be helpful to better reduce the search space in earlier stages of the branch-and-bound tree, boosting the efficacy of the algorithm. The effect of this setting is analyzed in our computational study.

Constructing Primal Solutions We use primal heuristics within the B&C framework to produce feasible solutions. These heuristics are modifications of the primal heuristics used in the Lagrangian relaxation approach and are driven by the values $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ of the LP-relaxations. In particular, the heuristics are adapted as follows:

- MDUMST: The upgrading scheme $\mathbf{y}^*$ in Step 1 is constructed by sorting $\tilde{\mathbf{y}}$ in an descending order and then picking nodes until B is reached.
- MCUMST: Nodes are sorted in descending order according to $\tilde{\mathbf{y}}$ in Step 1.

4. Computational Study

The proposed algorithms were implemented in C++. Runs were carried out on an Intel Xeon CPU with 2.5 GHz and 12GB memory. CPLEX 12.6.2 was used as MIP-solver for the B&C algorithm, all parameters (except branching priorities, see Subsection 3) were left at the default values and a time limit of 1800 seconds was set.

We compare four different configurations of our B&C algorithm: **i**) *basic*, which just consists of giving to CPLEX the model (with ($\mathbf{z}$-CONN) for $S = \{j\}, \forall j \in V$, and without valid inequalities), and separating the connectivity cuts ($\mathbf{z}$-CONN); **ii**) V, which additionally uses the valid inequalities ($\mathbf{z}$-GSEC-2), ($\mathbf{z}$-GSEC-j) (added at initialization); **iii**) VH, which adds the primal heuristic to V; and **iv**) VHP, which adds branching priorities to VH.

4.1 Benchmark Instances

We considered two sets of instances for our computational study: **i**) `EUCLIDEAN`, where the underlying graphs are complete random Euclidean graphs. Euclidean graphs are known to resemble topologies encountered in real-life telecommunication networks [see, e.g., Johnson et al., 2000]; and **ii**) `C`, which are based on the instance set `C` of the well-known SteinLIB instance library [see Koch et al., 2001].

To construct set `EUCLIDEAN`, $|V|$ points were randomly selected in a 100×100 plane. The delay values d^2_{ij} are set to the Euclidean distance between i and j rounded up to the next integer, for all $i, j \in V$. The delay values d^0_{ij} and d^1_{ij} are obtained as $d^1_{ij} = \lceil \alpha^1_{ij} d^2_{ij} \rceil$ and $d^0_{ij} = \lceil \alpha^0_{ij} d^1_{ij} \rceil$, for all $i, j \in V$. The values α^1_{ij}, α^0_{ij} are chosen randomly from $[1.1, 1.3]$. The upgrade costs c_v are randomly chosen integer from the range $[1, 10]$ for every $v \in V$. We created ten instances for each $|V| \in \{100, 150, 200, 500\}$; they are labeled as $e|V|{-}k$, with $k \in \{1, \dots, 10\}$. Since these instances are complete graphs, the number of edges $|E|$ is 4950, 11175, 19900 and 124750, respectively. This generation scheme has been chosen since in practical applications, some correlation between of the original delay and the upgraded delay can be expected, and upgrade actions are likely to have comparable cost.

To transform the Steiner tree instances from `C`, we proceeded in a similar way. The original edge costs of an instance, which is a random integer in $[1, 10]$ is used as d^2, and d^1, d^0 and c are constructed as for `EUCLIDEAN`. There are 20 instances in this set, and all have $|V| = 500$. The instances are named $c01 - c20$, and have different levels of sparsity, namely $|E| \in \{625, 1000, 2500, 12500\}$.

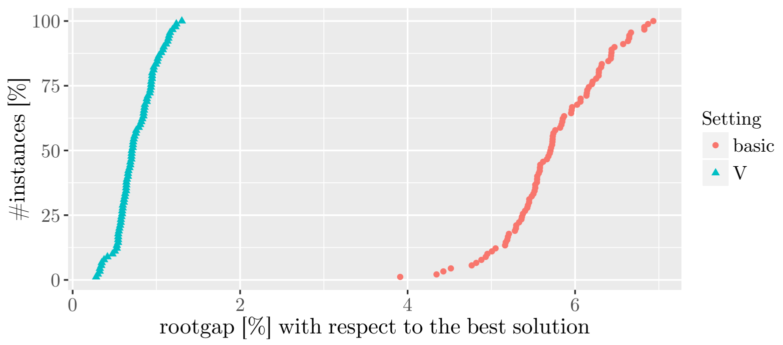
For every instance, we tested three different values for B, resp. W. For an instance, let $C = \sum_{i \in V} c_i$ and D be the delay value of a minimum spanning tree with no upgrade. The values for B are $b \cdot C$ and for W are $(b + 1) \cdot D$, with $b \in \{0.1, 0.2, 0.3\}$.

4.2 Results for MDUMST

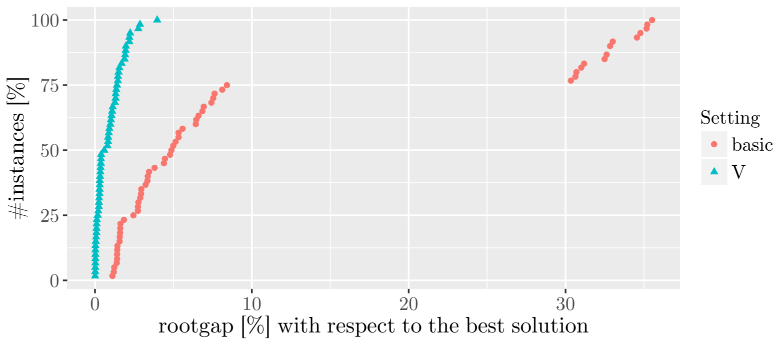
We first give a comparison of the different configurations for B&C, to establish which configuration performs best and then analyze the results produced by the Lagrangian approach and the best B&C setting in more detail.

B&C Performance Figures 2(a) and 2(b) give a comparison of the *root* gaps for configurations *basic* and V on sets `EUCLIDEAN` ($|V| \leq 200$), and `C`, respectively. The rootgap of an instance is calculated as $100 \cdot (z^* - RB)/z^*$, where z^* is the best solution found for the instance (using any configuration) and RB is the root bound obtained by the configuration. The importance of using

Figure 1: Rootgaps with respect to the best solution the best solution found by all settings for problem MDUMST on instance sets EUCLIDEAN and C.



(a) Set EUCLIDEAN with $|V| \in \{100, 150, 200\}$



(b) Set C

the valid inequalities can easily be seen; the rootgap for EUCLIDEAN is reduced by around 4%, and for C by up to 30%. The different densities of the instances of set C seems to influence the gap, especially when not using the valid inequalities. Adding the valid inequalities leads to maximal gaps under 2% for EUCLIDEAN, and under 5% for C.

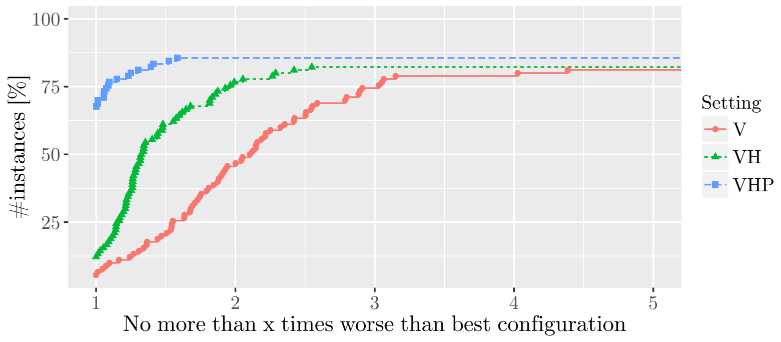
Next, we compare *V*, *VH* and *VHP* using performance profiles [Dolan and Moré, 2002]. Let $\mathcal{I}$ indicate the set of instances, $\mathcal{C} = \{V, VH, VHP\}$ and $t_{i,c}$ is the time that configuration c needs to solve instance i to optimality. The *performance ratio* $r_{i,c}$ for configuration c and instance i is defined as

$$r_{i,c} = \begin{cases} \frac{t_{i,c}}{\min_{c' \in \mathcal{C}}\{t_{i,c'}\}} & \text{if } c \text{ solves } i \\ r_M & \text{otherwise} \end{cases},$$

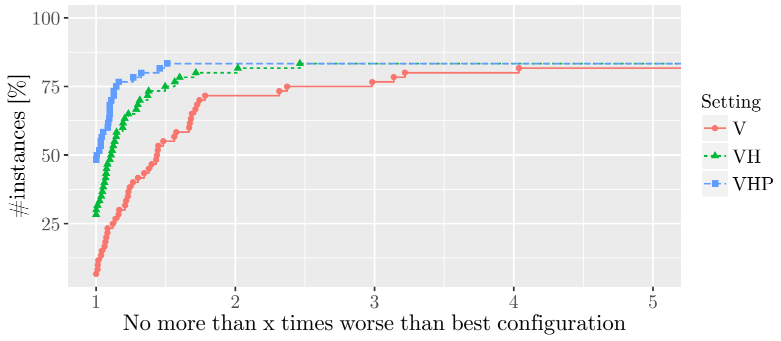
where $r_M = \max_{i \in \mathcal{I}, c \in \mathcal{C}: c \text{ solves } i} r_{i,c} + 1$. Performance profiles are plots of the cumulative distribution functions ρ_c of the performance ratio for each configuration $c \in \mathcal{C}$. The value of $\rho_c(1)$ gives how many times configuration c is the fastest and the value of $\rho_c(r_M - 1)$ indicates the percentage of instances that configuration c can solve within the time limit.

We do not include *basic* in these profiles, since no instance could be solved within the time limit

Figure 2: Performance profile of runtimes to optimality for problem MDUMST on instance sets `EUCLIDEAN` and `C`.



(a) Set `EUCLIDEAN` with $|V| \in \{100, 150, 200\}$



(b) Set `C`

using this setting. Configuration *VHP* is the winner in about 70% of the instances of set `EUCLIDEAN`, and in around 50% of instances of set `C`. For both set of instances, *VHP* dominates the other two configurations, and *VH* in turn dominates *V*. Moreover, for both sets, almost 80% of instances were be solved within the time limit.

Lagrangian Bounds v/s *VHP* We now compare the performance of the Lagrangian approach with the performance of *VHP* by reporting the results obtained by both approaches on the considered instances.

Table 1 reports detailed results of the Lagrangian approach on instance set `EUCLIDEAN`. We provide the value of the best solution (z^*), the attained lower bound (LB), the optimality gap ($g[\%]$, which is calculated as $100 \cdot (z^* - LB)/z^*$), and the runtime in seconds ($t[s]$). The largest gap is 4.06% and occurs for $e100 - 9$ with $b = 0.2$. Most of the gaps are under 2%, although instances of size $|V| = 500$ have slightly larger gaps (around 3%). A higher value of b seems to lead to larger gaps, this is more clear from $b = 0.1$ to $b = 0.2$, than from $b = 0.2$ to $b = 0.3$. The runtime for all instance with $|V| \leq 200$ is at most three seconds. For instances with $|V| > 200$ the runtime makes a jump to around 30 seconds; the largest runtime is 42 seconds for instance $e500 - 9$ with $b = 2$.

Table 2, equivalent to Table 1, shows results for the instance set `C`. Here the largest gap is

Table 1: Results of the Lagrangian approach on instance set `EUCLIDEAN` (MDUMST). z^* gives the value of the best solution found, LB the lower bound, $g[\%]$ the gap and $t[s]$ the runtime.

	b=0.1				b=0.2				b=0.3			
instance	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$
e100-1	974	967.36	0.68	1	910	901.16	0.97	1	873	861.11	1.36	0
e100-2	1019	1010.48	0.84	1	962	945.15	1.75	1	918	897.58	2.22	1
e100-3	974	964.14	1.01	1	909	897.22	1.30	1	863	848.32	1.70	0
e100-4	999	986.87	1.21	1	928	912.05	1.72	1	884	863.51	2.32	1
e100-5	1026	1004.37	2.11	1	955	929.23	2.70	1	904	877.93	2.88	0
e100-6	1020	1010.08	0.97	1	961	940.17	2.17	0	915	889.46	2.79	0
e100-7	1028	1016.42	1.13	1	962	946.50	1.61	1	914	895.51	2.02	1
e100-8	1020	999.92	1.97	0	949	927.78	2.24	0	900	877.77	2.47	0
e100-9	955	949.09	0.62	0	900	863.47	4.06	1	856	840.97	1.76	0
e100-10	1020	1008.93	1.08	1	959	938.73	2.11	1	916	889.50	2.89	1
e150-1	1165	1147.10	1.54	1	1093	1071.42	1.97	1	1035	1018.64	1.58	1
e150-2	1238	1220.02	1.45	2	1161	1131.13	2.57	1	1101	1070.76	2.75	1
e150-3	1217	1200.93	1.32	1	1135	1114.11	1.84	2	1075	1054.51	1.91	2
e150-4	1231	1215.02	1.30	1	1140	1124.55	1.36	1	1083	1063.77	1.78	1
e150-5	1260	1242.76	1.37	2	1177	1150.21	2.28	1	1115	1088.49	2.38	1
e150-6	1232	1213.97	1.46	1	1153	1125.94	2.35	1	1092	1063.71	2.59	2
e150-7	1247	1216.79	2.42	2	1163	1134.08	2.49	1	1101	1073.68	2.48	1
e150-8	1209	1193.48	1.28	2	1121	1107.40	1.21	1	1072	1047.54	2.28	2
e150-9	1248	1229.99	1.44	2	1166	1140.85	2.16	1	1103	1078.65	2.21	2
e150-10	1214	1202.34	0.96	1	1137	1114.97	1.94	2	1079	1053.00	2.41	2
e200-1	1402	1381.05	1.49	3	1308	1275.41	2.49	3	1236	1204.86	2.52	3
e200-2	1447	1430.38	1.15	2	1354	1323.30	2.27	3	1277	1242.82	2.68	2
e200-3	1444	1415.85	1.95	2	1343	1306.39	2.73	4	1274	1231.68	3.32	2
e200-4	1477	1455.81	1.43	2	1387	1353.67	2.40	2	1318	1282.84	2.67	3
e200-5	1471	1445.62	1.73	2	1374	1335.08	2.83	3	1296	1260.11	2.77	2
e200-6	1483	1462.92	1.35	2	1383	1348.77	2.47	2	1313	1269.98	3.28	3
e200-7	1466	1439.89	1.78	2	1365	1334.70	2.22	3	1296	1262.27	2.60	3
e200-8	1413	1391.03	1.55	3	1319	1286.73	2.45	3	1244	1215.39	2.30	3
e200-9	1486	1463.08	1.54	3	1392	1353.72	2.75	3	1320	1277.75	3.20	3
e200-10	1470	1451.38	1.27	3	1379	1345.19	2.45	3	1314	1272.62	3.15	3
e500-1	2512	2463.91	1.91	29	2324	2259.35	2.78	24	2190	2121.87	3.11	25
e500-2	2466	2427.65	1.56	24	2287	2218.40	3.00	25	2148	2080.17	3.16	26
e500-3	2568	2516.17	2.02	23	2370	2298.46	3.02	23	2228	2152.83	3.37	34
e500-4	2439	2394.87	1.81	23	2254	2203.13	2.26	27	2132	2067.84	3.01	28
e500-5	2424	2378.39	1.88	23	2244	2179.81	2.86	26	2107	2048.99	2.75	31
e500-6	2484	2437.22	1.88	41	2295	2239.69	2.41	27	2170	2106.12	2.94	24
e500-7	2455	2398.62	2.30	26	2259	2206.06	2.34	31	2127	2068.02	2.77	28
e500-8	2526	2464.58	2.43	29	2314	2251.06	2.72	25	2170	2110.99	2.72	27
e500-9	2473	2429.14	1.77	22	2296	2236.57	2.59	42	2164	2098.71	3.02	28
e500-10	2506	2456.80	1.96	27	2324	2253.72	3.02	30	2192	2129.14	2.87	33

31.98% (for instance $c12$ with $b = 0.3$), and there three additional instances with gaps over 10%. Nonetheless, most of the gaps are between 2% and 4%. The sparsity of the instances seems to influence the difficulty of the problem, with instances $c01$ to $c10$ ($|E| = 625$ and 1000) producing smaller gaps than $c11 - 20$ ($|E| = 2500$ and 12500); the four instances with gap over 10% also fall into this latter category. The longest runtime is seven seconds and occurs for $c16$ with $b = 0.3$. The value of b seems to have a larger influence on the runtime than the sparsity of the instances.

Table 3 reports the results on instance set `EUCLIDEAN` using *VHP*. In the table we report the value of the best found solution (z^*, in *italics* if optimality could not be proven within the time limit), the runtime ($t[s]$, with **TL** indicating that the time limit is reached), the rootgap ($g[\%]$), and the number of branch-and-bound nodes (*nodes*). All instances with $|V| \leq 150$ are solved to optimality for all values of b. For $|V| = 200$, about half of the instances can be solved within the time limit, amongst them all with $b = 0.1$. For $|V| = 500$, all instances remain unsolved. The deteriorating of the performance with the size of the instance is not surprising, as the size of the

Table 2: Results of the Lagrangian approach on instance set C (MDUMST). z^* gives the value of the best solution found, LB the lower bound, $g[\%]$ the gap and $t[s]$ the runtime.

	b=0.1				b=0.2				b=0.3			
instance	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$
c01	3458	3398.64	1.72	1	3175	3150.06	0.79	2	3025	2915.66	3.61	3
c02	3311	3262.19	1.47	1	3050	3014.70	1.16	3	2871	2846.89	0.84	4
c03	3326	3285.63	1.21	1	3047	3022.86	0.79	2	2903	2818.89	2.90	4
c04	3363	3324.89	1.13	1	3105	3070.19	1.12	3	2923	2901.62	0.73	3
c05	3341	3301.21	1.19	0	3086	3052.48	1.09	2	2903	2879.79	0.80	4
c06	2564	2504.57	2.32	1	2346	2301.03	1.92	3	2211	2153.07	2.62	4
c07	2562	2504.90	2.23	0	2352	2304.53	2.02	2	2219	2160.66	2.63	4
c08	2496	2432.91	2.53	1	2280	2231.34	2.13	2	2146	2075.24	3.30	4
c09	2443	2393.15	2.04	1	2234	2190.15	1.96	3	2084	2054.98	1.39	3
c10	2494	2438.94	2.21	1	2295	2233.30	2.69	2	2129	2103.34	1.21	4
c11	1533	1482.63	3.29	1	1396	1206.66	13.56	3	1246	1219.14	2.16	6
c12	1587	1534.84	3.29	1	1416	1367.16	3.45	2	1317	895.87	31.98	4
c13	1548	1508.61	2.54	1	1402	1330.33	5.11	3	1260	1232.35	2.19	5
c14	1563	1480.50	5.28	1	1418	1320.82	6.85	2	1260	1226.31	2.67	4
c15	1562	1455.54	6.82	0	1414	1307.05	7.56	3	1295	1093.69	15.54	4
c16	978	878.15	10.21	2	832	798.12	4.07	3	764	738.55	3.33	7
c17	971	888.67	8.48	2	831	790.19	4.91	5	755	735.72	2.55	6
c18	964	901.47	6.49	2	834	798.18	4.29	4	763	736.99	3.41	6
c19	951	887.55	6.67	3	835	797.47	4.49	4	761	740.86	2.65	5
c20	960	896.99	6.56	2	833	801.18	3.82	3	765	743.94	2.75	5

MIP model and the separation problem grows. In particular, for the instances with $|V|= 500$, in all cases the root-node could not be solved within the time limit. Note the gaps are relatively small even for the unsolved instances, the largest gap for instances with $|V|= 500$ is 3.31% for $e500-8$ with $b = 0.3$.

For instance set C, a similar situation is verified; the largest instances ($c16$ to $c20$) are also the hardest, as reported in Table 4. Only five of these fifteen instances could be solved within the time limit; interestingly, the solved instances are those with $b = 0.3$, while for EUCLIDEAN and $|V|= 200$, only the instances with $b = 0.1$ could be solved. Note that some of the smallest instances ($c01$ to $c05$) are solved in the root node.

The results show that the Lagrangian approach is an attractive choice for large-scale instances. On the one hand, because the attained gaps are often very near the root-gap achieved by the B&C approach; and on the other hand, because as the size of the instances grow, the Lagrangian approach scales better when compared to the running times of the B&C approach. This behavior can be seen in the larger EUCLIDEAN instances ($|V|= \{200, 500\}$) and in the larger C instances ($c16-c20$), where the B&C often struggles and reaches the time limit, while the Lagrange relaxation algorithm attains comparable gaps in about 40 (EUCLIDEAN) and 10 (C) seconds.

Table 3: Results of the *VHP* approach on instance set `EUCLIDEAN` (MDUMST). z^* gives the value of the best solution found, $t[s]$ the runtime, TL indicates that the instance could not be solved within a timelimit of 1 800s, $g[\%]$ the rootgap, if the instance is solved to optimality and the primal gap otherwise, and *nodes* the number of branch and bound nodes.

	b=0.1				b=0.2				b=0.3			
instance	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes
e100-1	973	9	0.51	51	908	18	0.30	38	867	9	0.32	19
e100-2	1018	14	0.65	38	955	26	0.66	256	913	75	1.01	1405
e100-3	972	14	0.53	65	906	28	0.76	65	859	14	0.68	42
e100-4	994	12	0.63	35	924	24	0.95	119	875	21	0.89	211
e100-5	1021	49	1.32	799	945	23	0.90	291	895	33	0.79	445
e100-6	1020	16	0.73	112	954	33	0.87	211	906	62	0.96	908
e100-7	1025	15	0.59	19	955	18	0.27	28	907	18	0.58	95
e100-8	1014	11	0.76	75	940	11	0.64	61	892	14	0.60	77
e100-9	954	7	0.32	15	892	9	0.34	29	848	12	0.47	32
e100-10	1016	13	0.51	53	954	20	1.12	237	907	61	1.00	773
e150-1	1160	121	0.82	403	1083	94	0.64	315	1029	43	0.39	30
e150-2	1232	125	0.73	275	1148	271	0.93	643	1088	199	0.86	782
e150-3	1211	81	0.51	63	1126	132	0.66	197	1067	58	0.60	76
e150-4	1225	95	0.53	140	1136	81	0.71	132	1074	83	0.52	44
e150-5	1256	129	0.96	412	1169	435	1.15	2379	1110	839	1.16	5414
e150-6	1227	376	0.94	2486	1141	224	0.74	1046	1080	109	0.58	307
e150-7	1235	85	0.79	367	1151	417	0.91	2598	1091	170	0.68	788
e150-8	1203	54	0.55	75	1119	66	0.69	206	1061	106	4.11	376
e150-9	1242	160	0.74	546	1154	161	0.75	570	1094	262	0.70	1198
e150-10	1208	41	0.42	115	1129	153	0.85	851	1069	56	0.53	124
e200-1	1396	725	0.82	2063	1293	846	0.60	1159	*1226*	**TL**	0.30	4435
e200-2	1438	242	0.38	64	1338	1166	0.71	3371	1260	564	0.57	1316
e200-3	1435	1404	1.03	3392	*1330*	**TL**	0.64	3473	*1258*	**TL**	0.54	3805
e200-4	1470	562	0.76	1091	*1374*	**TL**	0.52	3703	1300	306	0.55	432
e200-5	1461	526	0.73	1030	*1356*	**TL**	0.36	4000	*1282*	**TL**	0.23	4763
e200-6	1474	521	0.61	769	1370	1167	0.80	2733	*1299*	**TL**	0.61	3715
e200-7	1453	237	0.61	319	1350	461	0.60	1090	*1284*	**TL**	0.34	4797
e200-8	1407	1254	0.95	2400	*1304*	**TL**	0.13	5146	1233	359	0.66	776
e200-9	1477	1014	0.53	1034	*1379*	**TL**	0.65	3900	*1306*	**TL**	0.61	3929
e200-10	1467	1648	0.95	3976	*1371*	**TL**	0.44	3516	*1299*	**TL**	0.23	4900
e500-1	*2507*	**TL**	2.14	0	*2320*	**TL**	2.98	0	*2176*	**TL**	1.62	0
e500-2	*2459*	**TL**	1.08	0	*2279*	**TL**	2.75	0	*2141*	**TL**	1.87	0
e500-3	*2556*	**TL**	1.29	0	*2357*	**TL**	1.77	0	*2218*	**TL**	1.87	0
e500-4	*2432*	**TL**	2.17	0	*2247*	**TL**	1.36	0	*2123*	**TL**	1.48	0
e500-5	*2417*	**TL**	1.33	0	*2234*	**TL**	1.76	0	*2105*	**TL**	1.79	0
e500-6	*2470*	**TL**	1.08	0	*2298*	**TL**	1.94	0	*2166*	**TL**	1.91	0
e500-7	*2437*	**TL**	1.12	0	*2247*	**TL**	1.41	0	*2115*	**TL**	1.35	0
e500-8	*2514*	**TL**	2.59	0	*2307*	**TL**	1.76	0	*2172*	**TL**	3.31	0
e500-9	*2461*	**TL**	1.01	0	*2284*	**TL**	1.55	0	*2153*	**TL**	1.59	0
e500-10	*2493*	**TL**	1.08	0	*2317*	**TL**	1.79	0	*2185*	**TL**	1.67	0

Table 4: Results of the *VHP* approach on instance set `C` (MDUMST). z^* gives the value of the best solution found, $t[s]$ the runtime, TL indicates that the instance could not be solved within a timelimit of 1 800s, $g[\%]$ the rootgap, if the instance is solved to optimality and the primal gap otherwise, and *nodes* the number of branch and bound nodes.

	b=0.1				b=0.2				b=0.3			
instance	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes
c01	3422	2	0.98	3	3163	3	0.00	0	2978	6	0.09	10
c02	3283	5	1.97	6	3029	4	0.01	2	2855	5	0.00	0
c03	3299	15	0.00	0	3034	7	0.00	0	2854	7	0.00	1
c04	3337	9	0.36	3	3085	8	0.04	2	2910	11	0.11	5
c05	3315	6	0.01	3	3068	6	0.07	12	2887	5	0.00	2
c06	2534	32	2.87	260	2318	15	2.21	57	2172	15	0.12	30
c07	2532	30	0.18	143	2323	17	1.48	81	2179	14	0.89	15
c08	2461	25	0.26	122	2247	18	1.36	37	2105	13	0.33	21
c09	2415	37	0.54	268	2208	24	3.97	311	2069	30	0.24	257
c10	2463	22	1.95	20	2258	18	0.17	31	2116	24	0.80	152
c11	1509	102	1.07	551	1351	255	1.40	2457	1237	248	0.62	2171
c12	1553	27	1.91	75	1386	48	1.05	271	1271	56	0.38	353
c13	1524	17	1.88	68	1364	412	1.42	3647	1248	786	1.02	5245
c14	1530	50	1.32	261	1367	89	0.48	758	1241	92	0.39	871
c15	1529	22	0.42	42	1366	36	1.15	85	1249	26	0.36	199
c16	*953*	**TL**	2.66	301	*817*	**TL**	0.66	243	749	60	0.25	27
c17	*944*	**TL**	1.50	357	*814*	**TL**	0.51	270	745	49	0.26	25
c18	*946*	**TL**	2.01	370	*820*	**TL**	0.92	369	749	46	0.32	21
c19	*936*	**TL**	2.00	258	*819*	**TL**	1.20	305	748	417	0.30	210
c20	*939*	**TL**	1.38	406	*819*	**TL**	0.30	1296	754	56	0.35	34

4.3 Results for MCUMST

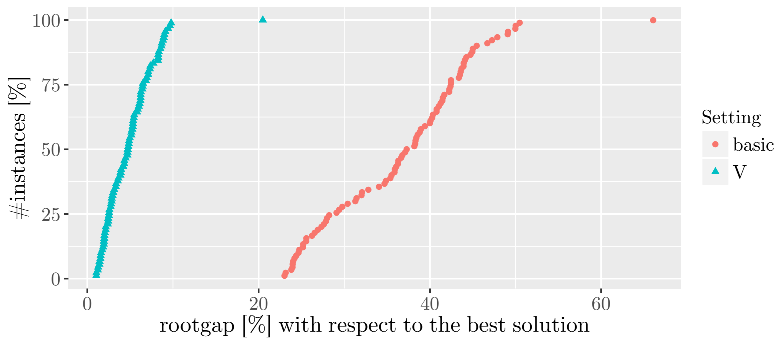
We now provide results for MCUMST in a similar fashion as for MDUMST, i.e., we first analyze the performance of the B&C configurations and then compare the Lagrangian approach and B&C.

B&C Performance The plots of the rootgaps for *basic* and V, reported in Figures 4(a) (`EUCLIDEAN`) and 4(b) (`C`), follow a pattern similar to those reported for MDUMST. The gaps for instance set `EUCLIDEAN` are up to 10% for V, and up to 50% for *basic* (excluding an outlier appearing for both settings, which may be caused by a bad best known solution). These gaps are larger than those for provided for MDUMST on the same instance set. For instance set `C`, the gaps for V are not greater than 5%, which is similar to MDUMST. For *basic*, we can again see the influence of graph sparsity on the gaps; this time it is more pronounced than for MDUMST, and the largest gaps are as high as 80%.

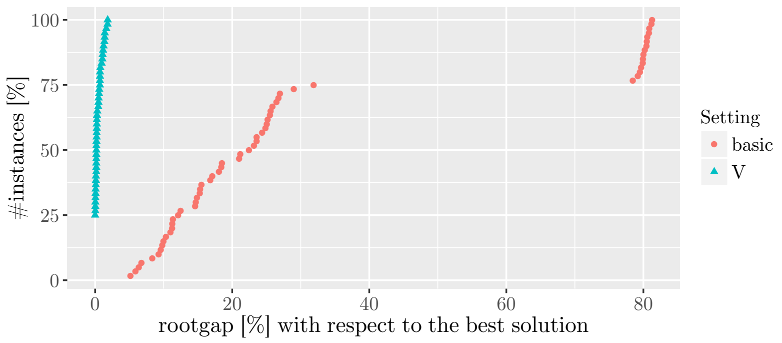
In the performance profiles comparing V, *VH* and *VHP*, given in Figures 5(a) (`EUCLIDEAN`) and 5(b) (`C`), we can again see the dominance of *VHP*. For both `EUCLIDEAN` and `C`, it is the most effective configuration in at least 75% of the instances. For `EUCLIDEAN`, over 85% of instances could be solved within the timelimit, and for `C` nearly 100%.

Lagrangian Bounds v/s *VHP* Regarding the performance of the Lagrangian approach, we see that the gaps reached for the MDUMST on `EUCLIDEAN` instances (reported in Table 5) are much higher than those attained for the MDUMST on the same instances. They are mostly in the interval of 10% to 20%; notwithstanding, instance $e500-7$ with $b = 0.3$ has a gap of 97%. Despite of this particular case, there are no other instances with such an extremely bad gap. For $|V|= 500$, b seems to influence the quality of the gaps; with $b = 0.1$, they are around 10%, while for $b = 0.3$, they are around 20%. For the smaller instances, this tendency can also be seen, but it is not so evident. The runtime is comparable to the runtime for MDUMST; for $|V|\leq 200$, it is at most 5

Figure 3: Rootgaps with respect to the best solution found by all settings for problem MCUMST on instance sets `EUCLIDEAN` and `C`.



(a) Set `EUCLIDEAN` with $|V| \in \{100, 150, 200\}$



(b) Set `C`

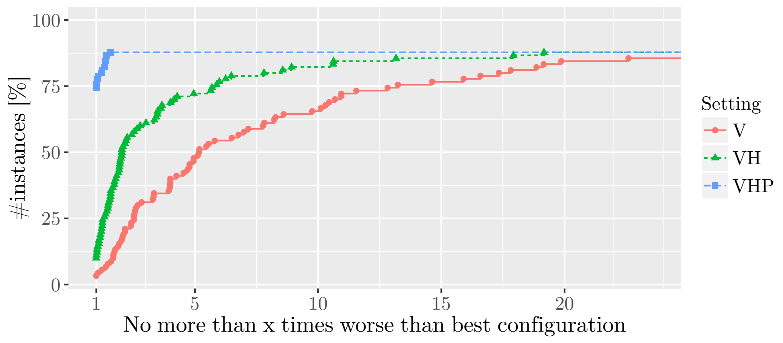
seconds, while for $|V| = 500$, it takes in $40 - 60$ seconds. However, for $e500 - 10$ and $b = 0.3$, the runtime is 94 seconds. Note that the runtime could be improved by not calling the primal heuristic at every iteration of the subgradient algorithm.

Interestingly, aside from a few outliers, the Lagrangian relaxation approach works much better for instances from set `C`, as reported in Table 6. The gaps are mostly in the range of $1 - 5\%$. The few clear outliers correspond to $c06$, $c08$ and $c09$, when setting $b = 0.3$; note that these three instances have the same number of edges (1000). On the other hand, the densest instances of this set, $c16 - c20$, which were the most troublesome when solving MDUMST, have been solved with gaps under 1% for $b = 0.1$.

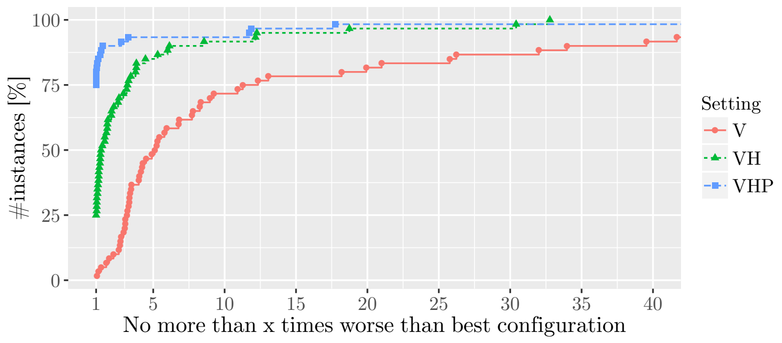
Table 7 shows the results when tackling instances `EUCLIDEAN` with the *VHP* configuration. The roo tgaps are larger than for MDUMST, and they seem to be influenced by b. They are up to 3% (aside from $e150 - 5$ with 3.90%) for $b = 0.1$ and up to 10% for $b = 0.3$. Similar to the performance for MDUMST, no instance with $|V| = 500$ and only few for $|V| = 200$ (among them all with $b = 0.1$), could be solved to optimality within the time limit. For $|V| = 500$, again the time limit is reached while still solving the root-node.

For set `C`, whose results are reported in Table 8, the *VHP* approach performs much better (as we

Figure 4: Performance profile of runtimes to optimality for problem MCUMST on instance sets `EUCLIDEAN` and `C`.



(a) Set **`EUCLIDEAN`** with $|V| \in \{100, 150, 200\}$



(b) Set `C`

as the Lagrangian relaxation did). The largest gap is 1.50% and around one third of the instances could even be solved within the root-node. Only one instance, $c17$ with $b = 0.2$ remains unsolved, note that also for $b = 0.1$ and $b = 0.3$, $c17$ is (by far) the most difficult one.

Comparing the Lagrangian approach with the B&C, a similar conclusion as for MDUMST can be drawn; namely for large-scale instances, the Lagrangian approach may be preferable. However, MCUMST seems to be more difficult as the obtained gaps for `EUCLIDEAN` are about five times higher than those attained when solving MDUMST. In particular the Lagrangian approach gives bad gaps for a handful of instances. Nonetheless, for instances from `C` the situation looks better since B&C managed to solve more instances in the root-node than for MDUMST. Overall, the performance for MCUMST seems to be more related to the instance class, while for MDUMST the size of the instance seems to be most influential characteristic.

Table 5: Results of the Lagrangian approach on instance set `EUCLIDEAN` (MCUMST). z^* gives the value of the best solution found, LB the lower bound, $g[\%]$ the gap and $t[s]$ the runtime

	b=0.1				b=0.2				b=0.3			
instance	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$
e100-1	295	278.83	5.48	1	152	138.03	9.19	1	71	63.81	10.13	1
e100-2	286	266.29	6.89	0	169	141.87	16.05	1	78	68.79	11.81	1
e100-3	279	255.96	8.26	1	156	139.15	10.80	1	81	72.44	10.57	0
e100-4	284	261.32	7.99	1	169	145.45	13.94	1	89	74.52	16.27	1
e100-5	310	274.70	11.39	1	182	150.05	17.56	1	98	79.42	18.95	1
e100-6	326	291.83	10.48	1	193	157.73	18.27	1	101	81.06	19.74	1
e100-7	297	271.93	8.44	0	178	156.53	12.06	1	94	81.97	12.80	0
e100-8	296	257.94	12.86	1	159	136.58	14.10	1	83	67.82	18.29	1
e100-9	289	264.20	8.58	0	162	144.56	10.77	1	79	72.25	8.54	1
e100-10	329	282.61	14.10	1	195	158.73	18.60	1	99	82.86	16.30	1
e150-1	398	351.98	11.56	2	208	183.31	11.87	2	104	86.30	17.02	2
e150-2	439	394.84	10.06	2	264	217.29	17.69	2	141	112.81	20.00	3
e150-3	437	394.84	9.65	2	250	217.05	13.18	2	134	113.95	14.97	2
e150-4	391	355.56	9.06	1	211	186.27	11.72	2	113	96.26	14.81	2
e150-5	445	397.68	10.63	2	278	230.68	17.02	2	151	121.06	19.83	2
e150-6	455	403.63	11.29	2	265	223.27	15.75	2	144	115.16	20.03	2
e150-7	455	411.54	9.55	2	270	226.71	16.03	2	146	117.86	19.28	2
e150-8	425	390.27	8.17	2	242	210.07	13.20	2	119	108.87	8.51	2
e150-9	459	422.53	7.94	2	272	237.79	12.58	2	145	128.41	11.44	2
e150-10	401	353.61	11.82	2	223	189.39	15.07	2	111	97.69	11.99	1
e200-1	615	551.41	10.34	4	371	309.04	16.70	5	198	165.00	16.67	4
e200-2	638	568.04	10.97	3	373	319.55	14.33	4	213	186.55	12.42	4
e200-3	658	585.75	10.98	4	399	336.45	15.68	4	227	183.83	19.02	4
e200-4	601	531.16	11.62	4	344	293.86	14.58	4	182	151.98	16.49	3
e200-5	637	563.32	11.57	4	384	322.20	16.09	4	214	175.78	17.86	4
e200-6	618	544.03	11.97	4	379	301.47	20.46	5	205	169.30	17.41	5
e200-7	635	555.54	12.51	4	384	314.14	18.19	4	199	164.88	17.15	4
e200-8	565	510.44	9.66	4	325	280.16	13.80	4	182	150.47	17.32	4
e200-9	708	630.52	10.94	4	453	366.39	19.12	4	258	207.78	19.46	3
e200-10	629	554.83	11.79	5	379	304.22	19.73	4	202	157.65	21.95	4
e500-1	1688	1525.98	9.60	46	1091	918.42	15.82	65	687	537.79	21.72	84
e500-2	1596	1449.16	9.20	55	1034	858.92	16.93	67	635	505.14	20.45	94
e500-3	1729	1564.59	9.51	54	1143	838.60	26.63	61	697	539.74	22.56	65
e500-4	1611	1466.44	8.97	63	999	833.24	16.59	61	576	458.20	20.45	79
e500-5	1647	1484.68	9.86	39	1039	871.24	16.15	41	638	486.59	23.73	73
e500-6	1706	1546.68	9.34	45	1098	909.92	17.13	62	656	518.96	20.89	92
e500-7	1633	1500.36	8.12	46	1041	823.85	20.86	50	655	16.81	97.43	60
e500-8	1667	1486.38	10.83	44	1052	875.55	16.77	64	644	513.36	20.29	74
e500-9	1728	1551.32	10.22	53	1109	901.16	18.74	65	667	529.12	20.67	69
e500-10	1672	1510.27	9.67	43	1075	874.38	18.66	54	644	509.37	20.91	94

Table 6: Results of the Lagrangian approach on instance set C (MCUMST). z^* gives the value of the best solution found, LB the lower bound, $g[\%]$ the gap and $t[s]$ the runtime

	b=0.1				b=0.2				b=0.3			
instance	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$	z^*	LB	$g[\%]$	$t[s]$
c01	1505	1480.35	1.64	1	929	907.27	2.34	1	550	543.70	1.14	0
c02	1491	1461.20	2.00	1	917	899.95	1.86	0	534	528.39	1.05	0
c03	1512	1489.54	1.49	0	956	940.96	1.57	0	576	565.65	1.80	1
c04	1481	1444.46	2.47	1	880	865.04	1.70	0	508	499.66	1.64	0
c05	1481	1438.81	2.85	0	894	865.37	3.20	1	563	328.16	41.71	0
c06	1739	1664.04	4.31	0	1163	1097.07	5.67	0	849	314.14	63.00	0
c07	1655	1594.76	3.64	1	1096	1034.60	5.60	0	700	645.42	7.80	0
c08	1676	1625.71	3.00	0	1112	1056.33	5.01	0	774	287.06	62.91	1
c09	1740	1664.22	4.36	1	1174	1102.62	6.08	1	826	313.09	62.10	0
c10	1696	1633.31	3.70	1	1130	1074.23	4.94	0	729	686.71	5.80	0
c11	2096	2027.48	3.27	1	1617	1520.69	5.96	1	1271	1152.59	9.32	0
c12	2080	2008.94	3.42	1	1594	1491.11	6.46	1	1238	1106.74	10.60	0
c13	2054	2007.68	2.26	1	1562	1453.44	6.95	0	1183	1089.06	7.94	1
c14	2129	2063.64	3.07	1	1679	1551.59	7.59	1	1305	1186.99	9.04	1
c15	2085	2026.37	2.81	1	1585	1481.95	6.50	0	1211	1105.25	8.73	1
c16	2205	2189.37	0.71	2	1773	1414.38	20.23	2	1412	1364.05	3.40	2
c17	2208	2199.08	0.40	2	1790	1737.99	2.91	3	1424	1368.74	3.88	2
c18	2199	2191.47	0.34	2	1760	1751.21	0.50	2	1415	1361.02	3.81	3
c19	2199	2185.43	0.62	2	1765	1749.12	0.90	3	1390	1354.88	2.53	3
c20	2205	2199.00	0.27	2	1784	1420.39	20.38	2	1403	1347.20	3.98	2

5. Conclusion and Further Work

In this paper, we have studied two variants of the upgrading spanning tree problem (UMST), which is a problem from the family of *budget constrained network upgrading problems* (BCNUP). In the problem, we are given a graph $G(V,E)$ and three delay values $d^0_{ij} > d^1_{ij} > d^2_{ij}$ for every edge $\{i,j\} \in E$, as well as upgrade costs c_i for each node $i \in V$. If a node i gets upgraded, the delay d^1_{ik} may be used for all its adjacent edges $\{i,k\}$; and if for an edge $\{i,j\}$ both nodes i,j are upgraded, the delay d^2_{ij} may be used for this edge. In one variant of UMST, the goal is to find the minimum delay spanning tree, while not exceeding a given upgrade budget B; while in the other variant, the goal is to find a minimum cost upgrading scheme, such that the minimum delay spanning tree does not exceed a given bound W. The problem has been introduced in [Krumke et al., 1999], where it was studied from an approximation perspective and arises in telecommunication and also in the design of electrical power grids.

We propose Lagrangian relaxation and branch-and-cut (B&C) approaches for both problems. The proposed approaches are enhanced using valid inequalities, and we also design primal heuristics and present variable fixing procedures for the Lagrangian approaches. We asses the efficiency of our proposed solution methods in a computational study. The B&C approach provides exact solutions for instances with up to 200 nodes and 19900 edges (resp. 500 nodes and 12500 edges) for the MDUMST (resp. MCUMST) within a time limit of 1800 seconds (although it typically takes much less). The proposed valid inequalities are shown to be very important for the efficacy of the algorithms; likewise, the designed primal heuristics and a strategy to set branching priorities also have a positive effect on the performance. The Lagrangian approaches are able to provide small optimality gaps for large-scale instances with 500 nodes and 124750 edges in about 50 seconds for most of the problems of this size; this makes the Lagrangian approaches an attractive alternative to the B&C algorithm, especially when dealing with large-scale instances.

Similar approaches to the proposed ones should also be applicable to other problems of the family

Table 7: Results of the *VHP* configuration on instance set `EUCLIDEAN` (MCUMST). z^* gives the value of the best solution found, $t[s]$ the runtime, TL indicates that the instance could not be solved within a timelimit of 1 800s, $g[\%]$ the rootgap, if the instance is solved to optimality and the primal gap otherwise, and *nodes* the number of branch and bound nodes.

	b=0.1				b=0.2				b=0.3			
instance	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes
e100-1	291	5	0.43	7	148	6	2.87	14	70	6	6.12	20
e100-2	284	14	2.55	73	164	50	7.71	1005	77	16	6.60	71
e100-3	277	11	2.14	93	154	13	4.97	58	79	10	5.30	39
e100-4	279	9	1.46	9	163	19	6.15	271	85	8	6.82	31
e100-5	305	7	1.12	32	174	23	5.34	297	94	18	9.31	315
e100-6	315	10	1.65	45	184	48	6.90	857	96	17	8.12	144
e100-7	290	7	1.72	18	173	16	3.81	137	91	7	3.39	12
e100-8	283	7	1.19	35	154	11	4.68	83	81	9	7.02	79
e100-9	281	7	1.86	26	156	11	3.14	87	78	9	3.03	19
e100-10	314	8	2.50	54	187	31	8.29	589	96	15	9.56	231
e150-1	382	15	0.54	6	204	49	4.27	195	100	93	8.08	553
e150-2	424	26	0.96	12	245	216	6.06	1138	133	179	8.66	919
e150-3	425	61	2.82	255	243	64	3.66	113	129	49	5.32	99
e150-4	384	49	2.64	129	205	38	4.02	55	108	32	5.14	47
e150-5	433	94	3.90	476	265	741	6.93	3990	144	171	8.52	901
e150-6	439	47	1.65	117	252	125	4.86	532	139	860	8.56	5881
e150-7	447	26	1.98	77	260	91	4.84	646	140	258	7.02	1577
e150-8	418	24	1.30	14	233	129	4.72	763	118	30	3.18	19
e150-9	450	32	2.29	55	265	182	4.47	1105	145	82	6.36	379
e150-10	391	30	2.15	91	215	72	4.72	349	110	194	8.74	1511
e200-1	598	197	2.68	371	*353*	**TL**	5.42	1969	192	736	5.28	1473
e200-2	615	166	2.06	397	361	705	4.03	1198	207	941	4.97	1529
e200-3	638	242	1.83	427	*391*	**TL**	6.76	3369	*220*	**TL**	6.05	1133
e200-4	585	238	2.17	442	331	247	3.73	473	*178*	**TL**	9.18	3515
e200-5	615	419	2.51	1093	365	1072	5.26	3125	*206*	**TL**	8.21	4509
e200-6	604	677	2.88	1856	*359*	**TL**	7.77	2959	195	635	5.80	1513
e200-7	615	702	1.84	674	*369*	**TL**	5.84	4647	192	234	4.73	320
e200-8	550	232	2.02	165	320	821	4.89	1630	173	1255	7.12	2323
e200-9	687	790	2.45	801	*434*	**TL**	6.08	4218	*248*	**TL**	9.06	3233
e200-10	614	662	2.81	1870	*362*	**TL**	5.94	3650	*197*	**TL**	10.33	2954
e500-1	*1688*	**TL**	4.99	0	*1091*	**TL**	7.65	0	*692*	**TL**	11.78	0
e500-2	*1592*	**TL**	4.79	0	*1028*	**TL**	8.78	0	*636*	**TL**	11.47	0
e500-3	*1721*	**TL**	3.63	0	*1113*	**TL**	8.94	0	*693*	**TL**	11.63	0
e500-4	*1600*	**TL**	4.22	0	*1000*	**TL**	8.39	0	*590*	**TL**	10.62	0
e500-5	*1629*	**TL**	4.37	0	*1040*	**TL**	7.77	0	*632*	**TL**	10.10	0
e500-6	*1706*	**TL**	4.85	0	*1085*	**TL**	8.19	0	*666*	**TL**	14.00	0
e500-7	*1620*	**TL**	2.62	0	*1045*	**TL**	9.66	0	*612*	**TL**	8.85	0
e500-8	*1646*	**TL**	4.74	0	*1059*	**TL**	9.54	0	*632*	**TL**	8.62	0
e500-9	*1717*	**TL**	4.95	0	*1108*	**TL**	9.60	0	*665*	**TL**	11.14	0
e500-10	*1655*	**TL**	3.38	0	*1064*	**TL**	6.98	0	*648*	**TL**	11.11	0

of BCNUPs. For example, an extension of the proposed approach to a version of UMST where the spanning tree constraint gets replaced with a Steiner tree constraint is straightforward, by, e.g., using Steiner connectivity cuts [Koch and Martin, 1998, Ljubić et al., 2006], or by using a spanning tree-based Steiner tree relaxation in the Lagrangian approach [see, e.g., Salles da Cunha et al., 2009]. Another fruitful avenue of further research could be the design of an algorithm based on Benders decomposition for the studied problems, as the presented formulation gives very good bounds, but suffers from scalability issues. Benders decomposition with only keeping the updating variables in the master may alleviate these issues. Again, such a Benders decomposition approach should also be applicable to other problems of the family BCNUP. Finally, studying a bi-objective version of the problem taking into account both objectives at the same time may also be a worthwhile topic for further studies.

Table 8: Results of the *VHP* configuration on instance set `C` (MCUMST). z^* gives the value of the best solution found, $t[s]$ the runtime, TL indicates that the instance could not be solved within a timelimit of 1 800s, $g[\%]$ the rootgap, if the instance is solved to optimality and the primal gap otherwise, and *nodes* the number of branch and bound nodes.

	b=0.1				b=0.2				b=0.3			
instance	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes	z^*	$t[s]$	$g[\%]$	nodes
c01	1501	4	0.07	5	919	3	0.82	9	547	3	0.00	2
c02	1482	3	0.24	6	911	4	0.00	0	531	3	0.19	22
c03	1507	4	0.27	7	949	4	0.00	0	571	3	0.00	0
c04	1467	5	0.00	0	876	6	0.51	7	505	5	0.30	9
c05	1468	4	0.37	12	878	3	0.11	6	514	3	0.39	12
c06	1716	6	0.17	9	1140	6	0.22	14	731	10	0.65	46
c07	1641	5	0.00	1	1066	5	0.09	2	668	7	1.09	17
c08	1667	6	0.28	29	1089	7	0.96	30	696	8	0.81	50
c09	1720	7	0.82	45	1151	6	0.70	27	749	16	1.18	195
c10	1676	6	0.17	5	1108	6	1.50	7	711	26	1.76	313
c11	2063	2	0.00	0	1581	31	0.46	143	1222	203	1.37	1846
c12	2045	1	0.00	0	1551	4	0.21	16	1186	34	0.53	164
c13	2037	4	0.34	19	1519	20	0.72	67	1153	426	1.40	3867
c14	2095	3	0.00	0	1634	64	0.78	536	1266	82	1.39	572
c15	2056	3	0.00	0	1551	59	0.97	572	1174	135	1.23	1514
c16	2201	29	0.18	17	1762	62	0.17	37	1375	74	0.25	38
c17	2208	330	0.18	167	*1773*	**TL**	0.27	765	1383	639	0.22	394
c18	2199	4	0.00	0	1758	7	0.00	0	1373	7	0.00	0
c19	2199	5	0.00	0	1758	7	0.00	0	1365	715	0.17	270
c20	2202	5	0.00	0	1758	5	0.00	0	1366	10	0.00	0

Acknowledgment E. Álvarez-Miranda acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the grant FONDECYT N.11140060 and through the Complex Engineering Systems Institute (ICM:P-05-004-F, CONICYT:FBO16). The research of M. Sinnl was supported by the Austrian Research Fund (FWF, Project P 26755-N19).

References

E. Álvarez-Miranda, M. Luipersbeck, and M. Sinnl. Optimal upgrading schemes for effective shortest paths in networks. In B. Gendron and C. Quimper, editors, *to appear in Proceedings of CPAIOR 2016*, LNCS. Springer, 2016.

A. Costa, P. França, and C. Lyra. Two-level network design with intermediate facilities: An application to electrical distribution systems. *Omega*, 39(1):3–13, 2011.

B. Dilkina, J. Lai, and C. Gomes. Upgrading shortest paths in networks. In T. Achterberg and C. Beck, editors, *Proceedings of CPAIOR 2011*, volume 6697 of *LNCS*, pages 76–91. Springer, 2011.

E. D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

M. X. Goemans. The steiner tree polytope and related polyhedra. *Mathematical Programming*, 63 (1-3):157–182, 1994.

M. Haouari, S. Layeb, and H. Sherali. The prize collecting Steiner tree problem: models and lagrangian dual optimization approaches. *Computational Optimization and Applications*, 40(1): 13–39, 2008.

R. Hemmecke, R. Schültz, and D. Woodruff. Interdicting stochastic networks with binary interdiction effort. In D. Woodruff, editor, *Network Interdiction and Stochastic Integer Programming*, volume 22 of *Operations Research/Computer Science Interfaces Series*, pages 69–84. Springer, 2003.

T. Ibaraki, Y. Vaxès, and X. Yang. Lowering eccentricity of a tree by node upgrading. *Networks*, 45, 2005.

E. Israeli and R. Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

D. Johnson, M. Minkoff, and S. Phillips. The prize collecting steiner tree problem: theory and practice. In D. Shmoys, editor, *Proceedings of the 11th Symposium on Discrete Algorithms*, pages 760–769. ACM/SIAM, 2000.

T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3): 207–232, 1998.

T. Koch, A. Martin, and S. Voß. *SteinLib: An updated library on Steiner tree problems in graphs.* Springer, 2001.

S. Krumke, M. Marathe, H. Noltemeier, R. Ravi, S. Ravi, R. Sundaram, and H. Wirth. Improving minimum cost spanning trees by upgrading nodes. *Journal of Algorithms*, 33(1):92–111, 1999.

I. Ljubić, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. *Mathematical Progamming*, Series B(105):427–449, 2006.

A. Lucena and M. Resende. Strong lower bounds for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141(1-3):277–294, 2004.

S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., 1990.

D. Paik and S. Sahni. Network upgrading problems. *Networks*, 26(1):45–58, 1995.

A. Salles da Cunha, A. Lucena, N. Maculan, and M. Resende. A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 157(6):1198–1217, 2009.

M. Savelsbergh and T. Volgenant. Edge exchanges in the degree-constrained minimum spanning tree problem. *Computers & Operations Research*, 12(4):341–348, 1985.

H. Sherali and O. Ulular. Conjugate gradient methods using quasi-newton updates with inexact line searches. *Journal of Mathematical Analysis and Applications*, 150(2):359–377, 1990.

L. Wolsey. *Integer Programming.* Wiley, 1998.