IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2021, accepted October 12, 2021, date of publication October 21, 2021, date of current version October 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3122024

Determining the Minimum Cost Steiner Tree for Delay Constrained Problems

LÚCIA MARTINS[1,2], DORABELLA SANTOS[2], TERESA GOMES[1,2], (Member, IEEE), AND RITA GIRÃO-SILVA[1,2]

[1]Department of Electrical and Computer Engineering, University of Coimbra, 3030-290 Coimbra, Portugal
[2]Institute for Systems Engineering and Computers at Coimbra, 3030-290 Coimbra, Portugal

Corresponding author: Lúcia Martins (lucia@deec.uc.pt)

This work was supported in part by European Regional Development Fund (ERDF) through the Centre's Regional Operational Program, and in part by the National Funds through Fundação para a Ciência e Tecnologia (FCT) under Project CENTRO-01-0145-FEDER-029312.

ABSTRACT We address a variant of the Steiner tree problem for delay constrained problems. The addressed problem consists in determining the minimum cost Steiner tree, while guaranteeing that the delay between any two terminal nodes does not exceed a given maximum value. This problem is known as the bounded diameter Steiner minimum tree problem. We propose a compact formulation based on integer linear programming (ILP) to obtain optimal solutions, which was efficiently solved on two telecommunication core networks up to 75 nodes. However, given that for traditional Steiner tree graphs the ILP proved to be inefficient, we propose a heuristic method and compare it with the ILP formulation. We show that the heuristic provides optimal solutions, except for two cases in our experiments where it provided near-optimal solutions, always in reasonable runtimes. Additionally, to reduce the complexity of the problem, we propose some novel and modified graph reductions specific for the addressed problem.

INDEX TERMS Delay-constrained, graph reductions, heuristic, integer linear programming, Steiner tree problem.

I. INTRODUCTION

The Steiner tree problem (STP) is one of the fundamental combinatorial optimization problems [1]. Consider a connected undirected graph, where each edge has an associated positive cost, which may represent the length of the edge or a delay incurred by using that edge, for instance. Given a set of terminal nodes in the network, the STP is the problem of finding a minimum cost tree that contains all the terminal nodes, possibly using additional nodes in the network, the so-called Steiner nodes.

This problem has been shown to be $\mathcal{NP}$-complete for general graphs [1]. Two well-known special cases of this problem may be considered: (i) if there are only two terminal nodes, then the STP reduces to the shortest path problem; (ii) if all nodes are terminal, then the STP reduces to the spanning tree problem. Both these problems can be solved exactly in polynomial time, for instance, by Dijkstra's or by Prim's algorithm, respectively.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenzhou Tang.

The STP arises in many network design problems, such as facility location and reliability problems. For a recent overview on applications of the STP in graphs and resolution approaches, see [2].

Steiner tree problems arise in many communication network problems [3]. An important application of Steiner trees is in multicast communications, which are very important in today's networks. In a many-to-many distribution perspective, different nodes contribute and receive information, and their interconnection may be established through other network nodes. The information exchange should take place within a certain delay limit, i.e., it is important to guarantee that the delay between the nodes exchanging information does not exceed a specified bound. This is the problem we focus on in this work: the STP with pairwise delay constraints, i.e. the aim is to find the minimum cost Steiner tree, while additionally guaranteeing that the delay between each pair of terminal nodes does not exceed a given maximum value. This is the bounded diameter Steiner minimum tree (BDSMT) problem, as described in [4]. The delay between any two nodes is given by the sum of the delays of the links belonging to the path connecting them.

Delay constrained STPs are a variant of the classical STP which have been tackled in multiple ways in the literature, with different formulations for the delay bounds. One such problem is the bounded radius Steiner minimum tree (BRSMT) problem, as described in [4], in which the STP is formulated with the additional constraint of guaranteeing that the delay between a given source (or root) node (one terminal node selected in advance) and any other terminal node cannot exceed a given value. This problem fits into multicast communications in a one-to-many distribution perspective. Possible variants of the BDSMT and the BRSMT consist in considering that the delay in each edge is unitary, which is equivalent to having hop constraints [5]–[7] rather than delay constraints. All these problems are generalizations of the classical STP and are, therefore, also $\mathcal{NP}$-complete.

Other applications of Steiner trees include energy saving in networks, as presented in [8], [9]. In [8], a Steiner tree is defined, considering the inverse of the available bandwidth as the link cost. A network element may be put into sleep mode if no traffic passes through it. The final goal is to minimize the number of active elements for routing. To achieve this goal, if there are very long paths in the tree, bypass paths can be found to modify the tree to have shorter routes on the tree. As in our case, there is the need to avoid excessively long routes on the tree, which we accomplish by constraining the tree diameter. In [9], a Steiner tree is defined to minimize the power consumption in the nodes for wireless networks. The power of a node is given by the maximum cost of the edges incident to it, and in turn the edge costs define the power to connect their endpoints via bidirectional links. These problems are $\mathcal{NP}$-hard, even for the spanning tree case.

Another application of a delay constrained STP is in the Software Defined Networking (SDN) controller synchronization problem, as stated in [10]. In this problem, the number of controllers is known in advance, but not their location. The problem aims at finding the appropriate placement of the SDN controllers and assigning the switches to the controllers that manage them. Different constraints are considered, namely maximal latency constraints between controllers and between the switches and the controllers that manage them. The objective function is to minimize the total cost of the Steiner tree (sum of the inter-controller delays), as it represents the inter-controller network state synchronization cost. This problem is formulated in a framework that takes into account the possible existence of single link failures and the subsequent need for recovery.

Another work related to SDN is [11], where reliable multicast routing in SDN is addressed. The work considers the Steiner tree problem, where besides the destination nodes, a candidate set of recovery nodes is also considered. The reliable Steiner tree needs to span both destination and recovery intermediate nodes, while minimizing the total tree cost. The recovery nodes are selected in such a way, that if the connection from the source node to a destination node fails, then the connection is recovered in one of the recovery nodes and restored to the affected destination node.

In the computational results, we consider the maximal acceptable delay between any two nodes given as a percentage of the graph diameter, where the graph diameter is the longest shortest path between any two nodes in the graph. This arises in problems such as the controller placement problem in SDN networks [12], [13], where the maximum value for the delays depends on how wide the graph is. The STP variant addressed here is directly applicable in this context. Moreover, we consider, as in [10], that the objective function is to minimize the total cost of the Steiner tree given by the sum of inter-controller delays.

We took advantage of considering the delay as the Steiner tree cost to develop a simple but effective heuristic for this variant of the BDSMT problem.

The contributions of our paper are the following:

1) We propose an integer linear programming (ILP) model, which is an extension to that of [14], to guarantee that the delay between any two terminal nodes does not exceed a maximum value.
2) We propose a heuristic algorithm to find good quality solutions in reasonable time.
3) We propose two novel and two modified graph reductions for the STP problem addressed.
4) We provide computational results that compare the heuristic solutions to those obtained with the ILP model to assess the quality of the heuristic, and show that the heuristic is capable of obtaining (near-)optimal solutions, in optical core network topologies and classic graphs for the STP.

The paper is organized as follows. In the next section, some related work is presented. In Section III, we present the compact formulation for the STP with delay constraints, as an ILP model. In Section IV, we propose a heuristic method to obtain good quality solutions for our problem, within reasonable time. In Section V, we propose and present graph reductions which allow to obtain simplified graphs for the Steiner tree problem variant addressed herein. In Section VI, we report the computational results and in Section VII, we draw the conclusions.

II. RELATED WORK

Many references where multicast routing is tackled by using a Steiner tree approach are cited in [2]. Here we emphasize some contributions in this topic, in particular regarding the BRSMT problem which is more common. In [15], a source-based routing algorithm is considered. The proposed heuristic starts by considering a closure graph, which is a fully meshed graph formed by edges with a cost equal to the constrained shortest path between every node pair in the original network. The heuristic proceeds by building a Steiner tree in this graph using a greedy approach, in which edges are added according to some specific selection functions. In [16], a distributed algorithm is proposed to find a minimum cost Steiner tree that is both delay constrained (in the sense of the BRSMT problem) and bandwidth constrained. When there is a call request, a tree must be obtained such that its links can satisfy

the bandwidth requirements of that call. In [17], the BRSMT problem is also tackled. The resolution approach starts by calculating a minimum delay tree, and in subsequent iterations, the cost of the tree is improved while keeping the tree feasible with regard to the delay constraint. In [14], a heuristic algorithm based on the Takahashi and Matsuyama [18] heuristic is considered, with adaptations similar to the ones proposed in [6]. The construction of the constrained tree starts with the delay constrained shortest path between the root and the most distant terminal node, and follows by successively inserting terminal nodes such that minimal constrained shortest paths are appended to the existing tree. In [19], a heuristic based on Particle Swarm Optimization (PSO) is proposed to solve the BRSMT problem. In these references, cost and delay are different metrics and are not used interchangeably, contrarily to our work. Other relevant papers are [20], [21], in which the purpose is finding a tree that not only satisfies a delay constraint between a root and all the terminal nodes, but also a constraint on the interdestination delay, i.e. a limit is imposed to the maximum difference between path delays from the root to any two terminal nodes. In particular, in [21] a problem regarding a more complex framework of virtual network embedding for cloud data centers is addressed. The delay-sensitive multicast problem is considered on the virtual layer with Quality of Service (QoS) constraints which translate into source-destination delay constraints and delay variation constraints. An ILP problem is formulated and two resolution approaches are considered, one using a 3-step strategy, and another using a Tabu search strategy. In [22], multiple QoS constraints are considered when devising a QoS multicast Steiner tree, including constraints in terms of bandwidth, delay (in the sense considered in the BRSMT problem), loss rate and jitter. The problem is solved by a hybridized bacteria foraging and PSO algorithmic approach.

As for the BDSMT problem, it is addressed in [4] using four heuristics. Their solution strategies involve first choosing the Steiner nodes using some heuristic, and then finding the Steiner tree (spanning tree) in the induced graph. The performance of the heuristics is evaluated by their success rate (finding a solution satisfying the diameter constraint), but their quality is not discussed.

The BDSMT problem is designated as the constrained diameter Steiner tree (CDST) problem in [23]. The heuristic in [23] for the BDSMT problem uses a new metric designated the *delta diameter* for each edge, such that its sum over the edges of the tree is related to the tree delay diameter. However, the value of the delta diameter on the edges will depend on the order they are added to the tree under construction. The solutions obtained for the BDSMT problem were evaluated for small instances, against the exact solution obtained using an exhaustive approach.

In [24], a variant of the BDSMT is considered. All the terminal nodes must appear as leaves on the tree and the problem is known as the terminal Steiner tree problem. This is a special case for the BDSMT, since in general some terminal nodes can appear as intermediate nodes in the Steiner tree. The problem is solved using a heuristic method with an interplay between the cost metric and the tree diameter metric. In [25], the same authors deal with the opposite problem of the minimum diameter cost-constrained Steiner tree, which is also $\mathcal{NP}$-hard. In this work, the terminal nodes must also appear as leaves of the tree.

Usually, the approaches dealing with the multicast routing problem formulated as a STP involve the use of heuristics. This is due to two main reasons: (i) the formulated problems are $\mathcal{NP}$-complete and therefore heuristic algorithms are more adequate to solve them; (ii) these are real time applications and therefore being able to find appropriate (eventually sub-optimal) solutions in a reasonable time is of the utmost importance.

III. ILP MODEL FOR THE STP WITH DELAY CONSTRAINTS

We propose a compact formulation for the delay constrained STP, in particular the BDSMT problem, as an ILP model which is an extension of the model presented in [14]. The model is extended in order to guarantee that the delay of the path connecting *any* two terminal nodes on the Steiner tree is bounded by $\mathcal{D}$. In [14], [23] the cost of the Steiner tree is unrelated to the delay. In our formulation the link cost is also the delay. In fact longer links tend to be more expensive, and according to [10], when terminal nodes are controllers, this cost represents the SDN network synchronization cost.

Consider an undirected graph $G = (N, E)$, where N is the set of nodes and E is the set of edges. The set of terminal nodes is $T \subseteq N$ and the remaining nodes (i.e., in $N \setminus T$) are the Steiner nodes. Each edge is denoted by its end nodes $\{i, j\}$ and has an associated delay $d_{ij} > 0$. For each node $i \in N$, the set of neighbouring or adjacent nodes to i is designated by $V(i)$. Also consider the set A as the set of arcs, where arc (i, j) represents the directed link from i to j.

We aim to find the minimum cost Steiner tree connecting the nodes in a given set of terminal nodes $T \subseteq N$, such that the delay between any two nodes in T is at most a given value $\mathcal{D}$. We assume that the delay of a path is the sum of the delay of its links [12]. Consider the following decision variables:

- y_{ij} binary variable that is 1 if link $\{i, j\} \in E$ belongs to the Steiner tree, and 0 otherwise
- x_{ij}^{tk} binary variable that is 1 if arc $(i, j) \in A$ belongs to the path from node $t \in T$ to node $k \in T \setminus \{t\}$ routed on the Steiner tree, and 0 otherwise
- u_i binary variable that is 1 if node $i \in N$ is either a terminal or a Steiner node, and 0 otherwise

The ILP model for the addressed STP problem is given by:

$$\min \sum_{\{i,j\} \in E} d_{ij} y_{ij} \tag{1}$$

$$\text{subject to} \sum_{j \in V(i)} \left(x_{ij}^{tk} - x_{ji}^{tk} \right) = \begin{cases} 1 & i = t \\ -1 & i = k \\ 0 & i \neq t, k \end{cases} \quad t \in T,\ k \in T \setminus \{t\},\ i \in N \tag{2}$$

$$x_{ij}^{tk} + x_{ji}^{tk} \leq y_{ij} \quad t \in T,\ k \in T \setminus \{t\},\ \{i, j\} \in E \tag{3}$$

$$y_{ij} \leq \sum_{t \in T} \sum_{k \in T \setminus \{t\}} \left(x_{ij}^{tk} + x_{ji}^{tk} \right) \quad \{i,j\} \in E \tag{4}$$

$$\sum_{\{i,j\} \in E} d_{ij} \left(x_{ij}^{tk} + x_{ji}^{tk} \right) \leq \mathcal{D} \quad t \in T,\ k \in T \setminus \{t\} \tag{5}$$

$$y_{ij} \leq u_i \quad \{i,j\} \in E \tag{6}$$

$$y_{ij} \leq u_j \quad \{i,j\} \in E \tag{7}$$

$$u_i \leq \sum_{j \in V(i)} y_{ij} \quad i \in N \tag{8}$$

$$\sum_{\{i,j\} \in E} y_{ij} \leq \sum_{i \in N} u_i - 1 \tag{9}$$

$$u_i \in \{0,1\} \quad i \in N \tag{10}$$

$$y_{ij} \in \{0,1\} \quad \{i,j\} \in E \tag{11}$$

$$x_{ij}^{tk} \in \{0,1\} \quad t \in T,\ k \in T \setminus \{t\},\ (i,j) \in A \tag{12}$$

The objective function (1) aims to minimize the cost (which in our case represents the delay) of the Steiner tree, and is given as the sum of the delays of the links belonging to the Steiner tree.

Constraints (2) are the flow conservation constraints, guaranteeing that for each terminal node t, there is a path going from t to all the other terminal nodes.

Constraints (3) and (4) guarantee the correct assignment of variables y_{ij}, that account for the links belonging to the Steiner tree, by making use of variables x_{ij}^{tk}: constraints (3) guarantee that each path defined in (2) can only transverse links belonging to the Steiner tree, whereas constraints (4) guarantee that a link belongs to the Steiner tree if it belongs to one of the paths defined in (2).

Constraints (5) guarantee that the delays of the paths connecting any two terminal nodes and routed on the Steiner tree do not exceed $\mathcal{D}$.

Constraints (6)-(8) relate variables y_{ij} with variables u_i, that account for all the nodes belonging to the Steiner tree, both terminal nodes and Steiner nodes: constraints (6) and (7) guarantee that if link $\{i,j\}$ belongs to the Steiner tree, then so must its end nodes i and j, whereas constraints (8) guarantee that if a node i belongs to a Steiner tree then it must be an end node of a link on the Steiner tree.

Constraint (9) guarantees that the selected set of links connecting the selected set of nodes necessarily forms a tree.

Finally, constraints (10)-(12) are the variable domain constraints.

This ILP model extends the one in [14], by considering that each terminal node is a source node. In this way, all terminal nodes are connected via a tree to each terminal node serving as the source node. Moreover, the delay between each terminal node and each source node is guaranteed to be at most $\mathcal{D}$. This is achieved by constraints (2)-(5), which are an extended version of the ones in [14].

The set of trees forms, among them, the final Steiner tree. This is achieved by the additional constraints (6)-(9) (not present in the model of [14]). The objective function ensures the final tree is the minimum cost Steiner tree that guarantees that the delay between any pair of terminal nodes is at most $\mathcal{D}$.

We solve the ILP model using the commercial optimization solver CPLEX 12.6 [26]. For bi-connected topologies, which are typical of optical core networks, the ILP model performs efficiently for the topologies we tested up to 75 nodes. However, for the classic topologies used for the STP, the ILP model showed to be computationally challenging.

IV. HEURISTIC METHOD

We propose a heuristic method as an alternative to the ILP model, to obtain good quality solutions within reasonable time, inspired on the well-known Takahashi and Matsuyama heuristic [18] for the STP. This heuristic starts the construction of the tree by selecting a terminal node and then connecting it to its closest terminal node. The next terminal node to be added is the one that is closest to the nodes already connected and so forth until all terminal nodes are connected.

The following approaches were followed to add a terminal node to the sub-tree. In the first approach the node connected in each iteration is the one for which the largest distance from it to each of the terminal nodes in the sub-tree already obtained is the smallest one. However, when we try to obtain a Steiner tree using only this approach we can get longer trees with higher delays between the terminal nodes than if we simply use the approach presented in [18] where the node connected in each iteration is the closest one (second approach). So the proposed strategy is to compute the first part of the tree using the second approach and afterwards, to compute the rest of the tree, using the first approach. This is achieved picking the parameter d randomly in a given set of values (in line 3) and with the *if-else* in lines 16 and 24 of Algorithm 1, respectively. Therefore if d is a small value, the tree is mainly computed through the first approach (lines 25-31) which in general leads to a higher number of trees respecting the delay constraint but with higher cost. On the other hand, if d is a higher value then the tree is mainly computed through the second approach (lines 17-23) and most of the trees do not respect the delay constraint but the ones that do respect have in general a lower cost. So in Algorithm 1, d varies in the set $\left\{ \left\lfloor \frac{3}{4}|T| \right\rfloor, \ldots, |T| \right\}$.

In Algorithm 1, $Dijkstra(s', a)$ finds the shortest path p and the cost c of this path from the node s' to the node a in the tree taking into account that the edges already in the tree have infinite distance (line 12). The function $MaxDistance(s', a, arv)$ computes the maximum distance between s' and each terminal node in the tree arv passing through a (line 25). The total cost of the tree is computed by the function $Cost\_Diameter(arv)$ and is the summation of the lengths of its links (line 36). All the trees for which $MaxDistance(s', a, arv) \leq \mathcal{D}$ are stored in $\mathcal{S}$ (line 35). The final tree Arv is the one with minimum cost and minimum diameter in $\mathcal{S}$. Every time that a tree is found for which the diameter, i.e., the maximum distance between a given terminal node s' and each terminal node in the tree arv is greater than $\mathcal{D}$, the algorithm aborts the computation of the tree arv and starts the computation of a new tree through

a new terminal node s (lines 14 and 15 of Algorithm 1). In line 32 a new path is added to the tree *arv* without repeating one of the path's end-nodes (the one already in *arv*).

The cost of the path between the node s' and node a in *arv* is evaluated in line 17, and the one with minimum cost is stored as well as node s' (line 20). If there is more than one path with minimum cost, the one with the minimum of the maximum distance between the node s' and each terminal node in the tree is selected (line 23).

Analogously, the maximum distance between the node s' and each terminal node is evaluated in line 25, and the one with minimum value is stored as well as node s' (line 28). If there is more than one path with minimum value, the one with the minimum cost is selected (line 31).

Our proposed heuristic, although inspired by the Takahashi and Matsuyama heuristic [18], has two significant differences from the latter which are the introduction of randomness and the use of a wider set of paths, than solely the shortest paths. This allows to obtain solutions that otherwise would not be found because it can diversify much more the search in the solutions space obtaining better solutions in less time. The basic idea is that each node to be added to the tree *arv* is picked randomly among those that are not yet in the tree (function $PickRandomly(T')$, line 10). Because of this, the path that connects the new node may contain other terminal nodes not yet in the tree, and so function $finds\_nodes(p, T')$ (line 33 of Algorithm 1) finds other terminal nodes in the path p different from s'. This is an important feature of the algorithm to improve its efficiency as it reduces the number of times that the while cycle (line 7 of Algorithm 1) needs to be executed.

To improve the cost of a Steiner tree, it was shown in [27] that it is better to try every terminal node to start the construction of the tree. This strategy is effective also for the present problem however it proved to be an even better strategy to pick randomly a terminal node from the set T (line 4 in Algorithm 1) and do it for a given number of times $\theta|T|$, where θ is a small constant that can be tuned for each problem and $|T|$ is the number of terminal nodes (line 2 in Algorithm 1). As this strategy also diversifies the search space it allows to obtain the same solutions in a lower number of iterations as compared with the one that picks sequentially all terminal nodes from the set T as in this latter case most of the times we get the same solutions.

Being $|N|$ the total number of nodes in the graph, the complexity of Dijkstra's algorithm is $\mathcal{O}(|N|^2)$ (which can be reduced recurring to a binary heap, for instance). Note that it is necessary to compute Dijkstra's algorithm many times in this algorithm because we need to consider that the links in *arv* have infinite cost, so the complexity of the core algorithm is $\mathcal{O}(|T|^2 \cdot |N|^2)$ instead of $\mathcal{O}(|T| \cdot |N|^2)$ of Takahashi and Matsuyama heuristic [18] because this classic algorithm can compute all the necessary shortest paths in advance only once as it only uses them to compute each tree. Considering the outer cycle, the total complexity of Algorithm 1 is thus $\mathcal{O}(|T|^3 \cdot |N|^2)$. Note that this is an upper bound, as in most cases a large number of trees is discarded (about halfway into their construction on average).

Algorithm 1: Minimum Delay Tree Algorithm

```
   input : G(N, E), T, D, θ
   output: Arv
1  S ← ∅ // set of trees
2  for i ← 1 to θ|T| do
3      d ← PickRandomly({⌊(3/4)|T|⌋, ..., |T|})
4      s ← PickRandomly(T)
5      arv ← {s}
6      T' ← T \ {s}
7      while T' ≠ ∅ do
8          MinCost ← ∞, MaxDist ← ∞
9          NewPath ← ∅
10         s' ← PickRandomly(T')
11         forall a ∈ arv do
12             (p, c) ← Dijkstra(s', a)
13             Dist ← MaxDistance(s', a, arv)
14             if Dist > D then
15                 abort arv and continue with other s ∈ T
                       // goes to line 2
16             if |T \ T'| < d then
17                 if c < MinCost then
18                     MinCost ← c
19                     MaxDist ← Dist
20                     NewPath ← p, s_new ← s'
21                 if c = MinCost ∧ MaxDist > Dist then
22                     MaxDist ← Dist
23                     NewPath ← p, s_new ← s'
24             else
25                 if Dist < MaxDist then
26                     MaxDist ← Dist
27                     MinCost ← c
28                     NewPath ← p, s_new ← s'
29                 if MaxDist = Dist ∧ c < MinCost then
30                     MinCost ← c
31                     NewPath = p, s_new ← s'
32         arv ← arv ⊕ NewPath
33         T'' ← finds_nodes(NewPath, T')
34         T' ← T' \ (T'' ∪ {s_new})
35     S ← S ∪ {arv}
36 Arv ← arg min_{arv∈S} Cost_Diameter(arv)
```

To assess the quality of the solutions obtained with the heuristic, we compare in section VI these solutions to those obtained with the ILP model (when the ILP model can still deliver them).

V. GRAPH REDUCTIONS

The complexity associated to the Steiner tree problem often leads the resolution of this problem to be in the reduced

graph obtained through the use of local properties of graph G. Sometimes the minimum cost tree can even be found through the reduction process (as observed in an example for the performed experiments).

In this work, we propose two novel reductions: Reduction 1b) and Reduction 5. We also adapted two reductions from the classical STP to integrate the delay constraints: Reduction 4a) and Reduction 4b). Finally, some classical reductions were also used: Reduction 1a), Reduction 2 and Reduction 3. Further details on reductions for the classical STP are presented in [28].

Reduction 1: Steiner nodes can be removed from G in the following cases:

(a) All the Steiner nodes with degree 1 can be removed from G as well as the respective incident edge.

(b) The Steiner nodes for which the longest shortest path to the terminal nodes plus the smallest shortest path (in terms of delay) is greater than the maximum delay can also be removed together with their incident edges.

Proof: Assume that $s \in N$ is a Steiner node.

(a) If $deg(s) = 1$, where $deg(i)$ denotes the degree of node i, then it cannot be an intermediate node in the Steiner tree and so could never be a Steiner node. Henceforth, it can be removed from G.

(b) Now assume that $deg(s) \geq 2$. Consider

$$\hat{d}_{\min} = \min_{t \in T} \hat{d}_{st} \quad \hat{d}_{\max} = \max_{t \in T} \hat{d}_{st} \quad t_M = \arg\max_{t \in T} \hat{d}_{st}$$

where $\hat{d}_{st}$ is the delay of the path from s to t on the Steiner tree. Assume that $\hat{d}_{\min} + \hat{d}_{\max} > \mathcal{D}$. Since s is a Steiner node, then there exists $t' \in T$ such that s belongs to the path from t' to t_M on the Steiner tree. Hence,

$$\hat{d}_{t't_M} = \hat{d}_{t's} + \hat{d}_{st_M} = \hat{d}_{t's} + \hat{d}_{\max} \geq \hat{d}_{\min} + \hat{d}_{\max} > \mathcal{D}$$

which is not possible since t' and t_M are terminal nodes and so must obey $\hat{d}_{t't_M} \leq \mathcal{D}$. Therefore, if $\hat{d}_{\min} + \hat{d}_{\max} > \mathcal{D}$, s cannot be a Steiner node and so it can be removed from G (together with its incident edges).

□

Reduction 2: All the Steiner nodes k with degree 2 can be removed from G. The two edges $\{i, k\}$ and $\{k, j\}$ are replaced by the edge $\{i, j\}$ with delay $d_{ij} = d_{ik} + d_{kj}$. If parallel edges appear in this operation, the one with the larger delay can be removed.

Reduction 3: All the edges $\{i, j\}$ such that $d_{ij} \geq p_{ij}$ (being p_{ij} the delay of the shortest path between i and j) can be removed from G.

Proof: Note that reductions 2 and 3 are straightforward applications of the classical ones. □

Reduction 4: The terminal node i with degree 1 can be removed from G as well as the respective incident arc in the following cases:

(a) If node i connects to a node k where $deg(k) = 2$, node i can be removed and then k becomes a terminal node (if it is not already). Assume node l is the other node that k connects to. Then, as node i is removed, $d_{lk} \leftarrow d_{lk} + d_{ki}$ to account for the contracted edge $\{i, k\}$ and $T \leftarrow T \cup \{k\} \setminus \{i\}$. Node i and edge $\{i, k\}$ will belong to the optimal Steiner tree.

(b) If node i connects to a node k with $deg(k) \geq 2$ and k is connected to a terminal node n with $deg(n) = 1$ such that $d_{kn} \geq d_{ki}$, then node i and edge $\{i, k\}$ can be removed. In this case $T \leftarrow T \setminus \{i\}$ and the node i and the edge $\{i, k\}$ will belong to the optimal Steiner tree.

Proof:

(a) Since node i is terminal and node k connects only to i and l, then i can be contracted into k, where k becomes the terminal node and d_{ik} is summed to d_{lk}. In the final optimal Steiner tree, i is added via edge $\{i, k\}$.

(b) Two terminal nodes i and n with degree 1 connect to a node k, where $d_{ki} \leq d_{kn}$. Then, node i can be removed in the reduced graph, since it will be added in the final tree via edge $\{i, k\}$, but is not crucial to guarantee the constrained delay in the tree, since d_{kn} is greater.

□

Reduction 5: Assume $i \in T$ and j is adjacent to i. If there exists $z \in T \setminus \{i; j\}$ such that $d_{ij} + \min\{p_{iz}, p_{jz}\} > \mathcal{D}$, then edge $\{i, j\}$ cannot be on the Steiner tree and so it can be removed from G.

Proof: Assume that $\{i, j\}$ belongs to the Steiner tree. Then there are at least two terminal nodes such that $\{i, j\}$ belongs to the path between them (over the tree).

By removing $\{i, j\}$ from the tree, we are left with two subtrees ST_1 and ST_2 such that ST_1 includes i and possibly other terminal nodes, and ST_2 includes j and possibly other terminal nodes.

Let $z \in T \setminus \{i; j\}$ be a node such that $d_{ij} + \min\{p_{iz}, p_{jz}\} > \mathcal{D}$. The following situations may occur:

(i) $z \in ST_1$:

(a) Node j is a Steiner node and so there exists $w \in T \cap ST_2$. We have that:

$$\begin{aligned}\hat{d}_{zw} &= \hat{d}_{zi} + d_{ij} + \hat{d}_{jw} \geq p_{zi} + d_{ij} + p_{jw} \\ &> p_{zi} + d_{ij} \geq \min\{p_{zi}, p_{zj}\} + d_{ij} > \mathcal{D}\end{aligned}$$

(b) Node j is a terminal node. We have that:

$$\begin{aligned}\hat{d}_{zj} &= \hat{d}_{zi} + d_{ij} \geq p_{zi} + d_{ij} \\ &\geq \min\{p_{zi}, p_{zj}\} + d_{ij} > \mathcal{D}\end{aligned}$$

(ii) $z \in ST_2$. We have that:

$$\begin{aligned}\hat{d}_{iz} &= d_{ij} + \hat{d}_{jz} \geq d_{ij} + p_{jz} \\ &\geq d_{ij} + \min\{p_{iz}, p_{jz}\} > \mathcal{D}\end{aligned}$$

In all the above situations, it is shown that there is a pair of terminal nodes such that the delay is greater than $\mathcal{D}$. This results from the assumption that $\{i, j\}$ is in the Steiner tree. We conclude that edge $\{i, j\}$ cannot be in the Steiner tree and therefore it can be removed from G. □

The reductions are applied sequentially, and for each iteration as long as one of them is applied, the process continues. The order of the applied reductions is the following:

- Reduction 1;
- Reduction 3;
- Reduction 2;
- Reduction 5;
- Reduction 4.

As a final note, we point out that one of the most effective reductions used in the classical Steiner problems, revisited next, cannot be applied in the present work.

Consider the terminal node i such that $deg(i) \geq 2$ and for which j is the closest node and k is the second-closest node (in terms of delay). If

$$d_{ij} + \min\{p_{jz} | z \in T \wedge z \neq i\} \leq d_{ik}$$

then the edge $\{i, j\}$ will belong to the optimal Steiner tree and the graph G can be contracted along the edge $\{i, j\}$ [28].

It is possible to construct a counter-example to show this reduction is not valid in our Steiner problem with delay constraints.

VI. COMPUTATIONAL RESULTS

In this section, we present computational results comparing the optimal solutions obtained by the exact method via the ILP model and the heuristic solutions. Since we calculate the optimal solutions, we can assess the performance of our heuristic for the set of networks considered.

As this problem was inspired by network design problems, we started by testing our heuristic on some of the SNDlib instances [29] representing core telecommunication network topologies. For the computational results, we considered two telecommunication bi-connected core networks: germany50 from SNDlib with 50 nodes and 88 links (the largest we tested of the SNDlib instances), and coronet available at http://www.monarchna.com/topology.html with 75 nodes and 99 links (representing a USA core network). For both networks the cost was given by the delays, that were calculated as the distances over the Earth's surface from the GPS coordinates of the nodes. We considered two sets of terminal nodes (controller sets) for each network: for germany50, we considered instance ger1 with $T = \{22, 24, 38, 40\}$ and instance ger2 with $T = \{4, 7, 24, 38\}$; for coronet, we considered instance cor1 with $T = \{10, 20, 26, 33, 37, 38, 45, 56\}$ and instance cor2 with $T = \{16, 20, 26, 38, 56, 65, 66, 72\}$. These instances were chosen from our previous work [13], where the controller placements are for a robust SDN network and for which the addressed Steiner tree problem is relevant. Moreover, we also considered the testset B from the SteinLib library [30] with topologies ranging from 50 nodes and 63 links to 100 nodes and 200 links, since they were generated to represent more difficult Steiner tree problems and are roughly the same size as the previously mentioned networks. Details on these networks (including the set of terminal nodes and the cost of the links) can be found in the SteinLib site (http://steinlib.zib.de/showset.php?B).

TABLE 1. Reductions for testset B from SteinLib.

			$\mathcal{D}$									
		Original	D_g	$0.95D_g$	$0.90D_g$	$0.85D_g$	$0.80D_g$	$0.75D_g$	$0.70D_g$	$0.65D_g$	$0.60D_g$	$0.55D_g$
b01	$\|N\|$	50	18	18	18	18	infeasible					
	$\|E\|$	63	26	26	25	25						
	$\|T\|$	9	8	8	8	8						
	No. L.		4	4	5	5						
b02	$\|N\|$	50	22	22	19	19	17	infeasible				
	$\|E\|$	63	31	31	25	24	20					
	$\|T\|$	13	12	12	12	12	12					
	No. L.		5	5	5	6	7					
b03	$\|N\|$	50	29	29	infeasible							
	$\|E\|$	63	39	39								
	$\|T\|$	25	21	21								
	No. L.		8	8								
b04	$\|N\|$	50	34	34	31	31	26	infeasible				
	$\|E\|$	100	70	70	62	60	49					
	$\|T\|$	9	9	9	9	9	9					
	No. L.		1	1	2	3	3					
b05	$\|N\|$	50	43	43	43	43	43	43	40	36	32	
	$\|E\|$	100	89	89	89	89	89	89	80	65	54	
	$\|T\|$	13	13	13	13	13	13	13	13	13	13	
	No. L.		2	2	2	2	2	2	2	2	3	infeasible
b06	$\|N\|$	50	46	42	40	infeasible						
	$\|E\|$	100	82	69	62							
	$\|T\|$	25	25	24	23							
	No. L.		5	9	10							
b07	$\|N\|$	75	25	21	9	9	infeasible					
	$\|E\|$	94	33	26	9	9						
	$\|T\|$	13	11	10	6	6						
	No. L.		6	7	4	4						
b08	$\|N\|$	75	33	33	33	33	33	31	infeasible			
	$\|E\|$	94	47	47	47	47	47	44				
	$\|T\|$	19	19	19	19	19	19	18				
	No. L.		5	5	5	5	5	5				
b09	$\|N\|$	75	43	43	43	infeasible						
	$\|E\|$	94	60	60	60							
	$\|T\|$	38	33	33	33							
	No. L.		9	9	9							
b10	$\|N\|$	75	51	47	46	42	infeasible					
	$\|E\|$	150	103	94	90	78						
	$\|T\|$	13	13	13	13	13						
	No. L.		3	4	4	4						
b11	$\|N\|$	75	50	infeasible								
	$\|E\|$	150	95									
	$\|T\|$	19	19									
	No. L.		3									
b12	$\|N\|$	75	66	infeasible								
	$\|E\|$	150	124									
	$\|T\|$	38	38									
	No. L.		5									
b13	$\|N\|$	100	38	36	32	infeasible						
	$\|E\|$	125	54	50	42							
	$\|T\|$	17	16	16	16							
	No. L.		7	8	10							
b14	$\|N\|$	100	51	48	infeasible							
	$\|E\|$	125	75	70								
	$\|T\|$	25	24	22								
	No. L.		9	10								
b15	$\|N\|$	100	62	62	61	infeasible						
	$\|E\|$	125	87	86	84							
	$\|T\|$	50	46	46	45							
	No. L.		11	12	13							
b16	$\|N\|$	100	74	infeasible								
	$\|E\|$	200	158									
	$\|T\|$	17	17									
	No. L.		1									
b17	$\|N\|$	100	58	infeasible								
	$\|E\|$	200	103									
	$\|T\|$	25	25									
	No. L.		6									
b18	$\|N\|$	100	83	infeasible								
	$\|E\|$	200	167									
	$\|T\|$	50	49									
	No. L.		8									

We considered $\mathcal{D}$ to be given as a percentage of the network diameter D_g [12], [13], and it is the maximum delay to be

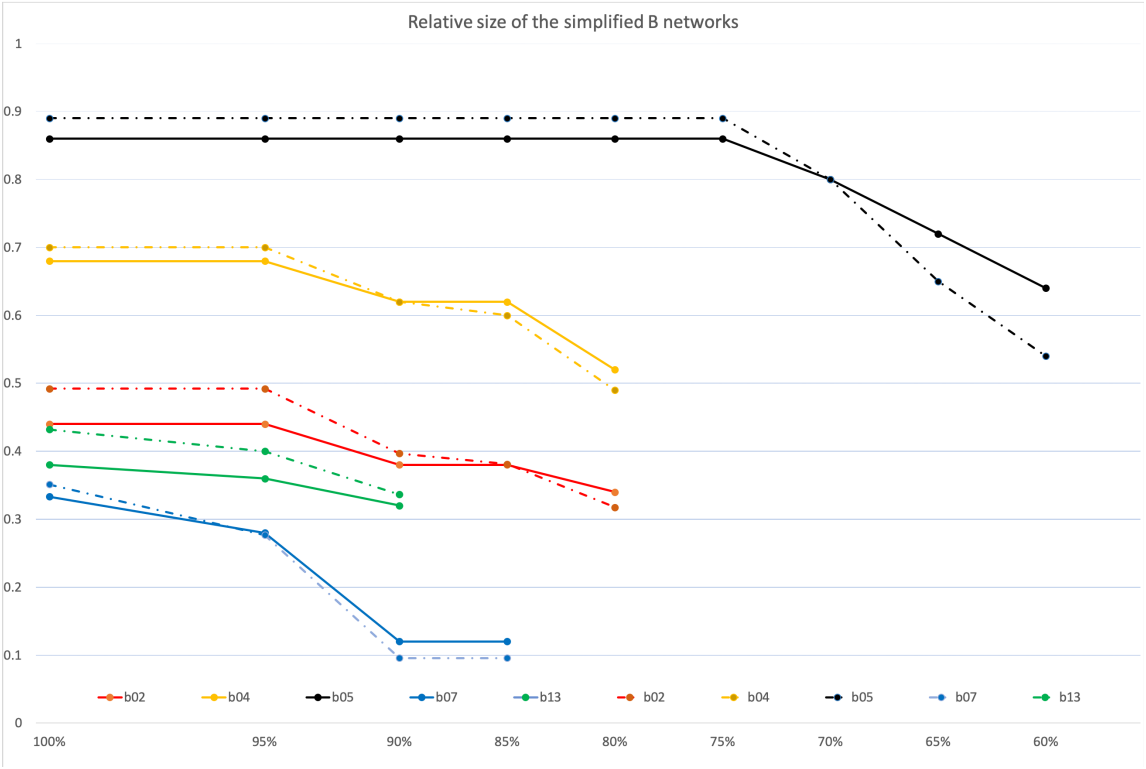


FIGURE 1. Relative number of nodes (solid lines) and links (dashed lines) of the simplified B networks versus $\mathcal{D}$ given as a percentage of D_g.

guaranteed between any two terminal nodes of the Steiner tree. For all networks, we started with $\mathcal{D} = D_g$, i.e., $\mathcal{D}$ is initially the network diameter. The value was then decreased by a factor of 0.05 until the problem became infeasible.

The ILP model and heuristic (Algorithm 1) were implemented in C/C++. For solving the ILP, the Callable libraries of CPLEX 12.6 were additionally used. All instances were run on a Dell Precision 7500, Intel(R) Xeon(R) CPU X5660 with 6 cores and 48 GB of RAM, running at 2.80 GHz.

Since the heuristic is stochastic in nature, it was run 10 times for each instance and we used $\theta = 2$ in all runs.

Reductions were applied for each instance as a pre-processing step. Then, the ILP model and heuristic are solved in the reduced graph, with the links that must be on the Steiner tree added to it. In the ILP model, these links are fixed by setting the corresponding y_{ij} variables to 1. We have imposed a time limit for the ILP model of 6 hours (21600 seconds), since it is computationally challenging for many instances of the testset B from SteinLib.

In Table 1, we show the impact of the reductions in each instance of testset B. For each instance, the number of nodes $|N|$, the number of links $|E|$ and the number of terminal nodes $|T|$ are shown for the original graph b01 to b18 and for each of the reduced graphs obtained for each instance. For the reduced graphs, the number of mandatory links (No. L.) needed to be added to the final Steiner tree is also shown.

While some reductions are more pronounced as in b13, others are less pronounced as in b05, since they strongly depend on the graph's topology and the set of terminal nodes. However, it is clear that the reductions in most graphs are significant. An advantage of using reductions, is that when the reduced graph becomes disconnected, then we know the instance is infeasible for that value of $\mathcal{D}$ and, consequently for lower values of $\mathcal{D}$.

Fig. 1 illustrates the impact of the reductions for some of the instances: b02, b04, b05, b07 and b13. We can see the relative number of nodes (shown as solid lines) and links (shown as dashed lines) obtained by applying the reductions for the different values of $\mathcal{D}$, in each instance.

Note that as the value of $\mathcal{D}$ decreases (x-axis), there is a tendency for the relative number of nodes and links to decrease too, as more reductions can be applied. Note that the lack of lines beyond some value $\mathcal{D}$ shows that the graph has become disconnected.

The results obtained with the ILP and heuristic are shown in Tables 2-4 for the testset B of SteinLib, for the two instances of germany50, and for the two instances of coronet (respectively). In all the tables, the first column identifies the instance and the second column shows the maximum delay $\mathcal{D}$ considered (as a percentage of the graph diameter D_g). The following three columns refer to the results obtained with the ILP model, while the final five columns refer to those obtained with the heuristic. For the ILP, the columns show the

TABLE 2. ILP and heuristic results for the testset B from SteinLib.

		ILP			Heuristic				
					best solution				
	$\mathcal{D}$	cost	diam	time (s)	cost	diam	avg cost	avg diam	avg time (s)
b01	D_g	82	41	0.64	82	41	82	41	0.02
	$0.95D_g$	82	41	0.33	82	41	82	41	0.02
	$0.90D_g$	82	41	0.27	82	41	82	41	0.02
	$0.85D_g$	82	41	0.28	82	41	82	41	0.02
b02	D_g	83	49	2.74	83	49	83	49	0.07
	$0.95D_g$	84	44	2.33	84	41	84	42.2	0.06
	$0.90D_g$	84	41	0.94	84	41	84	43.1	0.05
	$0.85D_g$	84	41	0.83	84	41	84	41	0.05
	$0.80D_g$	infeas		0.23	no sol				0.03
b03	D_g	138	48	15.91	138	48	138	48	0.34
	$0.95D_g$	138	48	12.40	138	48	138	48	0.30
b04	D_g	59	32	60.27	59	32	59.2	31.8	0.06
	$0.95D_g$	59	32	35.77	59	32	59.1	31.9	0.06
	$0.90D_g$	70	29	3235.86	73	29	73[a]	29[a]	0.04
	$0.85D_g$	no sol		TL	no sol				0.04
	$0.80D_g$	no sol		TL	no sol				0.03
b05	D_g	61	29	4826.96	61	28	61	29.5	0.21
	$0.95D_g$	61	28	1686.37	61	28	61.3	27.9	0.21
	$0.90D_g$	61	28	1879.12	61	28	61.4	28.2	0.21
	$0.85D_g$	61	28	1966.24	61	28	61.1	28.8	0.21
	$0.80D_g$	61	28	2812.43	61	28	61.3	28.2	0.21
	$0.75D_g$	61	29	2958.42	61	28	61.2	28.2	0.20
	$0.70D_g$	61	28	1848.67	61	28	61.5	27.7	0.17
	$0.65D_g$	63	26	2778.05	63	26	64.4	26	0.11
	$0.60D_g$	67	24	1223.71	67	24	67.8[b]	23.6[b]	0.07
b06	D_g	no sol		TL	no sol				0.17
	$0.95D_g$	no sol		TL	no sol				0.13
	$0.90D_g$	no sol		TL	no sol				0.10
b07	D_g	112	51	0.94	112	51	112	51	0.06
	$0.95D_g$	112	51	0.43	112	51	112	51	0.04
	$0.90D_g$	113	45	0.02	113	45	113	45	0.01
	$0.85D_g$	113	45	0.03	113	45	113	45	0.01
b08	D_g	104	58	78.59	104	58	104.1	57.6	0.04
	$0.95D_g$	104	58	66.68	104	58	104	58	0.35
	$0.90D_g$	105	54	151.94	105	54	105.3	53.3	0.30
	$0.85D_g$	107	52	243.12	107	52	107.1	52	0.28
	$0.80D_g$	108	47	237.93	108	47	108.2	46.4	0.22
	$0.75D_g$	109	44	27.98	109	44	109.4	44.1	0.15
b09	D_g	221	58	5715.89	221	58	221.1	58	1.81
	$0.95D_g$	no sol		TL	no sol				1.08
	$0.90D_g$	infeas		1461.68	no sol				0.96
b10	D_g	96	34	3323.47	97	33	98.5[c]	32.25[c]	0.16
	$0.95D_g$	97	33	6882.50	97	33	98.5[c]	33[c]	0.13
	$0.90D_g$	no sol		TL	no sol				0.11
	$0.85D_g$	no sol		TL	no sol				0.08
b11	D_g	no sol		TL	no sol				0.02
b12	D_g	no sol		TL	no sol				0.63
b13	D_g	174	59	186.87	174	59	176.2	59.1	0.16
	$0.95D_g$	176	56	7.19	176	56	182[d]	56[d]	0.11
	$0.90D_g$	infeas		42.45	no sol				0.08
b14	D_g	no sol		TL	no sol				0.28
	$0.95D_g$	no sol		TL	no sol				0.18
b15	D_g	no sol		TL	325	67	333.2	67.5	2.30
	$0.95D_g$	no sol		TL	337	64	343.4[b]	64[b]	1.60
	$0.90D_g$	no sol		TL	no sol				1.11
b16	D_g	no sol		TL	132	47	132.8	47	0.42
b17	D_g	no sol		TL	no sol				0.43
b18	D_g	no sol		TL	no sol				1.08

[a] 8 runs with no solution [b] 5 runs with no solution [c] 2 runs with no solution [d] 4 runs with no solution

cost of the Steiner tree, the diameter of the obtained minimum cost tree (which is given as the largest delay between any two terminal nodes in the Steiner tree), and the runtime in seconds.

For the heuristic, 10 runs were executed. The results displayed in the tables show the information regarding the best solution obtained (the cost of the Steiner tree and the diameter of the obtained minimum cost tree), and the average results

TABLE 3. ILP and heuristic results for germany50.

		ILP			Heuristic				
					best solution				
	$\mathcal{D}$	cost	diam	time (s)	cost	diam	avg cost	avg diam	avg time (s)
ger1	D_g	882	598	1.35	882	598	885.3	657.2	0.01
	$0.95D_g$	882	598	1.04	882	598	882.4	648.4	0.01
	$0.90D_g$	882	598	0.92	882	598	885	619.4	0.01
	$0.85D_g$	882	598	0.44	882	598	885.9	661	0.01
	$0.80D_g$	882	598	0.86	882	598	885.3	657.2	0.01
	$0.75D_g$	882	598	0.39	882	598	890.4	586.6	0.01
	$0.70D_g$	882	598	0.17	882	598	889[a]	588.5[a]	0.01
	$0.65D_g$	882	598	0.11	882	598	889	588.5	0.00
	$0.60D_g$	965	555	0.08	965	555	965[b]	555[b]	0.00
ger2	D_g	935	609	0.90	935	609	940.4	615.4	0.01
	$0.95D_g$	935	609	0.89	935	609	951.2	628.2	0.01
	$0.90D_g$	935	609	0.67	935	609	951.2	628.2	0.01
	$0.85D_g$	935	609	0.69	935	609	947.1	613.6	0.01
	$0.80D_g$	935	609	0.99	935	609	974.1	645.6	0.01
	$0.75D_g$	935	609	0.17	935	609	945.8	621.8	0.01
	$0.70D_g$	935	609	0.07	935	609	948.4	605.4	0.01
	$0.65D_g$	1002	591	0.03	1002	591	1002[b]	591[b]	0.00

[a] 2 runs with no solution [b] 1 run with no solution

TABLE 4. ILP and heuristic results for Coronet.

		ILP			Heuristic				
					best solution				
	$\mathcal{D}$	cost	diam	time (s)	cost	diam	avg cost	avg diam	avg time (s)
cor1	D_g	5916	3376	3.10	5916	3376	5916	3376	0.03
	$0.95D_g$	5916	3376	3.49	5916	3376	5916	3376	0.03
	$0.90D_g$	5916	3376	1.79	5916	3376	5916	3376	0.02
	$0.85D_g$	5916	3376	1.38	5916	3376	5916	3376	0.02
	$0.80D_g$	5916	3376	1.05	5916	3376	5916	3376	0.02
	$0.75D_g$	5916	3376	0.65	5916	3376	5916	3376	0.01
	$0.70D_g$	5916	3376	0.42	5916	3376	5916	3376	0.01
	$0.65D_g$	5916	3376	0.25	5916	3376	5916	3376	0.01
	$0.60D_g$	5916	3376	0.05	5916	3376	5916	3376	0.01
	$0.55D_g$	5916*	3376*		5916*	3376*			
cor2	D_g	4937	3376	1.20	4937	3376	4937	3376	0.03
	$0.95D_g$	4937	3376	1.19	4937	3376	4937	3376	0.03
	$0.90D_g$	4937	3376	0.69	4937	3376	4937	3376	0.02
	$0.85D_g$	4937	3376	0.72	4937	3376	4937	3376	0.02
	$0.80D_g$	4937	3376	0.75	4937	3376	4937	3376	0.02
	$0.75D_g$	4937	3376	0.27	4937	3376	4937	3376	0.02
	$0.70D_g$	4937	3376	0.17	4937	3376	4937	3376	0.02
	$0.65D_g$	4937	3376	0.14	4937	3376	4937	3376	0.01
	$0.60D_g$	4937	3376	0.14	4937	3376	4937	3376	0.01
	$0.55D_g$	4937*	3376*		4937*	3376*			

$*$ solution obtained directly from the reduced graph

(cost, diameter and runtime in seconds) for the 10 runs. Note that in some situations (identified in the tables) there were no solutions for some of the 10 runs.

Concerning Table 2, some of the instances of testset B are very challenging for both the ILP, not finding any solution after the time limit of 6 hours (denoted by TL in the table), and the heuristic (that could not find any solution either). The runtimes for the heuristic are very small (almost always less than a second on average), while the runtimes for the ILP model can become very high.

In many cases, the solutions found by the heuristic in the 10 runs for each instance are such that the average value of their cost is the same as the minimum value of the cost, which means that the heuristic was able to find minimum cost solutions in all those 10 runs. This may be observable in the heuristic solutions for instances b01, b02, b03, b07 (for all values of $\mathcal{D}$). This is also observable for b05 with $\mathcal{D} = D_g$ and b08 with $\mathcal{D} = 0.95D_g$. In all cases, the optimal solution is found at least once (shown in best solution), except for instances b04 with $\mathcal{D} = 0.9\,D_g$ and b10 with $\mathcal{D} = D_g$. For these instances, increasing the value of θ in the heuristic may help find better solutions, since the search space becomes wider. This is also true for those instances where at least one of the 10 runs did not retrieve a solution, such as b04 with $\mathcal{D} = 0.90D_g$, b05 with $\mathcal{D} = 0.60D_g$, b10 with $\mathcal{D} = D_g$ and $0.95D_g$, b13 and b15 with $\mathcal{D} = 0.95D_g$.

The average cost of the obtained solutions is in general only slightly higher than the minimum value, which means that even when the heuristic is not able to find the minimum cost solution in a specific run, the obtained solution has a cost

close to the minimum (0.9% higher on average) and it may be considered sub-optimal.

Also note that the heuristic sometimes finds a minimum cost tree with a smaller diameter than the ILP, as for b02 with $\mathcal{D} = 0.95D_g$ and b05 with $\mathcal{D} = D_g$. This reinforces the idea that the heuristic tries to find the minimum cost solution satisfying a delay constraint, but it also tends to keep the solutions with the lowest possible delay. In the ILP, the cost is minimized as long as the pairwise delay constraint is satisfied (no emphasis is given to the diameter). However, it is possible to obtain the minimum cost Steiner tree (while satisfying the maximum delay constraint) with minimum diameter, by solving a second ILP model which is similar to the previous one but now minimizing the diameter with an additional constraint to guarantee the known minimum cost.

Finally, note that the heuristic was able to find solutions for b15 with $\mathcal{D} = D_g$ and $\mathcal{D} = 0.9\ D_g$, and for b16 with $\mathcal{D} = D_g$ (in very small runtimes), whereas the ILP model was not (in the given time limit). For some instances, such as b02 with $\mathcal{D} = 0.8\ D_g$, b09 and b13 with $\mathcal{D} = 0.9\ D_g$, the ILP model was able to determine that the problem was infeasible, but the heuristic was not.

Concerning Table 3, the germany50 instances were efficiently solved with the ILP model as shown by the runtimes. This is due to the fact that germany50 represents a core telecommunication network, and so it is bi-connected and has an average node degree of 3.52. Moreover, the number of terminal nodes is very small for both network cases, $|T| = 4$, whereas the number of terminal nodes in testset B ranges from 9 to 50.

The table shows that our proposed heuristic is also able to find the optimal solutions for all the instances of germany50 (with the same tree diameter) at least once, and with even smaller runtimes in general. For the germany50 instances, only for ger1 with $\mathcal{D} = 0.60\ D_g$ and for ger2 with $\mathcal{D} = 0.65\ D_g$ were all the solutions (for the different successful runs) minimum cost solutions. In the remaining cases, the average cost of the obtained solutions is on average 1.1% higher than the minimum cost (a higher relative variation than the one for the results for the testset B networks).

Concerning Table 4, the coronet instances were also efficiently solved with the ILP model as shown by the runtimes (under 4 seconds). Coronet also represents a core telecommunication network, being bi-connected and has an average node degree of 2.64. The number of terminal nodes considered is $|T| = 8$.

Interestingly, for both instances cor1 and cor2 with $\mathcal{D} = 0.55D_g$, the optimal solution was obtained through the graph reductions alone. For all other instances, all the 10 runs retrieved the optimal solution. Interestingly enough, for all instances of cor1, the optimal Steiner tree is the same (for all values of $\mathcal{D}$). This is also true for all instances of cor2. While the minimum cost is smaller for cor2 than for cor1,

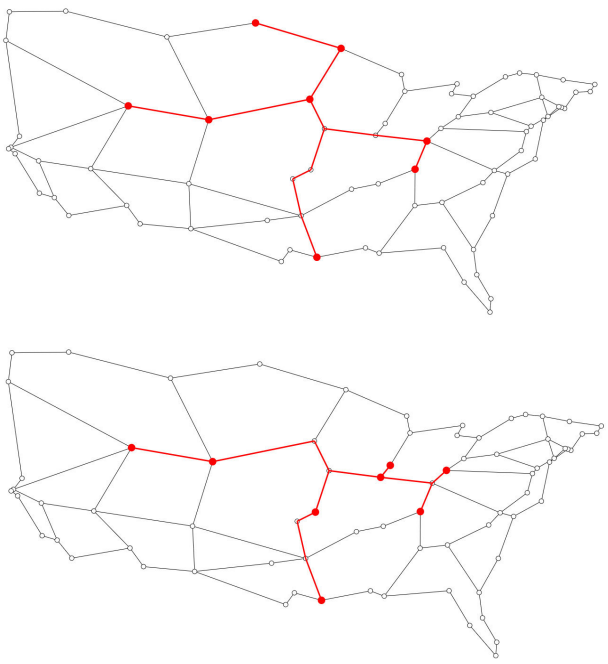


FIGURE 2. Optimal Steiner tree for cor1 (top) and for cor2 (bottom) respectively. The Steiner tree is shown in red and the terminal nodes are shown as red dots.

the diameter of the trees is the same. These results suggest that the variety of Steiner trees for each set of terminal nodes may be very small. The Steiner trees for cor1 and cor2 are shown in Fig. 2. The diameter of the trees is the same, since the diameter is determined by the subtree from the furthest left terminal node down to the bottom terminal node (in both cases). We can also see that the cost of the first tree is greater, because of the two long edges connecting the top terminal nodes.

Finally, to illustrate the effectiveness of the reductions, we also show the relative runtimes for the heuristic, of the networks of testset B, in Fig. 3. These relative times were calculated as the ratio of the runtime of the heuristic for the reduced networks to the runtime of the heuristic for the original networks. Each group of colored bars refers to a network (b01 to b18). Each colored bar refers to an instance of that network, corresponding to a different $\mathcal{D}$ as a percentage (in the legend) of D_g. The relative runtimes for the ILP are not shown, since there are many instances ending due to the imposed time limit.

For all instances, the best solutions obtained were the same for the original and reduced graphs (among 10 runs), except for b15 with $\mathcal{D} = D_g$ and $0.95D_g$, where the reduced graph allowed to obtain better solutions.

Note that for all the instances of b01, b02, b03, b07, b09 and b13 the time diminishes to less than 50% in the reduced graphs. For all the other instances, the time decreases

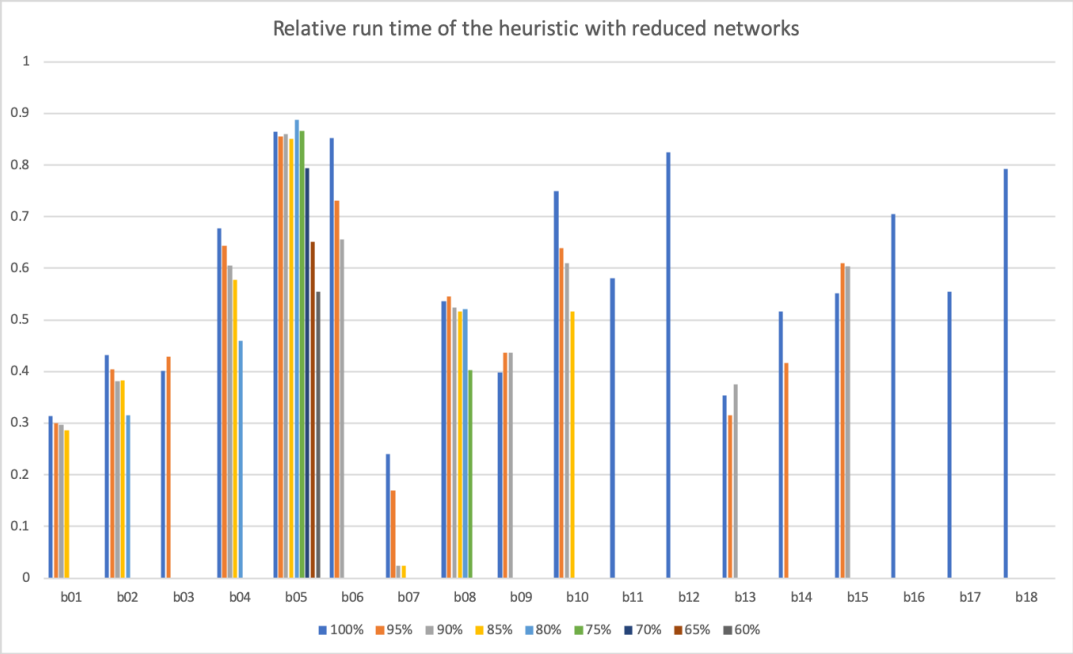


FIGURE 3. Relative time for the heuristic with the reduced networks, for all instances of testset B.

to less than 80% of the original runtime, except for b05 with $\mathcal{D}$ above $0.70D_g$, b06 with $\mathcal{D} = D_g$ and b12.

VII. CONCLUSION

In this paper, we addressed the problem of finding the minimum cost Steiner tree with delay constraints. We proposed an exact method via an ILP model for the minimum cost Steiner tree while guaranteeing that the delay between any two terminal nodes is at most a maximum value $\mathcal{D}$.

We also proposed two novel and two modified reductions for the particular STP addressed here. The set of reductions used allowed for significant graph reductions in most instances we tested. For the coronet instances with $\mathcal{D} = 0.55D_g$, we observed that the graph reductions were so significant, that they alone were able to find the optimal solution.

The ILP model efficiently solved all instances for germany50 and coronet (graphs representing core communication networks), but struggled in many instances of the testset B (even for the corresponding reduced graphs). Therefore, we also proposed a simple and efficient heuristic which always found the optimal solutions, except in two cases: b04 with $\mathcal{D} = 0.90\ D_g$; and b10 with $\mathcal{D} = D_g$. Therefore, this ensures the quality of our heuristic. We observed that the heuristic is very fast (all runtimes under 3 seconds).

Finally, we observed that the heuristic also found solutions for some of the instances that the ILP model did not. These observations demonstrate the suitability of the heuristic for larger and more complicated instances.

As a final note, we point out that increasing θ can help find better solutions with increased runtimes, for more difficult cases.

REFERENCES

[1] M. R. Garey, R. L. Graham, and D. S. Johnson, "Some NP-complete geometric problems," in *Proc. 8th Annu. ACM Symp. Theory Comput.*, 1976, pp. 10–22.

[2] I. Ljubić, "Solving Steiner trees: Recent advances, challenges, and perspectives," *Networks*, vol. 77, no. 2, pp. 177–204, Mar. 2021.

[3] D. Du and X. Hu, *Steiner Tree Problems in Computer Communication Networks*. Singapore: World Scientific, 2008.

[4] K.-H. Vik, P. Halvorsen, and C. Griwodz, "Evaluating Steiner-tree heuristics and diameter variations for application layer multicast," *Comput. Netw.*, vol. 52, no. 15, pp. 2872–2893, Oct. 2008.

[5] S. Voß, "The Steiner tree problem with hop constraints," *Ann. Oper. Res.*, vol. 86, pp. 321–345, Jan. 1999.

[6] A. M. Costa, J.-F. Cordeau, and G. Laporte, "Models and branch-and-cut algorithms for the steiner tree problem with revenues, budget and hop constraints," *Networks*, vol. 53, no. 2, pp. 141–159, Mar. 2009.

[7] M. Santos, L. M. A. Drummond, and E. Uchoa, "A distributed dual ascent algorithm for the hop-constrained steiner tree problem," *Oper. Res. Lett.*, vol. 38, no. 1, pp. 57–62, Jan. 2010.

[8] H. Matsuura, "Energy-saving routing algorithm using Steiner tree," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Ghent, Belgium, May 2013, pp. 378–386.

[9] F. Grandoni, "On min-power Steiner tree," in *Algorithms* (Lecture Notes in Computer Science), vol. 7501, L. Epstein and P. Ferragina, Eds. Berlin, Germany: Springer, Sep. 2012, pp. 527–538.

[10] T. Das and M. Gurusamy, "Controller placement for resilient network state synchronization in multi-controller SDN," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1299–1303, Jun. 2020.

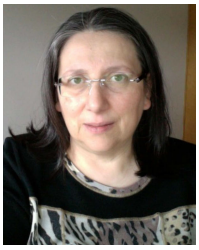
[11] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *Proc. IEEE Conf. Comput. Commun.*, Hong Kong, Apr. 2015, pp. 181–189.

[12] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient SDN architecture," in *Proc. 12th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, Paris, France, Mar. 2016, pp. 145–151.

[13] D. Santos, A. de Sousa, and C. Mas-Machuca, "Combined control and data plane robustness of SDN networks against malicious node attacks," in *Proc. CNSM*, Rome, Italy, Nov. 2018, pp. 54–62.

[14] V. Leggieri, M. Haouari, and C. Triki, "The Steiner tree problem with delays: A compact formulation and reduction procedures," *Discrete Appl. Math.*, vol. 164, pp. 178–190, Feb. 2014.

[15] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communication," *Netw., IEEE/ACM Trans. on*, vol. 1, no. 3, pp. 286–292, Jun. 1993.
[16] R. Sriram, G. Manimaran, and C. S. R. Murthy, "Algorithms for delay-constrained low-cost multicast tree construction," *Comput. Commun.*, vol. 21, no. 18, pp. 1693–1706, Dec. 1998.
[17] M. Parsa, Q. Zhu, and J. J. Garcia-Luna-Aceves, "An iterative algorithm for delay-constrained minimum-cost multicasting," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 461–474, Aug. 1998.
[18] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.
[19] R. Qu, Y. Xu, J. P. Castro, and D. Landa-Silva, "Particle swarm optimization for the steiner tree in graph and delay-constrained multicast routing problems," *J. Heuristics*, vol. 19, no. 2, pp. 317–342, Apr. 2013.
[20] H. A. Harutyunyan and M. Terzian, "Directional core selection approach and dynamic tree reorganization for delay and delay variation multicast routing," *Int. J. Commun. Syst.*, vol. 31, no. 4, p. e3489, Mar. 2018.
[21] S. Ayoubi, C. Assi, K. Shaban, and L. Narayanan, "MINTED: *Multicast virtual network* embedding in cloud data centers with delay constraints," *IEEE Trans. Commun.*, vol. 63, no. 4, pp. 1291–1305, Apr. 2015.
[22] S. P. Sahoo and M. R. Kabat, "Multi-constrained multicast routing improved by hybrid bacteria foraging/particle swarm optimization," *Comput. Sci.*, vol. 20, no. 2, pp. 245–269, 2019.
[23] S. Aggarwal, M. Limaye, A. Netravali, and K. Sabnani, "Constrained diameter steiner trees for multicast conferences in overlay networks," in *Proc. 1st Int. Conf. Qual. Service Heterogeneous Wired/Wireless Netw.*, Dallas, TX, USA, 2004, pp. 262–271.
[24] W. Ding, G. Lin, and G. Xue, "Diameter-constrained Steiner tree," in *Proc. 4th Int. Conf. Combinat. Optim. Appl.* (Lecture Notes in Computer Science), vol. 6509, W. Wu and O. Daescu, Eds. Berlin, Germany: Springer, Dec. 2010, pp. 243–253.
[25] W. Ding and G. Xue, "Minimum diameter cost-constrained Steiner trees," *J. Combinat. Optim.*, vol. 27, no. 1, pp. 32–48, Jan. 2014.
[26] *IBM ILOG CPLEX Optimization Studio V12.6*. IBM, Armonk, NY, USA, 2013.
[27] P. Winter and J. MacGregor Smith, "Path-distance heuristics for the Steiner problem in undirected networks," *Algorithmica*, vol. 7, nos. 1–6, pp. 309–327, Jun. 1992.
[28] P. Winter, "Steiner problem in networks: A survey," *Network*, vol. 17, no. 2, pp. 129–167, Jan. 1987.
[29] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0-survivable network design library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010. [Online]. Available: http://sndlib.zib.de
[30] T. Koch, A. Martin, and S. Voß, "SteinLib: An updated library on Steiner tree problems in graphs," in *Steiner Trees in Industry* (Combinatorial Optimization), vol. 11, X. Z. Cheng and D.-Z. Du, Eds. Boston, MA, USA: Springer, 2001, pp. 285–325.



LÚCIA MARTINS received the Ph.D. degree in electrical engineering (telecommunications and electronics) from the University of Coimbra, in 2004. She worked as a Development Engineer for several years at a Portugal telecom public operator. She is an Assistant Professor with the Department of Electrical and Computer Engineering, Faculty of Sciences and Technology, University of Coimbra. Her research interest includes multiple criteria network design and optimization.

DORABELLA SANTOS received the M.Sc. degree in mathematics and the Ph.D. degree in electrotechnical engineering from the University of Aveiro, Portugal, in 2003 and 2007, respectively. She has been a Researcher at the Institute for Systems Engineering and Computers at Coimbra (INESC Coimbra), since December 2018. She has coauthored several scientific publications in international journals, conference proceedings, and book chapters. Her research interests include mathematical optimization problems for software-defined networking, communication network design problems, and network routing problems. She also served as a TPC Member for some international conferences.

TERESA GOMES (Member, IEEE) received the Ph.D. degree in electrical engineering (telecommunications and electronics) from the University of Coimbra, in 1998. From April 2013 to July 2013, she was a Visiting Researcher at the School of Information Sciences, University of Pittsburgh, USA. She is an Assistant Professor with the University of Coimbra. She is also a Researcher at the Institute for Systems Engineering and Computers at Coimbra (INESC Coimbra). She is the author/coauthor of over 90 technical publications in international journals, conference proceedings, and book chapters, including one European patent. She has been responsible for several national projects, mainly with industry. Her research interests include routing, protection and reliability analysis models, and algorithms for communication networks. She has also served as a TPC Member of numerous international conferences and was the Co-Chair of DRCN 2019. She is an Associate Editor of *Journal of Network and Systems Management* (Springer) and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT.

RITA GIRÃO-SILVA received the Ph.D. Diploma degree in electrical engineering (telecommunications and electronics) from the University of Coimbra, in 2009. She is an Assistant Professor with the Department of Electrical and Computer Engineering, University of Coimbra, and a Researcher at the Institute for Systems Engineering and Computers at Coimbra (INESC Coimbra). She has participated in different research and development projects, including projects of cooperation between the university and industry. Her research interests include routing models, network reliability, and network flow models for routing in telecommunications networks, as well as applications of operational research techniques to problems in telecommunication networks.