Mixed-Integer Programming Approaches for the Tree t^*-spanner Problem

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

The tree t^*-spanner problem is an NP-hard problem, which is concerned with finding a spanning tree in a given undirected weighted graph, such that for each pair of nodes the ratio of the shortest distance in the spanning tree and the shortest distance in the given graph is bounded by t. The goal is to find a spanning tree, which gives the minimal t. This problem is associated with many network design applications, but in particular, in the context of architecture of distributed systems.

We introduce mixed-integer programming formulations for the tree t^*-spanner problem, and present a branch-and-cut solution approach based on these formulations. The branch-and-cut is enhanced with an initialization procedure and a primal heuristic. A computational study is done to assess the effectiveness of our proposed algorithmic strategies. To the best of our knowledge, this is the first time that an exact approach is proposed for this problem.

1. Introduction and motivation

Given an edge-weighted undirected graph $G(V, E)$, a subgraph $G'(V, E')$ is called a *t-spanner* if the ratio, say r_{uv}, between $D^{uv}_{G'}$ (the shortest distance between u and v in G') and D^{uv} (the shortest distance between u and v in G) is at most $t \geq 1$ for all pairs (u, v) in G. The term *t-spanner* was first introduced in Peleg and Ullman (1989), in the context of designing synchronous protocols in distributed computation. Furthermore, for a given $G(V, E)$ the *stretch factor* , or simply *stretch*, of a subgraph $G'(V, E')$ corresponds to $t^* = \max_{(u,v)\in V} r_{uv}$, i.e., the maximum ratio among all pairs of nodes in V; t^* can also be regarded as a *worst-case* performance measure of G' with respect to G (see, e.g., Bharath-Kumar and Jaffe, 1983; Peleg and Upfal, 1988, for early references using this concept).

In this paper, we study the problem of finding, in a given undirected weighted graph, a spanning tree so that the corresponding stretch factor is minimum. This problem is known as the tree t^*-spanner problem (Tr-t^*-Sp) or minimum (max) stretch spanning tree problem, and it was first presented in (Cai and Corneil, 1995), where it was shown to be NP-hard. The problem can be formally defined as follows.

Definition 1 *(Tree t^*-spanner problem) Let $G = (V, E)$ be a given undirected graph with cost function $\mathbf{c} : E \to \mathbb{R}_{\geq 0}$, and for each pair of nodes $s, u \in V$, let D^{su} be the shortest distance (i.e, length of the shortest path) between s and u for the cost function $\mathbf{c}$. The* tree t-spanner problem *is the problem of finding a spanning tree $G' = (V, E)$ of $G = (V, E)$ so that the induced stretch factor t^* is minimum.*

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

Network design problems involving t-spanners have been studied over the last three decades. They are mainly associated with applications in the design of distributed systems and communication networks in parallel computing (see, e.g. Althöfer et al., 1993; Awerbuch, 1985; Awerbuch et al., 1992; Bhatt et al., 1986; Liestman and Shermer, 1993; Peleg and Ullman, 1989; Peleg and Upfal, 1988). However, t-spanners have been also used for the reconstruction of phylogenetic trees (Bandelt and Dress, 1986) and motion planning in robotics control optimization (see, e.g., Cai and Mark Keil, 1997; Marble and Bekris, 2017).

To algorithmically tackle t-spanners problems, approximations algorithms have been developed (see, e.g. Althöfer et al., 1993; Cai and Corneil, 1995; Elkin et al., 2008; Peleg and Ullman, 1989; Venkatesan et al., 1997, and the references therein). For the particular case of the Tr-t^*-Sp, (Emek and Peleg, 2009) gave an $O(\log n)$-approximation algorithm. Very recently, two metaheuristics for the Tr-t^*-Sp were proposed, namely a genetic algorithm and an artificial bee colony algorithm (Singh and Sundar, 2018).

Contribution and Outline Despite of the broad body of literature devoted to studying theoretical and practical issues of t-spanner problems, including the Tr-t^*-Sp, to the best of our knowledge, no exact approaches exist for solving these problems.

In this work, we present a first exact approach for the Tr-t^*-Sp, using mixed-integer programming (MIP). We provide a compact first formulation, and an exponential-size formulation. While the first formulation is tackled by directly using an off-the-shelf MIP solver, the second is approached by a specially tailored hybrid decomposition algorithm. In order to study the effectiveness of the proposed approaches, we generate instances according to the rules described in (Singh and Sundar, 2018), and carry out extensive numerical experiments. The obtained results show that the Tr-t^*-Sp is an extremely challenging optimization problem, as MIP ad-hoc methods manage to compute optimal solutions only for instances of limited size.

The paper is organized as follows. In Section 2 we give a compact mixed-integer programming formulation of the problem. An exponential-size formulation along with branch-and-cut (B&C) algorithm, which is based on a Benders-like decomposition, is presented in Section 3. In Section 4 we report an extensive computational study, and final remarks and conclusions are reported in Section (5).

2. A compact MIP formulation for the Tr-t^*-Sp

The formulation presented in this section relies on a polynomial-size representation of the spanning tree polyhedron, based on multi-commodity flows, as well as on a single-commodity flow modeling strategy for the shortest paths. For this formulation, the following notation is needed. Besides the original graph G, we also require its bi-directed counterpart, $G_d = (V, A)$, where $A = \{(i,j),(j,i) \mid \{i,j\} \in E\}$ and the arc costs are given by $c_{ij} = c_{ji} = c_e$ for all $e : \{i,j\} \in E$. For a set $S \subset V$, let $\delta^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$ and $\delta^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ be the incoming and outgoing *cutset*, respectively. Finally, let $P = \{(s,u) \in V \times V : s < u\}$ be the set of all pairs of nodes in V.

Along with the presented notation, we need three set of decision variables. Let $\mathbf{x} \in \{0,1\}^{|E|}$ be a vector of binary variables, so that $x_{ij} = 1$, if edge $e : \{i,j\} \in E$ is in the spanning tree solution, and $x_{ij} = 0$, otherwise. Complementary, let $\mathbf{y} \in \{0,1\}^{|A|\times|P|}$ be a vector a binary variables so that $y_{ij}^{su} = 1$ if arc $(i,j) \in A$ is part of the path between the pair $(s,u) \in P$ in the solution, and $y_{ij}^{su} = 0$, otherwise. And finally, let θ be a continuous auxiliary variable for measuring the value of the largest stretch. Using these variables and notation presented above, we obtain the following formulation, which we will refer to as (MCF), for the

Tr-t^*-Sp.

$$
\begin{aligned}
\text{(MCF)} \qquad & t^* = \min \theta && && \text{(MCF.1)}\\
\text{s.t.} \qquad & \theta \geq \frac{1}{D^{su}} \sum_{(i,j)\in A} c_{ij} y_{ij}^{su} && \forall (s,u) \in P && \text{(MCF.2)}\\
& \sum_{(i,v)\in\delta^-(v)} y_{iv}^{su} - \sum_{(v,i)\in\delta^+(v)} y_{vi}^{su} = \begin{cases} -1, & v = s \\ 0, & v \neq s,u \\ 1, & v = u \end{cases} && \forall v \in V,\ \forall (s,u) \in P && \text{(MCF.3)}\\
& y_{ij}^{su} + y_{ji}^{su} \leq x_{ij}, && \forall \{i,j\} \in E,\ \forall (s,u) \in P && \text{(MCF.4)}\\
& \mathbf{x} \in SpT && && \text{(MCF.5)}\\
& \theta \geq 0, \quad \mathbf{y} \in \{0,1\}^{|A|\times|P|} \quad \text{and} \quad \mathbf{x} \in \{0,1\}^{|E|} && && \text{(MCF.6)}
\end{aligned}
$$

The objective function (MCF.1) minimizes the value θ, while constraints (MCF.2) ensure that θ encodes the stretch factor. Constraint-set (MCF.3) ensures that for all s,u-pairs, the corresponding $\mathbf{y}$-variables define a path between s and u. We observe, that the $\mathbf{y}$-variables can be relaxed to be continuous, since for each s,u-pair they describe the shortest path problem in the graph induced by the $\mathbf{x}$-variables. Constraints (MCF.4) ensure that only those edges which are in the spanning tree (i.e., those associated $\mathbf{x}$-variables taking value 1) can be used for the s,u-paths. The spanning tree topology induced by $\mathbf{x}$-variables is generically modeled by constraint (MCF.5) (further details are given below). And finally, the nature of the variable is imposed by constraint (MCF.6).

To obtain a polynomial-size formulation, constraint (MCF.5) is modeled by the well-known multicommodity flow formulation for the spanning tree (see, e.g., Magnanti and Wolsey, 1995). For this formulation, let $r \in V$ be a dedicated root node, and let $\mathbf{f} \in [0,1]^{|V\setminus\{r\}|\times|A|}$ be a vector of auxiliary variables so that f_{ij}^k is the *flow* from r to $k \in V \setminus \{r\}$ that passes through (i,j) A. With this notation, the space of spanning trees of G can be modeled as:

$$
\begin{aligned}
& \sum_{(i,j)\in\delta^-(r)} f_{ij}^k - \sum_{(i,j)\in\delta^+(r)} f_{ij}^k = \begin{cases} -1, & v = r \\ 0, & v \neq r,k\ , \\ 1, & v = k \end{cases} && \forall k \in V\setminus\{r\}, v \in V && \text{(MCF.5a)}\\
& f_{ij}^k \leq w_{ij} && \forall (i,j) \in A && \text{(MCF.5b)}\\
& \sum_{(i,j)\in A} w_{ij} = |V| - 1 && && \text{(MCF.5c)}\\
& w_{ij} + w_{ji} = x_{ij} && \forall \{i,j\} \in E && \text{(MCF.5d)}\\
& \mathbf{f} \in [0,1]^{|V\setminus\{r\}|\times|A|} \quad \text{and} \quad \mathbf{w} \in \{0,1\}^{|A|} && && \text{(MCF.5e)}
\end{aligned}
$$

Using constraints (MCF.5a)-(MCF.5e) to model $\mathbf{x} \in SpT$, we get that (MCF) is a polynomial-size, i.e., compact, MIP formulation for the Tr-t^*-Sp. This formulation can be directly solved by an off-the shelf MIP solver; however, it is still quite large ($O(n^4)$ variables and $O(n^4)$ constraints), which burdens the capacity of solvers even for small instances (see Section 4 for computational results). As alternative, we present in the following section a MIP approach, based on a formulation with fewer variables, but an exponential number of constraints. In this approach, the constraints are separated and added *on-the-fly* within a B&C framework.

3. An algorithmic framework for the Tr-t^*-Sp

To obtain a formulation with a fewer number of variables, we proceed in two steps: (i) we use an alternative formulation for imposing a spanning tree topology; and (ii) we use a Benders-like Decomposition to get rid of the variables associated with the shortest paths.

Instead of using constraints (MCF.5a), (MCF.5b) and (MCF.5e) for imposing a spanning tree topolgy to the solution induced by $\mathbf{x}$, we use a connectivity cut-based formulation for the Spanning Arborescence problem (see, e.g., Magnanti and Wolsey, 1995), which is encoded by constraints

$$\sum_{(i,j)\in\delta^-(S)} w_{ij} \geq 1, \quad \forall S \subset V \mid r \notin S \tag{CUT.1}$$

$$\text{(MCF.5c) and (MCF.5d);} \tag{CUT.2}$$

the exponential-sized set of constraints (CUT.1) ensures that there is directed path (induced by $\mathbf{w}$) from r to all nodes in $V\backslash\{r\}$, while constraints (CUT.2) complementary ensure that the solution is cycle-free as well as a correct linkage with the $\mathbf{x}$ variables. As typical for such algorithmic approaches with an exponential number of constraints, we use a dynamic separation scheme that adds them on-the-fly within a B&C framework (further details are given in the Section 3.1).

One of the bottlenecks of formulation (MCF) are constraints (MCF.3), which explicitly model the paths among all node pairs in P. In order to avoid these constraints, we project them out, along with the corresponding $\mathbf{y}$ variables, through a Benders-like decomposition. In this sense, we observe that the Tr-t^*-Sp shares similarities with fixed-charge network design problems, such as the uncapacitated network design problem (UNDP), which have been successfully tackled with Benders decomposition (see, e.g., Costa, 2005; Magnanti et al., 1986). In the UNDP, the goal is to design a network and then route demand between node pairs, from a given set of pairs, on the selected network, while minimizing the total design and routing cost. Roughly speaking, at each iteration of a Benders decomposition approach for the UNDP, the solution of the *master* problem corresponds to a feasible designed network, while the *subproblems* consist of shortest paths problems, for the given set of node pairs, on the network induced by such *master* solution; the sum of the costs of all these shortest paths is encoded as a Benders cut which is therefore added to the formulation of the Master, and the process is repeated until a stopping criterion is verified.

Assuming a classical Benders decomposition scheme (see Costa, 2005, for further details), we have that in the case of the Tr-t^*-Sp, the Master solution corresponds to a spanning tree, while the subproblems correspond to shortest paths problems among all node pairs. Thus, instead of a single Benders cut, resulting from the total cost of all shortest paths, for the Tr-t^*-Sp we add one Benders cut for each shortest path problem, which ensures that θ will correspond to the stretch factor (see constraints (MCF.2)). In this way, instead of using constraints (MCF.2), (MCF.3) and (MCF.4), the optimal stretch factor, and the shortest path inducing it, can be encoded by so-called Benders optimality cuts. Hence, at a given iteration, say ℓ, the formulation of the ℓ-th master problem is given by

$$(\text{Master})^\ell \qquad t^*_\ell = \min \theta \tag{MP.1}$$

$$\text{s.t.} \quad \theta \geq \Theta(\mathbf{x}, l),\ \forall l \in \mathcal{L}^\ell \tag{MP.2}$$

$$\text{(MCF.5c), (MCF.5d), (CUT.1)} \tag{MP.3}$$

$$\mathbf{w} \in \{0,1\}^{|A|} \text{ and } \mathbf{x} \in \{0,1\}^{|E|}, \tag{MP.4}$$

where $\mathcal{L}^\ell$ accounts for the set of the Benders (optimality) cuts found up to the ℓ-th iteration, while function $\Theta(\mathbf{x}, l)$ induces the l-th Benders cut (with $l \in \mathcal{L}^\ell$). Note that constraints (MP.3) ensure the spanning tree topology of the master solution. For a given master solution, say $\mathbf{x}^{\ell'}$, and a given node pair $(s, u) \in P$, the

corresponding subproblem is given by

$$\text{(Subproblem)}^{\ell'}_{s,u} \qquad SP^{\ell'}_{s,u} = \min \frac{1}{D^{su}} \sum_{(i,j)\in A} c_{ij} y^{su}_{ij} \tag{SP.1}$$

$$\text{s.t.} \quad \sum_{(i,v)\in\delta^-(v)} y^{su}_{iv} - \sum_{(v,i)\in\delta^+(v)} y^{su}_{vi} = \begin{cases} -1, & v = s \\ 0, & v \neq s, u \\ 1, & v = u \end{cases}, \qquad \forall v \in V \tag{SP.2}$$

$$y^{su}_{ij} + y^{su}_{ji} \leq x^{\ell'}_{ij}, \qquad \forall \{i,j\} \in E \tag{SP.3}$$

$$\mathbf{y}^{su} \in [0,1]^{|A|}, \tag{SP.4}$$

i.e., simply the (weighted) shortest path problem between u and u on the network induced by $\mathbf{x}^{\ell'}$ (which is ensured by (SP.3)). The corresponding Benders optimality cut is

$$\theta \geq \alpha - \sum_{e\in E} \beta_e x_e, \tag{BC}$$

where α is a constant associated with the dual multipliers of constraint set (SP.2), and $\boldsymbol{\beta} \in \mathbb{R}^{|E|}_{\geq 0}$ are the dual multipliers associated with constraints (SP.4). This cut is added to formulation of the master problem for imposing a stronger valid lower-bound on θ, and the process is repeated until a stopping criterion (e.g., a certain solution quality is attained or a time limit is reached) is met.

The decomposition scheme as described above can be regarded as *classical* Benders decomposition (see, e.g., Fischetti et al., 2017b, for a recent discussion on this), since it assumes that the master problem is solved *to optimality* at each iteration. Nonetheless, in our case the master problem (MP.1)-(MP.4) is actually a quite hard problem (due to constraints (CUT.1)), so instead of solving it to optimality at each iteration, we rather embed the whole decomposition into a B&C scheme so that each node of the search tree, cut-set inequalities (CUT.1) and Benders cuts (BC) are both separated from the corresponding Linear Programming (LP) solution. Such strategy exploits the benefits of the both, decomposition and B&C, allowing to hybridize the separation of both types of cuts, which ultimately leads to accelerate the improvement of both dual and primal bounds (see, e.g., Álvarez-Miranda et al., 2015; Fischetti et al., 2017b, for recent examples on applying this technique). Further details are given in the remainder of this section.

3.1 Separation of cut-set inequalities (CUT.1)

Let $(\tilde{\mathbf{x}}, \tilde{\mathbf{w}})$ be the LP solution at a given node of the B&C tree. If the solution is fractional, then following separation procedure is performed. Let $\tilde{D}_d = (V, A, \tilde{\mathbf{x}}, \tilde{\mathbf{w}})$ be a capacitated graph with $\tilde{\mathbf{w}}$ as capacities. Violated connectivity cuts of type (CUT.1) are identified by solving maximum flow problems on $\tilde{G}_d$ from the dedicated root node r to all other nodes in $V \setminus \{r\}$ (see, e.g., Fischetti et al., 2017a, for further details). Concretely, the separation is performed using the preflow-push maximum flow algorithm (Cherkassky and Goldberg, 1995). This max-flow-based separation is further exploited by the separation of so-called *minimal* and *back* cuts (see Koch and Martin, 1998, for further details on these standard procedures).

Instead, if the LP solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{w}})$ yields integer values, then a purely combinatorial scheme is performed. For an arbitrary node, say v', we construct a connected component, say $\tilde{H}_{v'}(\tilde{\mathbf{w}})$, comprised solely by those arcs induced by $\tilde{\mathbf{w}}$, using a *reverse* breadth-first search (BFS) strategy. If the root node r is reached, then the construction process stops, since we have verified that v' can be reached from r on the current solution. On the contrary if the BFS completes the search process and the node r is not reached, then all the nodes comprising $\tilde{H}_{v'}(\tilde{\mathbf{w}})$ induce a violated cut-set (CUT.1). To ensure connectivity of the solution, this *reverse* BFS is done from each node.

3.2 Separation of Benders optimality cuts (BC)

In our approach, we embed the separation of Benders optimality cuts (BC) within the B&C framework. In particular, at each node of the B&C tree, we first separate inequalities (CUT.1) (as described in the previous

section), and then only separate Benders cuts for paths between nodes u and v, for which the flow to the root node r has value one. With this, we ensure, that the Benders subproblem for this pair is feasible, as this implies an flow of one between u and v in the space of $\mathbf{x}$-variables. Note that we do not solve the Benders subproblem for other pairs (i.e., where the subproblem is infeasible), as the resulting feasibility cuts would just ensure connectivity of the solution (in the space of $\mathbf{x}$-variables) and for ensuring connectivity of the solution, we already separated constraints (CUT.1). The separation of Benders optimality cuts is done by solving the LP (SP.1)-(SP.4) for each admissible u, v pair and the given $\tilde{\mathbf{x}}$ at the node of the B&C tree.

3.3 Further implementation details

We initialize our model with a set of cut-set inequalities (CUT.1). These are obtained by using Wong's dual ascent algorithm Wong (1984) (which Wong proposed for the Steiner arborescence problem, but in the spanning arborescence case is similar to the well-known Edmonds's algorithm, (see Wong, 1984)) to find a minimum spanning arborescence. Moreover, we also initialize the model with a set of Benders optimality cuts. These Benders cuts are obtained by running our separation routine as described in Section 3.2 on the solution $\hat{\mathbf{x}}$ implied by the minimum spanning tree (MST) of G.

To find high-quality primal solution within the B&C, we implemented a primal heuristic, which is run at the end of every node of the B&C (and for each LP-solution at the root node of it). The primal heuristic works by constructing a MST using edge weights $c_e(1 - \tilde{x}_e)$, where $\tilde{\mathbf{x}}$ is the LP solution at the node. Finally, the MST using the original edge weights is added as primal solution at initialization. The MSTs are calculated using Prim's algorithm Prim (1957).

4. Computational results

The B&C framework was implemented in C++ using CPLEX 12.7, which was left at default settings. The runs were carried out on an Intel Xeon E5 v4 CPU with 2.2 GHz and 3GB memory and using a single thread. The timelimit for a run was set to 600 seconds.

4.1 Instances

To create instances for our computational experiments, we followed the procedure described in (Singh and Sundar, 2018) (unfortunately the instances created in (Singh and Sundar, 2018) were not available): For a given number of nodes $|V|$, complete Euclidean graphs are created in a 100x100 plane. The Euclidean distance is used as cost function. Ten random complete Euclidean graphs are constructed this way for a given $|V|$. Based on each complete graph, three additional sparser graphs are constructed by removing edges from it (while ensuring that the graph is still connected). In these graphs, 80%, 60% and 40%, respectively, of the edges of the underlying complete graph is kept. We created instances for $|V| = \{10, 15, 20\}$. Thus, our test-set has 120 instances. The instances are made available online at `https://msinnl.github.io/pages/instancescodes.html`.

4.2 Analyzing the ingredients of the framework

To analyze the ingredient of our framework, we compare the following six different configurations.

- `mcf`: The compact formulation (MCF)
- `b`: The Benders approach, without initialization or starting heuristic or primal heuristic, and only separation in case that the LP solution at a B&C-node is integer
- `bh`: `b` but with the staring heuristic and primal heuristic
- `bhi`: `bh` but with the initialization

- `bhif`: `bhi` but with separation also for fractional LP solutions. In order to speed-up fractional separation, for each node in the B&C-tree except the root node, only one round of separation is done. This turned out to be the most effective strategy for separation in preliminary runs
- `bhif+`: `bhif` but separation of fractional solutions only in the first two nodes (and the root node) of the B&C-tree

Figure 1 gives a plot of the optimality gap for these configurations. The optimality gap is calculated as $100 \cdot (z^* - LB)/(z^*)$, where z^* is the best solution found by the configuration and LB is the obtained lower bound. The plot reveals, that all ingredients in the B&C bring a further improvement in the performance, and the setting `bhif+` with all improvement gives the best performance. Moreover, already in it most basic setting `b`, the B&C outperforms the compact approach `mcf`.

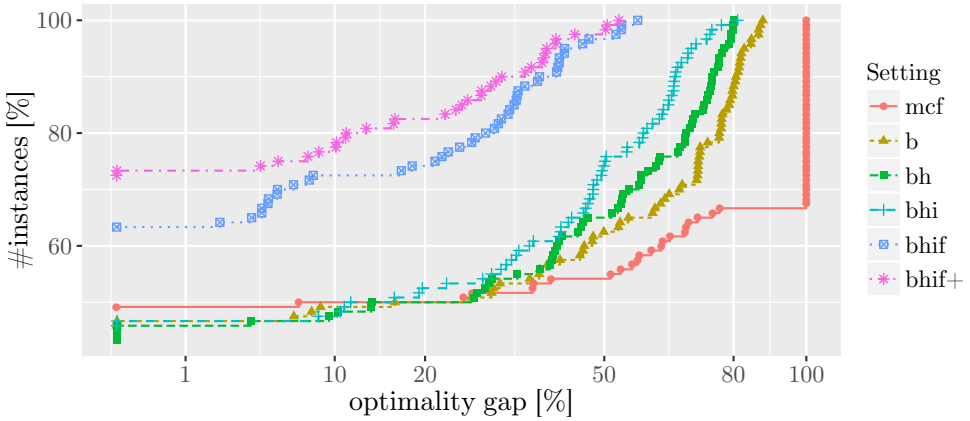

Figure 1: Optimality gap for six different configurations.

4.3 Detailed results

In Tables 1-3 we report detailed results for the instances with 10, 15 and 20 nodes, respectively, when using setting `bhif+`. In these Tables, we report for each instance the runtime in seconds ($t[s]$, with TL indicating that the timelimit of 600 seconds was reached), the lower bound (LB), the value of the best solution found (z^*), the optimality gap ($g[\%]$), the number of nodes in the branch-and-bound tree (#BBn), the time spent at the root node ($t_r[s]$), the lower bound at the root node (LB_r), the optimality gap at the root node ($g_r[\%]$, calculated as $(z^* - LB_r)/(z^*)$), the value of the starting heuristic solution (z_H), the primal gap between this solution and the best solution found ($g_H[\%]$, calculated as $(z_H - z^*)/(z_H)$), and the percentage of edges (P), which is in the minimum spanning tree used as starting solution and also the best solution found. Column *per* gives the percentage of edges in the instance, and column *id* gives the ID of the instances (recall that for each combination of number of nodes and percentage of edges, ten instances were created).

Table 1 shows that our approach can solve instances with 10 nodes easily, the longest runtime is two seconds. Most of them can be solved already in the root node, however, for some instances there is quite a large root gap of up to 55%. For six of the 40 instances, the minimum spanning tree solution (MST) used as initialization is also the optimal solution. For some instances, only 44% of the edges of this MST are in the optimal solution, however, the solution quality of the MST for this instances is often still very good, for example for instance `0.60-5`, the primal gap is under 6%.

The instances with 15 nodes (Table 2) already become more difficult. Five of the 40 instances cannot be solved to optimality within the timelimit, all these five instances are complete graphs, thus the density of the instances may play a role in their difficulty. All the instances with 40% of edges of a complete graph are solved to optimality within 10 seconds. A reason for this could be that the separation becomes more

Table 1: Detailed results for setting bhif+ and the instances with 10 nodes

per	id	$t[s]$	LB	z^*	$g[\%]$	#BBn	$t_r[s]$	LB_r	$g_r[\%]$	z_H	$g_H[\%]$	P
0.40	1	0	2.34832	2.34832	0.00	0	0	2.34832	0.00	2.34832	0.00	1.00
0.40	2	0	1.82013	1.82013	0.00	0	0	1.82013	0.00	1.82013	0.00	1.00
0.40	3	0	2.36265	2.36265	0.00	0	0	2.36265	0.00	2.36803	0.23	0.89
0.40	4	1	3.12429	3.12429	0.00	24	0	2.10783	32.53	3.12429	0.00	1.00
0.40	5	0	1.94652	1.94652	0.00	3	0	1.77835	8.64	2.20777	11.83	0.89
0.40	6	0	2.54325	2.54325	0.00	0	0	2.54325	0.00	2.60153	2.24	0.67
0.40	7	1	4.11222	4.11222	0.00	189	1	1.81643	55.83	4.63938	11.36	0.78
0.40	8	0	2.02350	2.02350	0.00	0	0	2.02350	0.00	2.02350	0.00	1.00
0.40	9	0	2.94497	2.94497	0.00	24	0	1.74086	40.89	3.24482	9.24	0.78
0.40	10	0	2.63148	2.63148	0.00	0	0	2.63148	0.00	3.03134	13.19	0.78
0.60	1	1	3.04562	3.04562	0.00	336	0	1.91425	37.15	4.30049	29.18	0.67
0.60	2	1	3.00520	3.00520	0.00	158	0	1.75845	41.49	3.06337	1.90	0.89
0.60	3	1	3.28100	3.28100	0.00	259	1	1.99795	39.11	3.91895	16.28	0.56
0.60	4	0	2.34643	2.34643	0.00	1	0	2.34643	0.00	2.43503	3.64	0.89
0.60	5	1	2.81656	2.81656	0.00	4	1	2.38835	15.20	2.99377	5.92	0.44
0.60	6	0	2.81718	2.81718	0.00	0	0	2.81718	0.00	2.93219	3.92	0.67
0.60	7	1	2.77694	2.77694	0.00	13	0	2.57739	7.19	2.94739	5.78	0.67
0.60	8	0	2.47110	2.47110	0.00	0	0	2.47110	0.00	2.48188	0.43	0.78
0.60	9	0	2.22966	2.22966	0.00	0	0	2.22966	0.00	2.45688	9.25	0.56
0.60	10	1	2.50688	2.50688	0.00	3	1	2.43352	2.93	2.85075	12.06	0.67
0.80	1	1	2.87155	2.87155	0.00	607	0	1.72237	40.02	4.46661	35.71	0.67
0.80	2	1	2.81394	2.81394	0.00	0	1	2.81394	0.00	3.17401	11.34	0.89
0.80	3	1	2.55968	2.55968	0.00	3	1	2.55161	0.31	3.33238	23.19	0.67
0.80	4	0	2.20930	2.20930	0.00	0	0	2.20930	0.00	2.20930	0.00	1.00
0.80	5	1	2.20306	2.20306	0.00	532	0	1.53312	30.41	2.74632	19.78	0.44
0.80	6	1	2.58228	2.58228	0.00	134	1	1.85406	28.20	2.93700	12.08	0.78
0.80	7	1	2.31663	2.31663	0.00	753	1	1.64038	29.19	3.56665	35.05	0.67
0.80	8	1	2.04286	2.04286	0.00	0	1	2.04286	0.00	2.12304	3.78	0.67
0.80	9	1	2.36027	2.36027	0.00	0	1	2.36027	0.00	2.45045	3.68	0.56
0.80	10	0	2.14140	2.14140	0.00	0	0	2.14140	0.00	2.50277	14.44	0.78
1.00	1	1	2.72879	2.72879	0.00	351	1	1.61376	40.86	3.58893	23.97	0.56
1.00	2	1	2.11044	2.11044	0.00	0	1	2.11044	0.00	2.43338	13.27	0.89
1.00	3	2	2.61375	2.61375	0.00	869	1	1.72433	34.03	4.46504	41.46	0.56
1.00	4	0	1.70843	1.70843	0.00	0	0	1.70843	0.00	1.78039	4.04	0.78
1.00	5	1	2.19945	2.19945	0.00	207	1	1.52669	30.59	2.74632	19.91	0.56
1.00	6	1	2.68737	2.68737	0.00	38	1	1.95659	27.19	3.39304	20.80	0.78
1.00	7	2	2.18015	2.18015	0.00	423	1	1.70722	21.69	3.10123	29.70	0.67
1.00	8	1	2.13067	2.13067	0.00	0	1	2.13067	0.00	2.61161	18.42	0.56
1.00	9	0	1.89999	1.89999	0.00	0	0	1.89999	0.00	1.89999	0.00	1.00
1.00	10	1	1.89164	1.89164	0.00	435	1	1.66941	11.75	2.18449	13.41	0.78

burdensome for denser graphs, and also, there are more edges to branch when the graph is denser. Two of the instances are solved in the root node; interestingly, one of these two instances is a complete graph. The root gap is up to 67% (which happens for an instance with 40% edges), and on average is about 40-60%. For only one instance, the initial MST calculated is also the optimal solution; aside from this instance, the primal gap of the MST ranges from 2% to 36%.

Table 3 shows the results for instances with 20 nodes. For these instances, 26 of 40 remain unsolved within the timelimit. The optimality gap for the unsolved instances is up to 54%, while the rootgap is up to 63%. The primal gap is between about 9% and 44%. Similar to the instances with 15 nodes, the sparser instances seem somehow easier to solve: e.g., five out of the ten instances with $per = 0.40$ are solved, compared to only two out of ten with $per = 1.00$.

5. Conclusion

In this work, we presented two mixed-integer programming (MIP) approaches for the tree t^*-spanner problem, namely a compact formulation based on flows, and an exponential formulation based on a Benders-like decomposition. The Tr-t^*-Sp is an NP-hard problem with applications in the context of distributed systems, phylogeny and robotics control. To the best of our knowledge, these two MIP approaches are the first exact approaches proposed for this problem. Our computational study revealed that the decomposition approach outperformed the direct resolution of the compact formulation; however, the Tr-t^*-Sp seems quite

Table 2: Detailed results for setting bhif+ and the instances with 15 nodes

per	id	$t[s]$	LB	z^*	$g[\%]$	#BBn	$t_r[s]$	LB_r	$g_r[\%]$	z_H	$g_H[\%]$	P
0.40	1	5	3.03539	3.03539	0.00	448	2	1.89751	37.49	3.10057	2.10	0.71
0.40	2	3	2.86330	2.86330	0.00	692	1	1.96414	31.40	3.29547	13.11	0.64
0.40	3	4	2.27563	2.27563	0.00	45	2	1.90858	16.13	2.77903	18.11	0.79
0.40	4	3	2.77654	2.77654	0.00	998	1	1.70453	38.61	3.58791	22.61	0.71
0.40	5	6	3.61086	3.61086	0.00	5083	1	1.85333	48.67	4.77045	24.31	0.79
0.40	6	3	3.05302	3.05302	0.00	1452	0	1.00000	67.25	4.09753	25.49	0.57
0.40	7	3	3.11067	3.11067	0.00	368	1	1.95312	37.21	3.40647	8.68	0.79
0.40	8	9	3.69861	3.69861	0.00	12908	1	1.84519	50.11	3.99590	7.44	0.93
0.40	9	2	2.70706	2.70706	0.00	12	1	2.31796	14.37	2.82260	4.09	0.79
0.40	10	2	2.19823	2.19823	0.00	5	1	1.88988	14.03	2.73135	19.52	0.86
0.60	1	4	2.88617	2.88617	0.00	46	3	2.29712	20.41	2.98787	3.40	0.79
0.60	2	23	3.16700	3.16700	0.00	14914	2	1.72844	45.42	3.71760	14.81	0.50
0.60	3	5	2.65919	2.65919	0.00	1036	2	1.74095	34.53	3.27670	18.85	0.71
0.60	4	1	2.11537	2.11537	0.00	0	1	2.11537	0.00	2.11537	0.00	1.00
0.60	5	46	3.40065	3.40065	0.00	40980	2	1.77119	47.92	4.08750	16.80	0.64
0.60	6	31	3.07007	3.07007	0.00	19592	2	1.80867	41.09	3.56183	13.81	0.79
0.60	7	3	2.53812	2.53812	0.00	160	2	1.65912	34.63	3.09176	17.91	0.86
0.60	8	288	3.11862	3.11862	0.00	163027	3	1.68370	46.01	4.27364	27.03	0.64
0.60	9	307	3.23815	3.23816	0.00	205452	2	1.71227	47.12	4.45115	27.25	0.64
0.60	10	107	3.42128	3.42128	0.00	74655	2	1.70083	50.29	4.61586	25.88	0.79
0.80	1	15	2.71151	2.71151	0.00	11970	2	1.72970	36.21	2.89126	6.22	0.86
0.80	2	7	2.89696	2.89696	0.00	149	5	1.95548	32.50	3.19431	9.31	0.64
0.80	3	15	2.81212	2.81212	0.00	10039	3	1.68344	40.14	4.11664	31.69	0.57
0.80	4	7	2.74118	2.74118	0.00	769	5	1.84790	32.59	3.20465	14.46	0.71
0.80	5	185	3.38203	3.38203	0.00	102891	3	1.65437	51.08	5.27295	35.86	0.71
0.80	6	266	3.48709	3.48709	0.00	225547	3	1.65278	52.60	3.69105	5.53	0.71
0.80	7	7	2.32213	2.32213	0.00	2400	3	1.53249	34.00	2.56296	9.40	0.86
0.80	8	488	3.12071	3.12071	0.00	292759	2	1.58470	49.22	4.21839	26.02	0.79
0.80	9	340	3.02601	3.02601	0.00	122375	3	1.66720	44.90	3.46316	12.62	0.43
0.80	10	8	2.89465	2.89465	0.00	1867	4	1.90919	34.04	3.79320	23.69	0.71
1.00	1	8	2.71151	2.71151	0.00	1675	4	2.03291	25.03	2.89126	6.22	0.71
1.00	2	**TL**	2.91476	3.15763	7.69	307983	3	1.56269	50.51	4.10179	23.02	0.50
1.00	3	12	2.76445	2.76445	0.00	100	7	1.97515	28.55	3.21349	13.97	0.57
1.00	4	6	2.23030	2.23030	0.00	142	4	1.73646	22.14	2.92211	23.67	0.64
1.00	5	**TL**	3.03838	3.38203	10.16	216054	4	1.67687	50.42	5.27295	35.86	0.57
1.00	6	**TL**	2.88118	3.44192	16.29	297791	5	1.70193	50.55	4.15243	17.11	0.50
1.00	7	5	2.32213	2.32213	0.00	0	5	2.32213	0.00	3.02786	23.31	0.79
1.00	8	**TL**	2.85241	3.12071	8.60	186070	3	1.67976	46.17	4.21839	26.02	0.57
1.00	9	**TL**	2.64788	2.94563	10.11	105264	5	1.67563	43.11	4.10682	28.27	0.64
1.00	10	21	2.88175	2.88175	0.00	12648	4	1.77760	38.32	3.79320	24.03	0.57

challenging even for this approach. Thus, one avenue for further research could be the development of other exact approaches. Using Lagrangian relaxation could be interesting, as the problem would decompose in a spanning tree problem and shortest path problems. However, dealing with the min-max-objective of the problem could be challenging in a Lagrangian context. Complementary, since solving the problem exactly seems quite challenging, the development of further (meta-)heuristic algorithms could also be of interest. Moreover, extending the presented approaches to different versions of spanner-type problems, for example, where the required graph is not a spanning tree, but must be two-connected, could be a fruitful research topic.

Acknowledgements

E.A.-M. acknowledges the support of the Chilean Council of Scientific and Technological Research, CONICYT, through the grant FONDECYT N.1180670 and through the Complex Engineering Systems Institute (ICM-FIC:P-05-004-F, CONICYT:FB0816).

References

I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

Table 3: Detailed results for setting bhif+ and the instances with 20 nodes

per	id	$t[s]$	LB	z^*	$g[\%]$	#BBn	$t_r[s]$	LB_r	$g_r[\%]$	z_H	$g_H[\%]$	P
0.40	1	**TL**	3.67387	3.84104	4.35	75215	5	1.85527	51.70	4.96343	22.61	0.84
0.40	2	233	3.55172	3.55172	0.00	102342	5	1.56133	56.04	4.08967	13.15	0.63
0.40	3	**TL**	3.10255	3.48955	11.09	70697	7	1.73183	50.37	4.16383	16.19	0.58
0.40	4	**TL**	3.39189	4.52635	25.06	60044	4	1.92147	57.55	5.85445	22.69	0.63
0.40	5	220	3.40229	3.40229	0.00	56872	4	1.72387	49.33	5.12968	33.67	0.68
0.40	6	29	3.45974	3.45974	0.00	10530	5	1.82004	47.39	4.18656	17.36	0.79
0.40	7	56	3.01197	3.01197	0.00	19514	4	1.73818	42.29	3.75310	19.75	0.58
0.40	8	**TL**	2.54582	3.29292	22.69	35345	4	1.75562	46.69	4.88956	32.65	0.68
0.40	9	**TL**	2.97085	4.57595	35.08	55750	6	1.84828	59.61	6.02862	24.10	0.74
0.40	10	440	3.74582	3.74582	0.00	64021	8	1.96891	47.44	4.47730	16.34	0.68
0.60	1	**TL**	2.64357	3.48763	24.20	69390	6	1.69734	51.33	4.62272	24.55	0.79
0.60	2	**TL**	2.34768	3.80654	38.33	37510	7	1.71320	54.99	5.26496	27.70	0.63
0.60	3	590	3.14765	3.14765	0.00	254151	6	1.78752	43.21	4.73006	33.45	0.74
0.60	4	**TL**	2.72561	3.13703	13.11	102472	9	1.66181	47.03	4.29516	26.96	0.79
0.60	5	24	3.00848	3.00848	0.00	4099	5	1.65831	44.88	3.89301	22.72	0.68
0.60	6	92	2.88071	2.88071	0.00	23480	9	1.90658	33.82	3.21826	10.49	0.58
0.60	7	**TL**	2.82978	2.99422	5.49	42069	5	1.71850	42.61	3.36910	11.13	0.74
0.60	8	**TL**	2.88646	3.44112	16.12	86207	6	1.80973	47.41	4.25142	19.06	0.79
0.60	9	**TL**	2.97629	4.66189	36.16	42852	8	1.83199	60.70	6.07186	23.22	0.79
0.60	10	**TL**	2.56313	3.67551	30.26	47250	6	1.67115	54.53	4.75522	22.71	0.53
0.80	1	**TL**	2.60750	3.62166	28.00	81588	13	1.71128	52.75	4.74777	23.72	0.74
0.80	2	**TL**	2.29427	3.74122	38.68	27955	14	1.70037	54.55	4.61537	18.94	0.58
0.80	3	**TL**	2.49487	3.53616	29.45	53250	9	1.64624	53.45	4.79722	26.29	0.63
0.80	4	**TL**	2.72825	3.06612	11.02	98475	10	1.68614	45.01	4.23777	27.65	0.89
0.80	5	19	2.59987	2.59987	0.00	1029	11	1.83041	29.60	3.88195	33.03	0.68
0.80	6	175	2.73395	2.73395	0.00	32033	11	1.77808	34.96	3.95446	30.86	0.63
0.80	7	46	2.78614	2.78614	0.00	8606	10	1.72065	38.24	4.63660	39.91	0.47
0.80	8	**TL**	2.51333	3.48728	27.93	32791	9	1.71864	50.72	6.31373	44.77	0.47
0.80	9	**TL**	2.46503	4.41167	44.12	29350	9	1.69952	61.48	4.84227	8.89	0.74
0.80	10	**TL**	2.75116	3.99696	31.17	31500	13	1.74380	56.37	4.73173	15.53	0.79
1.00	1	**TL**	2.07473	4.41855	53.05	21867	19	1.63925	62.90	5.52401	20.01	0.68
1.00	2	**TL**	2.25680	3.64352	38.06	31866	15	1.60042	56.07	4.61537	21.06	0.63
1.00	3	**TL**	2.28795	3.85107	40.59	24863	13	1.66185	56.85	5.43034	29.08	0.53
1.00	4	**TL**	1.95906	3.96484	50.59	31154	11	1.56269	60.59	6.26683	36.73	0.74
1.00	5	39	2.57100	2.57100	0.00	2356	15	1.68510	34.46	3.15894	18.61	0.68
1.00	6	87	2.61271	2.61271	0.00	15046	12	1.65494	36.66	4.59690	43.16	0.53
1.00	7	**TL**	2.16276	2.91878	25.90	60056	11	1.56182	46.49	4.84962	39.81	0.58
1.00	8	**TL**	2.07683	4.17430	50.25	19547	11	1.63610	60.81	5.22911	20.17	0.58
1.00	9	**TL**	2.26355	3.70360	38.88	29986	17	1.65708	55.26	5.38744	31.25	0.74
1.00	10	**TL**	2.31879	3.89946	40.54	23479	14	1.59881	59.00	4.73173	17.59	0.68

E. Álvarez-Miranda, E. Fernández, and I. Ljubić. The recoverable robust facility location problem. *Transportation Research Part B: Methodological*, 79:93 – 120, 2015.

B. Awerbuch. Complexity of network synchronization. *Journal of the Association for Computing Machinery,*, 32(4):804–823, 1985.

B. Awerbuch, A. Baratz, and D. Peleg. Effcient broadcast and light-weight spanners. *Weizmann Institute of Science*, Technical Report N. CS92-22, 1992.

H. Bandelt and A. Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Advances in Applied Mathematics*, 7(3):309–343, 1986.

K. Bharath-Kumar and J. Jaffe. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications*, 31(3):343–351, 1983.

S. Bhatt, F. Chung, T. Leighton, and A. Rosenberg. Optimal simulations of tree machines. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 274–282, 1986.

L. Cai and D. Corneil. Tree spanners. *SIAM Journal on Discrete Mathematics*, 8(3):359–387, 1995.

L. Cai and J. Mark Keil. Computing visibility information in an inaccurate simple polygon. *International Journal of Computational Geometry & Applications*, 7(6):515–537, 1997.

B. Cherkassky and A. Goldberg. On implementing push-relabel method for the maximum flow problem. In E. Balas and J. Clausen, editors, *Proceedings of IPCO IV*, volume 920 of *LNCS*, pages 157–171. Springer, 1995.

A. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429–1450, 2005.

M. Elkin, Y. Emek, D. Spielman, and S. Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 38(2):608–628, 2008.

Y. Emek and D. Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM Journal on Computing*, 38(5):1761–1781, 2009.

M. Fischetti, M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017a.

M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning benders decomposition for large-scale facility location. *Management Science*, 63(7):2146–2162, 2017b.

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

A. Liestman and T. Shermer. Additive graph spanners. *Networks*, 23(4):343–363, 1993.

T. Magnanti and L. A. Wolsey. Optimal trees. *Handbooks in operations research and management science*, 7:503–615, 1995.

T. Magnanti, P. Mireault, and R. Wong. Tailoring benders decomposition for uncapacitated network design. In G. Gallo and C. Sandi, editors, *Netflow at Pisa*, volume 26 of *Mathematical Programming Studies*, pages 112–154. Springer Berlin Heidelberg, 1986.

J. Marble and K. Bekris. *Asymptotically Near-Optimal Is Good Enough for Motion Planning*, pages 419–436. 2017.

D. Peleg and J. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18(4): 740–747, 1989.

D. Peleg and E. Upfal. A tradeoff between space and efficiency for routing tables. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 43–52. ACM, 1988.

R. C. Prim. Shortest connection networks and some generalizations. *Bell Labs Technical Journal*, 36(6): 1389–1401, 1957.

K. Singh and S. Sundar. Artifical bee colony algorithm using problem-specific neighborhood strategies for the tree t-spanner problem. *Applied Soft Computing*, 62:110–118, 2018.

G. Venkatesan, U. Rotics, M. Madanlal, J. Makowsky, and C. Pandu Rangan. Restrictions of minimum spanner problems. *Information and Computation*, 136(2):143–164, 1997.

R. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28(3):271–287, 1984.