Received: 4 December 2018 | Revised: 14 August 2019 | Accepted: 23 August 2019 | Published on: 8 October 2019

DOI: 10.1002/net.21913

RESEARCH ARTICLE

WILEY

A linear programming based approach to the Steiner tree problem with a fixed number of terminals

Matias Siebert | **Shabbir Ahmed** | **George Nemhauser**

H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia

Correspondence
Matias Siebert, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332.
Email: msiebert6@gatech.edu

Funding information
This research was supported by the CONICYT, 72160393. Office on Naval Research, N00014-15-1-2078, N00014-18-1-2075.

Abstract

We present a set of integer programs (IPs) for the Steiner tree problem with the property that the best solution obtained by solving all IPs provides an optimal Steiner tree. Each IP is polynomial in the size of the underlying graph and our main result is that the linear programming (LP) relaxation of each IP is integral so that it can be solved as a linear program. However, the number of IPs grows exponentially with the number of terminals in the Steiner tree. As a consequence, we are able to solve the Steiner tree problem by solving a polynomial number of LPs, when the number of terminals is fixed.

KEYWORDS

convex hull, independent subproblems, integer programming, laminar family, linear programming, Steiner tree

1 | INTRODUCTION

Given an undirected graph $G = (V, E)$, where V is the set of nodes, E is the set of edges, $c_e \geq 0$ is the cost of each edge $e \in E$, and $R \subseteq V$ is a set of *terminal nodes*, a Steiner tree is a subgraph of G, which is a tree such that all terminal nodes are connected. The Steiner tree problem is to find such a tree that minimizes the sum of the edge costs over all edges in the tree. The tree may contain nodes in $V \backslash R$, which are called *Steiner nodes*. Note that we may assume without loss of generality that $c > 0$, since we can contract all the edges with cost 0. The Steiner tree problem is NP-Hard [21], in fact, it is even NP-Hard to find approximate solutions whose cost is within a factor $\frac{96}{95}$ of the cost of an optimal solution [3, 8].

An important line of research has been to study integer linear programming (LP) formulations for this problem. In Goemans and Myung [18], the authors provide a catalog of Steiner tree formulations, and show the equivalence of some of these formulations. In Goemans [17], the author studies the vertex-weighted version of the undirected Steiner tree problem, and presents a complete description of the polytope when the graph is *series-parallel*. By projecting this formulation, some facet-defining inequalities for the Steiner tree polytope are obtained. In Polzin and Daneshmand [26], the authors compare different formulations of the Steiner tree problem in terms of the strength of their linear relaxations (LP), and propose a new polynomial size formulation with a stronger LP relaxation than the ones analyzed in their paper.

In every Steiner tree, we can pick an arbitrary node $r \in R$ and find a direction of the edges, such that we can construct a unique arborescence rooted at r whose leaf nodes correspond to nodes in $R \backslash \{r\}$. Note that an r-arborescence is a directed tree where every node in the arborescence is reachable from the root node r. Consequently, the Steiner tree problem can be modeled as an r-arborescence whose leaves are nodes in $R \backslash \{r\}$ and where all terminal nodes are reached from r. A well-studied formulation that uses this fact is the *Bidirected Cut* formulation [13, 35]. This is a compact and simple formulation, whose integrality gap is known to be at most 2, but is believed to be close to one. The best known lower bound on the integrality gap for this formulation is $\frac{36}{31} \approx 1.16$ [6]. An upper bound on the integrality gap for this formulation has been found for special cases. For *quasi-bipartite* graphs, the integrality gap is upper bounded by $\frac{3}{2}$ [30], and for *claw-free* graphs it is upper bounded by ln(4) [14].

Another well-studied formulation is the so called *Hypergraphic* formulation [24, 29, 33]. This formulation has one variable for each *component* of the graph, where a *component* is a tree whose leaves correspond to terminal nodes. The number

of variables and constraints for this formulation are proportional to the number of *components*, which is exponential in the number of terminal nodes. On the positive side, we have that the *Hypergraphic* formulation is stronger than the *Bidirected Cut* formulation [29]. The integrality gap of the *Hypergraphic* formulation is lower bounded by $\frac{8}{7} \approx 1.14$ [24] and upper bounded by ln(4) [19]. Moreover, in Feldmann et al. [14] the authors show that in *claw-free* graphs both formulations are equivalent. On the down side, solving the *Hypergraphic* formulation is strongly NP-Hard [19]. To address this problem, researchers have proposed to solve it only considering components with at most k leaves, where k is a fixed value. This considerably reduces the number of constraints and variables, and the solution of this restricted formulation is proven to be within a factor of ρ_k of an optimum solution, where $\rho_k \leq 1 + \frac{1}{\lfloor \log_2(k) \rfloor}$ [5]. A complete review of the *Hypergraphic* formulation and its variants can be found in Chakrabarty et al. [7].

Finally, in Dreyfus and Wagner [11], the authors propose a dynamic programming (DP) algorithm that finds an optimal solution for the Steiner tree problem in $\mathcal{O}(n^3 + n^2 2^b + n3^b)$ time, where $n = |V|$, $m = |A|$, and $b + 1 = |R|$. The result of this paper suggests that, for a fixed number of terminals, there exists a polynomial size LP formulation of the Steiner tree problem [25], which is the motivation for the results presented in this paper. The main drawback of this approach, common in DP algorithms, is that it only finds a solution when the algorithm terminates. For real-world size instances, this algorithm is not practical since its running time is exponential in the number of terminals.

Our main contribution is a set of independent IPs with the property that the best solution obtained by solving all of them, provides an optimal Steiner tree. Each IP is polynomial in the size of the underlying graph and our main result is that the LP relaxation of each IP is integral. Furthermore, the set of LPs can be solved in parallel. The drawback is that the number of LPs grows exponentially with the number of terminal nodes so the complete algorithm is polynomial only for a fixed number of terminals. But, in contrast to the DP approach proposed in Dreyfus and Wagner [11], each IP delivers a feasible solution to the problem, and therefore we get a new solution every time we solve a new IP, which implies that we may get good solutions by solving just a subset of the IPs.

The remainder of this paper is organized as follows. Section 2 analyzes the structure of Steiner trees, and also introduces definitions used in the paper. Section 3 presents the proposed model, the proof of integrality, and an analysis of the size of the problem. In Section 4 we present computational experiments obtained by solving the Steiner tree problem using our proposed formulation. Section 5 gives conclusions.

2 | SOLUTION STRUCTURE

As mentioned before, several formulations use the fact that every Steiner tree contains an r-arborescence, some of which use a flow-based formulation, and others a cut-based formulation. Both of those common approaches work with the directed graph version D of G. In our approach, we exploit two main properties that every solution must have.

1. We can always pick an arbitrary node $r \in R$ as our root node, and find the min-cost r-arborescence that spans $R \backslash \{r\}$. Moreover, this r-arborescence can be modeled as a min-cost multicommodity network flow, where we have one commodity for every node in $R \backslash \{r\}$, and we want to send 1 unit of flow from r to the corresponding node $i \in R \backslash \{r\}$.
2. Since the solution must correspond to an r-arborescence, there must be exactly one path from the root node r to each one of the nodes in $R \backslash \{r\}$, which means that if any subset S of the corresponding commodities of nodes in $R \backslash \{r\}$ share a path from r to node $i \in V$, and then split in node i, then they will never meet again. For example, if $S = \{k_1, k_2, k_3\}$ and in node i the set S splits into $S_1 = \{k_1, k_2\}$ and $S_2 = \{k_3\}$, then k_1 and k_2 will never share an arc again with k_3.

Using these two properties, our formulation is based on two type of decisions, (1) at which nodes we split the subsets of commodities, and (2) into which subsets of commodities. Note that all the commodities have the same source node, so they all start together. At some point, the set of all commodities will split into different subsets of commodities, and those subsets will also split into other subsets, and so on, until we have each commodity by itself. Our idea is to include a variable that tells us where a subset splits, and the partition that it uses. For instance, say that at some point we have the set $S = \{k_1, k_2, k_3\}$ of commodities that share a path. Eventually, S is split, so the model has to decide where the split is going to happen. In addition, the model has to decide the way that S is split. Note that there are 4 possible partitions of S.

1. $\{k_1, k_2, k_3\} \rightarrow \{\{k_1, k_2\}, \{k_3\}\}$
2. $\{k_1, k_2, k_3\} \rightarrow \{\{k_1, k_3\}, \{k_2\}\}$
3. $\{k_1, k_2, k_3\} \rightarrow \{\{k_2, k_3\}, \{k_1\}\}$
4. $\{k_1, k_2, k_3\} \rightarrow \{\{k_1\}, \{k_2\}, \{k_3\}\}$

Two observations are, first, we only consider proper partitions, and second, the next partition that we can use will depend on the way that we partitioned S. If we take, for instance, partition (1) then we will have to eventually partition the set $\{k_1, k_2\}$ into $\{k_1\}$, $\{k_2\}$, but if we take partition (4) then we do not have to perform any more partitions.

The algorithm proposed in Dreyfus and Wagner [11] uses a DP algorithm to choose the node where each partition occurs, and the way the sets are partitioned. Because of the latter type decisions, the running complexity of the DP algorithm depends on 3^b. In contrast, we propose to fix the way the sets are partitioned, and just decide the nodes where each partition is performed. To guarantee optimality, we need to solve the problem for every possible way the partitions can occur.

2.1 | Notation

In this section, we define the notation used in the rest of the paper. In Section 2.2, we provide an example to clarify the concepts introduced in this section.

We define $D = (V, A)$ as a directed graph such that, for every edge $\{i, j\}$ in E we have two arcs (i, j) and (j, i) in A, where V and E are the set of nodes and edges of the original undirected graph G, and where $c_a = c_e$ for all $a \in A$ such that both of its end nodes correspond to the end nodes of $e \in E$. Let $r \in R$ be an arbitrarily selected root node.

We define a set K of commodities, with one commodity for each of the nodes in $R\backslash\{r\}$, and let $b = |K|$. All of the commodities in K share the same source node r, and the sink node of commodity $k \in K$, denoted by t_k, is the corresponding terminal node in $R\backslash\{r\}$. Let S be the set of all subsets of K with at least one element, that is, $|S| = 2^{|K|} - 1$, and let P be the set of all proper partitions of the elements of S that have cardinality of at least 2.

The underlying r-arborescence of any Steiner tree uses only a subset of the elements of S, which contains the set of all commodities, and all the singletons. This collection of elements of S forms a laminar family. A family C of sets is called *laminar* if for every $A, B \in C$ we have $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. Note that laminar families of sets have been used in solving several combinatorial optimization problems; for further references see [31]. We define $\mathcal{L}_b$ to be the set of admissible laminar families that describe the structure of a Steiner tree with b commodities, where an admissible laminar family is one that contains the set of all commodities, and all the singletons. For laminar family $l \in \mathcal{L}_b$, let $S(l)$ be the set of elements in S that define laminar family l, and let $P(l)$ be the collection of partitions used to split the sets in $S(l)$.

For any $s' \in S(l)$ such that $|s'| \leq |K| - 1$, we say that $\hat{s}$ is its *parent* set, if s' is one of the sets obtained when we partition set $\hat{s}$ according to laminar family l. Note that there is only one parent set for every $s' \in S(l)$, $|s'| \leq |K| - 1$. Similarly, for any set $\hat{s} \in S(l)$ such that $|\hat{s}| \geq 2$, we say that s' is a *child* set of $\hat{s}$, if we obtain s' when we partition $\hat{s}$. Note that for every set $\hat{s} \in S(l)$, $|\hat{s}| \geq 2$ we have at least 2 child sets. Let $S_l(p)$ be the set of all child nodes of partition p in laminar family l, and let $P_l(s)$ be the partition p that splits s in laminar family l, which is defined only for sets s, such that $|s| \geq 2$. Finally, we say that a Steiner tree T in G follows an (r, l) *structure*, if the arborescence rooted in r follows the structure defined by laminar family $l \in \mathcal{L}_b$, where $b = |R\backslash\{r\}|$.

2.2 | Example

Suppose we have an instance with four terminal nodes. Let r be the root node, and let $K = \{k_1, k_2, k_3\}$. Then,

$$S = \left\{ \underbrace{\{k_1\}}_{s_1} ; \underbrace{\{k_2\}}_{s_2} ; \underbrace{\{k_3\}}_{s_3} ; \underbrace{\{k_1, k_2\}}_{s_4}; \underbrace{\{k_1, k_3\}}_{s_5}; \underbrace{\{k_2, k_3\}}_{s_6}; \underbrace{\{k_1, k_2, k_3\}}_{s_7} \right\}$$

and

$$P = \left\{ \underbrace{\{(k_1, k_2), (k_3)\}}_{p_1}; \underbrace{\{(k_1, k_3), (k_2)\}}_{p_2}; \underbrace{\{(k_2, k_3), (k_1)\}}_{p_3}; \underbrace{\{(k_1), (k_2), (k_3)\}}_{p_4} \right.$$
$$\left. \underbrace{\{(k_1), (k_2)\}}_{p_5}; \underbrace{\{(k_1), (k_3)\}}_{p_6}; \underbrace{\{(k_2), (k_3)\}}_{p_7} \right\}$$
$$= \left\{ \underbrace{\{s_4, s_3\}}_{p_1}; \underbrace{\{s_5, s_2\}}_{p_2}; \underbrace{\{s_6, s_1\}}_{p_3}; \underbrace{\{s_1, s_2, s_3\}}_{p_4}; \underbrace{\{s_1, s_2\}}_{p_5}; \underbrace{\{s_1, s_3\}}_{p_6}; \underbrace{\{s_2, s_3\}}_{p_7} \right\}$$

In this case p_1, p_2, p_3, and p_4 are the partitions of the set whose cardinality is 3, which is the set $\{k_1, k_2, k_3\}$. The partitions p_5, p_6, and p_7 are the partitions of sets of cardinality 2, which are the 3 sets $\{k_1, k_2\}$, $\{k_1, k_3\}$, and $\{k_2, k_3\}$.

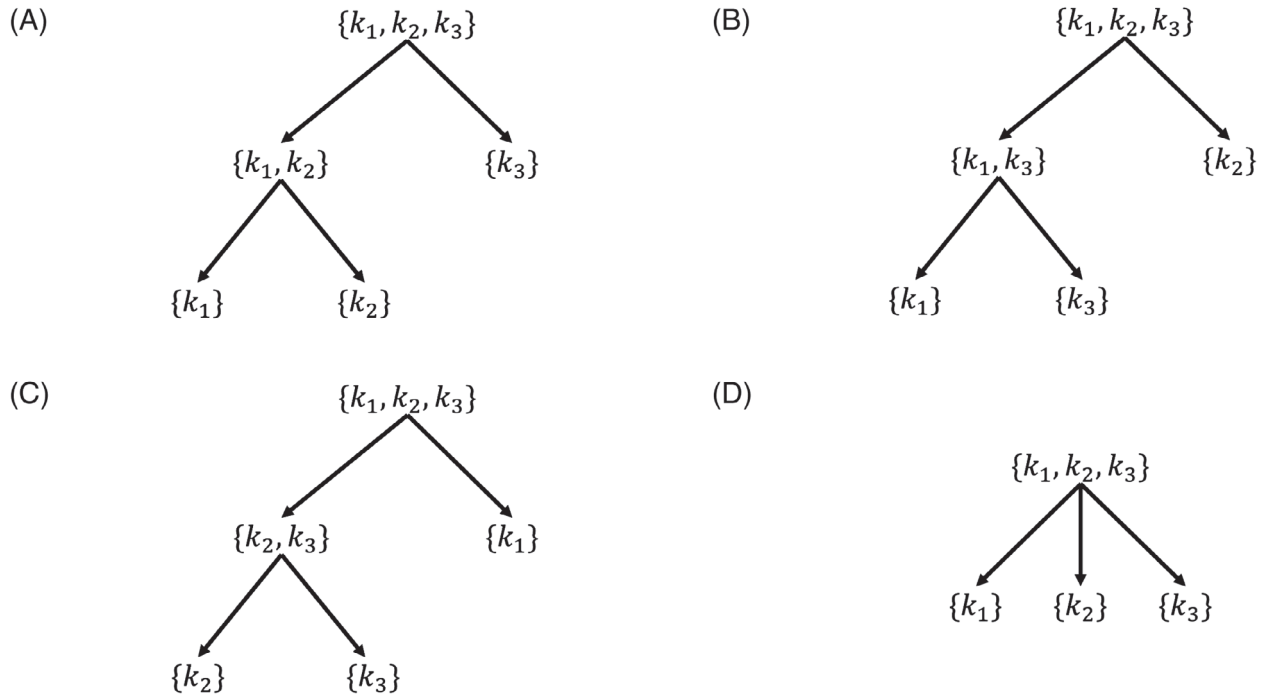

FIGURE 1 Tree representation of all possible laminar families for the case $K = \{k_1, k_2, k_3\}$. (A) Tree representation of laminar family $l_1 \in \mathcal{L}_3$. (B) Tree representation of laminar family $l_2 \in \mathcal{L}_3$. (C) Tree representation of laminar family $l_3 \in \mathcal{L}_3$. (D) Tree representation of laminar family $l_4 \in \mathcal{L}_3$

Figure 1 shows the tree representation of all the possible laminar families for the case $K = \{k_1, k_2, k_3\}$. In this case, we have

$$
\begin{aligned}
P(l_1) &= \{p_1, p_5\} \qquad S(l_1) = \{\{k_1, k_2, k_3\}; \{k_1, k_2\}; \{k_1\}; \{k_2\}; \{k_3\}\} = \{s_7, s_4, s_1, s_2, s_3\} \\
P(l_2) &= \{p_2, p_6\} \qquad S(l_2) = \{\{k_1, k_2, k_3\}; \{k_1, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\} = \{s_7, s_5, s_1, s_2, s_3\} \\
P(l_3) &= \{p_3, p_7\} \qquad S(l_3) = \{\{k_1, k_2, k_3\}; \{k_2, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\} = \{s_7, s_6, s_1, s_2, s_3\} \\
P(l_4) &= \{p_4\} \qquad S(l_4) = \{\{k_1, k_2, k_3\}; \{k_1\}; \{k_2\}; \{k_3\}\} = \{s_7, s_1, s_2, s_3\}
\end{aligned}
$$

Consider the laminar family $l_1 \in \mathcal{L}_3$ shown in Figure 1A. We have that $\{k_1, k_2, k_3\}$ is the *parent* set of $\{k_1, k_2\}$ and $\{k_3\}$, and that $\{k_1, k_2\}$ is the *parent* set of $\{k_1\}$ and $\{k_2\}$. On the other hand, sets $\{k_1\}$ and $\{k_2\}$ are the *child* sets of $\{k_1, k_2\}$, and, set $\{k_1, k_2\}$ and $\{k_3\}$ are *child* sets of $\{k_1, k_2, k_3\}$. Finally, we have

$$
\begin{aligned}
S_{l_1}(p_1) &= \{\{k_1, k_2\}; \{k_3\}\} = \{s_4, s_3\} \\
S_{l_1}(p_5) &= \{\{k_1\}; \{k_2\}\} = \{s_1, s_2\} \\
P_{l_1}(\{k_1, k_2, k_3\}) &= P_{l_1}(s_7) = \{p_1\} \\
P_{l_1}(\{k_1, k_2\}) &= P_{l_1}(s_4) = \{p_5\}
\end{aligned}
$$

3 | IP MODEL

We propose an IP model $\mathcal{Z}_l$ that finds an optimal solution for a given laminar family $l \in \mathcal{L}_b$, where $b = |K|$. Since we are considering a fixed laminar family l, we are given $S(l)$, the set of subsets of K that we use, and the partitions that are performed, as well as the *parent* and *child* sets of each set in $S(l)$. Therefore, the main purpose of our model is to choose the node where each partition is performed. Once we know where each partition is performed, it is easy to determine the arcs that minimize cost, since the problem reduces to finding a shortest path between the nodes where each subset $s \in S(l)$ begins and ends.

For notational simplicity, we do not index the variables of $\mathcal{Z}_l$ by l. Our proposed formulation $\mathcal{Z}_l$ is

$$
\begin{aligned}
\mathcal{Z}_l : \qquad & \min \sum_{a \in A} c_a \left(\sum_{s \in S(l)} f_a^s \right), \\
& \sum_{a \in \delta^+(i)} f_a^s - \sum_{a \in \delta^-(i)} f_a^s = \widehat{y}_i^s - \overline{y}_i^s \qquad \forall i \in V,\ s \in S(l), \qquad (1)
\end{aligned}
$$

$$w_i^p = \overline{y}_i^s \qquad \forall i \in V,\ s \in S(l) : |s| \geq 2,\ p = P_l(s), \tag{2}$$

$$w_i^p = \widehat{y}_i^s \qquad \forall i \in V,\ p \in P(l),\ s \in S_l(p), \tag{3}$$

$$\sum_{i \in V} w_i^p = 1 \qquad \forall p \in P(l), \tag{4}$$

$$\widehat{y}_r^K = 1, \tag{5}$$

$$\widehat{y}_i^K = 0 \qquad \forall i \in V \backslash \{r\}, \tag{6}$$

$$\overline{y}_{t_k}^k = 1 \qquad \forall k \in K, \tag{7}$$

$$\overline{y}_i^k = 0 \qquad \forall k \in K,\ i \in V \backslash \{t_k\}, \tag{8}$$

$$f \in \{0,1\}^{|A| \times |S(l)|}, \tag{9}$$

$$(\widehat{y}, \overline{y}) \in \{0,1\}^{|V| \times |S(l)|}, \tag{10}$$

$$w \in \{0,1\}^{|V| \times |P(l)|}. \tag{11}$$

The variable f_a^s is 1 if we send flow in arc a for subset s, and 0 otherwise. This means that all of the commodities that compose subset s use arc a. The variable $\widehat{y}_i^s$ is 1 if commodities in set *s start* sharing a path in node i but commodities in parent set s' do not, and 0 otherwise. The variable $\overline{y}_i^s$ is 1 if commodities in set *s end* sharing a path in node i, and 0 otherwise. Finally, w_i^p is 1 if a partition p is performed in node i, and 0 otherwise. Constraint (1) is the flow conservation constraints for all the subsets that belong to laminar family l, which relates variables f, $\widehat{y}$ and $\overline{y}$. Constraint (2) states that if partition p is performed in node i, then the subset s that is partitioned has to stop sharing in node i. Constraint (3) states that if partition p occurs in node i, then the *child* sets of partition p start to share in node i. Constraint (4) imposes that the partitions can occur only in one node. Constraints (5) and (6) say that the set of all commodities start sharing in the root node. Finally, constraints (7) and (8) state that every commodity has to send its flow to its sink node.

The LP relaxation of $\mathcal{Z}_l$, which we refer to as $LP(\mathcal{Z}_l)$, is defined by the same objective function and set of linear equalities, but relaxing the integrality condition of the variables. This means that we replace (9) to (11) with

$$f \in [0,1]^{|A| \times |S(l)|}, \tag{12}$$

$$(\widehat{y}, \overline{y}) \in [0,1]^{|V| \times |S(l)|}, \tag{13}$$

$$w \in [0,1]^{|V| \times |P(l)|}. \tag{14}$$

For all $l \in \mathcal{L}_b$, let $\mathcal{Q}_l^{IP}$ and $\mathcal{Q}_l^{LP}$ be the feasible regions of $\mathcal{Z}_l$ and $LP(\mathcal{Z}_l)$ respectively. For all $e \in E$, let $A(e) = \{a \in A : a$ is a directed arc of $e \in E\}$, and let $\chi \in \{0,1\}^{|E|}$. χ represents a solution to the Steiner tree problem in G, where $\chi_e = 1$ if edge e belongs to the solution.

Definition. Let $l \in \mathcal{L}_b$, and let $x = (f, w, \widehat{y}, \overline{y})$ be a feasible solution for $\mathcal{Q}_l^{IP}$. We define ϕ to be a mapping of x to the original problem space, where

$$\phi(x) = \chi \in \{0,1\}^{|E|}$$

is given by

$$\chi_e = \min \left\{ 1, \sum_{s \in S(l)} \sum_{a \in A(e)} f_a^s \right\} \qquad \text{for all } e \in E.$$

3.1 | Structure of feasible solutions of $\mathcal{Z}_l$

We assume that $\mathcal{Q}_l^{IP}$ is not empty, and let $x = (f, w, \widehat{y}, \overline{y})$ be an arbitrary feasible solution for $\mathcal{Z}_l$. A feasible solution can have two components, it must have an acyclic component, denoted by x_{nc}, and it may also have a cycle component, denoted by x_c. Then, $x = x_{nc} + x_c$. The acyclic component is always present and is a feasible solution to the problem, while the cycle component may not be present, that is, we may have $x_c = 0$. The cycle component is not feasible on its own, since it is only composed of cycles, and therefore it only fulfills the flow constraints (1). We analyze each component separately.

First, consider the acyclic component, $x_{nc} = (f_{nc}, w_{nc}, \widehat{y}_{nc}, \overline{y}_{nc})$. Since this component has to be feasible, then by constraints (2) to (8), $\widehat{y}_{nc}^s$ and $\overline{y}_{nc}^s$ have exactly one nonzero component for all $s \in S(l)$, which we call i_s and j_s respectively. There are two cases, either $i_s \neq j_s$, or $i_s = j_s$. In the first case, because of constraint (1), there must be a path for s from i_s to j_s. In the second case, since x_{nc} has no cycles, $f_{nc}^s = 0$, or equivalently the set s has no path. On the other hand, constraints (2) to (4) ensure that the path of every parent set ends in the same node where the path of its child sets start. By constraints (5) and (6) we have that the set of all commodities must start in r, and by constraints (7) and (8), each singleton k must end its path in t_k. Therefore, in every feasible solution, there is a path from r to t_k for all k in K, which corresponds to the union of the paths of all sets containing

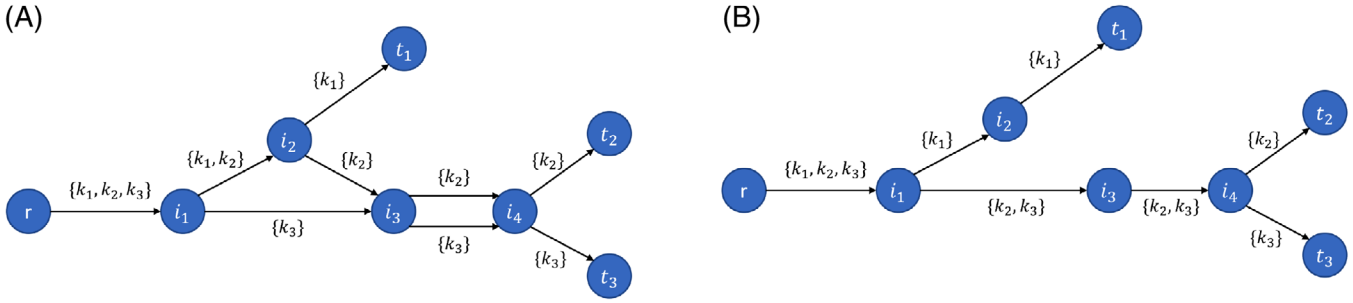

FIGURE 2 Feasible solution x of $\mathcal{Z}_{l_1}$, whose mapping $\phi(x)$ is not an (r, l_1) structured Steiner tree, and (r, l_3) structured Steiner tree contained in support of $\phi(x)$. (A) Feasible solution x for $\mathcal{Z}_{l_1}$. (B) Steiner tree with (r, l_3) structure, contained in support of $\phi(x)$ [Color figure can be viewed at wileyonlinelibrary.com]

k. Consequently, all Steiner trees with (r, l) structure have a corresponding acyclic feasible solution to $\mathcal{Z}_l$, but not all acyclic solutions of $\mathcal{Z}_l$ map to an (r, l) structured Steiner tree. We will discuss the case in which a feasible acyclic solution is not an (r, l) structured Steiner tree in Proposition 1.

Now, consider the cycle component. Let $x_c = (f_c, w_c, \hat{y}_c, \bar{y}_c) \neq 0$, that is, there is at least one set $\tilde{s} \in S(l)$ which has at least one cycle, and for simplicity of the argument, suppose that $\tilde{s}$ has only one cycle. The only way that this can happen is when the cycle uses arcs that are not in the path between $i_{\tilde{s}}$ and $j_{\tilde{s}}$, the path used by $\tilde{s}$ in the noncycle component, otherwise it would be an infeasible solution because it would violate constraints (1) and (9). In the cycle component, w_c, $\hat{y}_c$ and $\bar{y}_c$ are always 0, and if $x_c \neq 0$ then $f_c \neq 0$. Consequently, we have $w = w_{nc}$, $\hat{y} = \hat{y}_{nc}$, $\bar{y} = \bar{y}_{nc}$, and $f = f_{nc} + f_c$. If a solution does not contain cycles, then $f = f_{nc}$.

Proposition 1. *For any $l \in \mathcal{L}_b$, let x be a feasible solution of $\mathcal{Z}_l$ that does not contain any cycles. Then $\phi(x)$ is either an (r, l) structured Steiner tree, or the support of $\phi(x)$ contains a Steiner tree with a different (r, l) structure.*

Proof. As was pointed out in Section 3.1, in the noncycle component of every feasible solution x of $\mathcal{Z}_l$, there is a path from r to t_k for every commodity $k \in K$. This is a property of every Steiner tree with an (r, l) structure. But, there are also other cases where a feasible solution x of $\mathcal{Z}_l$ does not map to an (r, l) structured Steiner tree, as shown in Figure 2. We observe in Figure 2A, a feasible solution x of $\mathcal{Z}_{l_1}$ whose mapping $\phi(x)$ is not an (r, l_1) structured Steiner tree.[1] Note that commodities k_2 and k_3 share arcs, but not using set $\{k_2, k_3\}$ because this set is not in l_1.

Now, suppose $x \in Q_l^{IP}$ is acyclic and does not map to an (r, l) structured Steiner tree. The mapping $\phi(x)$ tells us which edges of the original undirected graph G are used by x. We can construct a subgraph defined by those edges, and we know there must be a Steiner tree in it, because in solution x there is at least one path from r to t_k for all $k \in K$. Then, we can take any Steiner tree within the constructed subgraph, which will have a unique (r, l^*) structure, with l^* not necessarily equal to l. We can do this by taking any spanning tree in the constructed subgraph. For instance, Figure 2B shows an (r, l_3) structured Steiner tree within the support of $\phi(x)$ shown in Figure 2B. ∎

3.2 | Structure of feasible solutions of $LP(\mathcal{Z}_l)$

The structure of the feasible solutions of $LP(\mathcal{Z}_l)$ is similar to the solutions of $\mathcal{Z}_l$, because they also have a noncycle, and a cycle component. For simplicity, we use the same notation as in Section 3.1. We assume that Q_l^{LP} is not empty, and let $x = (f, w, \hat{y}, \bar{y})$ be an arbitrary solution of $LP(\mathcal{Z}_l)$. Let $x = x_{nc} + x_c$, where x_{nc} is the noncycle component of x, and x_c is the cycle component of x.

The analysis for this case is similar to the analysis in Section 3.1, but since the variables now can be fractional, we may have solutions where the noncycle and the cycle component use the same arc a for the same set s, but the sum of acyclic and cyclic components has to be at most 1 for all arcs and for all sets. Moreover, in the cycle component, we may have that two different cycles, for the same s, use the same arc a. Thus, the cycle part will be a collection of weighted cycles for each $s \in S(l)$, such that for every arc $a \in A$, the sum of acylic and cyclic flow in a for s is at most 1.

Let $LP(\mathcal{Z}_l)(\lambda)$ denote the formulation $LP(\mathcal{Z}_l)$, when we replace 1 by $\lambda \in (0, 1]$ in constraints (4), (5), (7), (12), (13), and (14). Let $OPT(\lambda)$ denote the value of an optimal solution to $LP(\mathcal{Z}_l)(\lambda)$, for $\lambda \in (0, 1]$.

Lemma 2. *For every $l \in \mathcal{L}_b$ and $\lambda \in (0, 1]$, we have $OPT(\lambda) = \lambda[OPT(1)]$.*

[1] See Figure 1 for a description of laminar families l_1 and l_3.

Proof. For any $\lambda \in (0, 1]$, let $(f^\lambda, \hat{y}^\lambda, \overline{y}^\lambda, w^\lambda)$ denote an optimal solution for $LP(\mathcal{Z}_l)(\lambda)$. Then,

- $OPT(\lambda) \leq \lambda[OPT(1)]$. Since $(\lambda f^1, \lambda \hat{y}^1, \lambda \overline{y}^1, \lambda w^1)$ is a feasible solution to $LP(\mathcal{Z}_l)(\lambda)$, it holds that $OPT(\lambda) \leq \lambda[OPT(1)]$.
- $OPT(\lambda) \geq \lambda[OPT(1)]$. Since $\left(\frac{1}{\lambda} f^\lambda, \frac{1}{\lambda} \hat{y}^\lambda, \frac{1}{\lambda} \overline{y}^\lambda, \frac{1}{\lambda} w^\lambda\right)$ is a feasible solution to $LP(\mathcal{Z}_l)(1)$, it holds that $\frac{1}{\lambda}[OPT(\lambda)] \geq OPT(1)$

Then, $OPT(\lambda) = \lambda[OPT(1)]$ for all $\lambda \in (0, 1]$. ∎

Our main result is stated in the following theorem.

Theorem 3. *For any $l \in \mathcal{L}_b$, we have* $\text{Conv}(\mathcal{Q}_l^{IP}) = \mathcal{Q}_l^{LP}$.

Proof. We will prove that for all cost vectors c, there exists an optimal solution to $LP(\mathcal{Z}_l)$ that is integral.

The following definitions will be used in the proof. For all $i \in V$, $s \in S(l)$, let $LP(\mathcal{Z}_l(i, s))$ be a subformulation of $LP(\mathcal{Z}_l)$, where the root is i (instead of r), the terminal nodes are t_k for all $k \in s$, and the splitting sequence is defined by the way s, and its subsets, are split in laminar family l. The cost vector of $LP(\mathcal{Z}_l(i, s))$ is the same as the one used in $LP(\mathcal{Z}_l)$, but only considering the components of $LP(\mathcal{Z}_l)$ that are present in $LP(\mathcal{Z}_l(i, s))$. Let $x^*(i, s)$ be an optimal solution to $LP(\mathcal{Z}_l(i, s))$.

Let $x^* = (f^*, \hat{y}^*, \overline{y}^*, w^*)$ be an optimal solution of $LP(\mathcal{Z}_l)$. Note that if w^* is integer, then $\hat{y}^*$ and $\overline{y}^*$ are integer because of constraints (2) and (3). This implies that for all $s \in S(l)$, there is only one nonzero component in vectors $\hat{y}^s$ and $\overline{y}^s$, which is equal to 1. Let i_s and j_s be those indexes, respectively. Now, for all $s \in S(l)$, constraints (1) and (9) correspond to the shortest $i_s - j_s$ path polytope, which has only integer extreme points. Since each of these polytopes is independent of each other, we have that f^* is integer. Therefore, if w^* is integer, then x^* is integer.

Now, suppose that w^* is fractional. This implies that for at least one set $s \in S(l)$, $\overline{y}^s$ is fractional. Note that we may have that $\hat{y}^s$ is also fractional, but for at least one $s \in S(l)$, we claim that $\hat{y}^s$ is integer, and $\overline{y}^s$ is fractional. The justification for this claim is the following. Suppose the partitions in $P(l)$ are ordered in decreasing order by the cardinality of the set they split, that is, the first element of $P(l)$ splits K, and the last set of $P(l)$ splits a set of cardinality 2. Now, consider w^*, and let $\dot{p}$ be the first element in $P(l)$ such that $w^{\dot{p}}$ is fractional. Let $\dot{s}$ be the set that $\dot{p}$ splits. Then, it is clear that $\hat{y}^{\dot{s}}$ is integer and $\overline{y}^{\dot{s}}$ is fractional.

Let $\hat{s}$ be the largest set in $S(l)$ such that $\hat{y}^{\hat{s}}$ is integral, and $\overline{y}^{\hat{s}}$ is fractional, and for simplicity of the argument, suppose $\hat{s} = K$. Let $I(\hat{s})$ be the set of indexes such that $\overline{y}_i^{\hat{s}} = \lambda_i > 0$ for all $i \in I(\hat{s})$, and $\sum_{i \in I(\hat{s})} \lambda_i = 1$. Let $\hat{p} \in P(l)$ be the partition that splits $\hat{s}$ in laminar family l. By constraint (2) we have $w_i^{\hat{p}} = \lambda_i$ for all $i \in I(\hat{s})$, as shown in Figure 3.

For all $i \in I(\hat{s})$ and $s \in S_l(\hat{p})$, by constraint (3), we have $\hat{y}_i^s = \lambda_i$. Now, take a fixed $s \in S_l(\hat{p})$, and a fixed $i \in I(\hat{s})$. Because $\hat{y}_i^s = \lambda_i$, then we know that within solution x^*, we are sending λ_i units of flow from node i to $\{t_k\}_{k \in s}$. Since x^* is optimal, then by Lemma 2, that part of the solution has to correspond to $\lambda_i x^*(i, s)$, otherwise, we can improve solution x^*. On the other hand, within solution x^* we are sending λ_i units of flow from r to i, for all $i \in I(\hat{s})$. Let $x^*(r, i, \hat{s})$ be the lowest cost solution sending 1 unit of flow from r to $i \in I(\hat{s})$ for set $\hat{s}$, then, by Lemma 2, $\lambda_i x^*(r, i, \hat{s})$ corresponds to the lowest cost solution sending λ_i units of flow from r to $i \in I(\hat{s})$ for set $\hat{s}$, otherwise we can improve solution x^*. Figure 4 shows a representation of the last statement.

Putting everything together, we have

$$x^* = \sum_{i \in I(\hat{s})} \lambda_i \left[x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s) \right]$$

But, for all $i \in I(\hat{s})$, $x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s)$ is a feasible solution. Then, x^* is a convex combination of $\left\{ x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s) \right\}_{i \in I(\hat{s})}$, and therefore, it is not an extreme point. Note that, $x^*(r, i, \hat{s})$ and $\sum_{s \in S_l(\hat{p})} x^*(i, s)$ may contain cycles. Furthermore, if $\hat{s} \neq K$, we can use the same argument, but we have to consider the solution of all the sets that contain $\hat{s}$ as a subset. By definition of $\hat{s}$, in the solution of all those sets, $\hat{y}$ and $\overline{y}$ will be integers, and therefore, the entire solution of all those sets has to be integer. Thus, we can include that part of the solution in $x^*(r, i, \hat{s}) + \sum_{s \in S_l(\hat{p})} x^*(i, s)$ for all $i \in I(\hat{s})$, and use the same argument as before. Consequently, all extreme point solutions have $\hat{y}, \overline{y}$, and w integral, and therefore, all extreme points of $\mathcal{Q}_l^{LP}$ are integral. ∎

As a corollary, we can solve the entire Steiner tree problem by optimizing each laminar family subproblem independently, and taking the solution with the lowest cost.

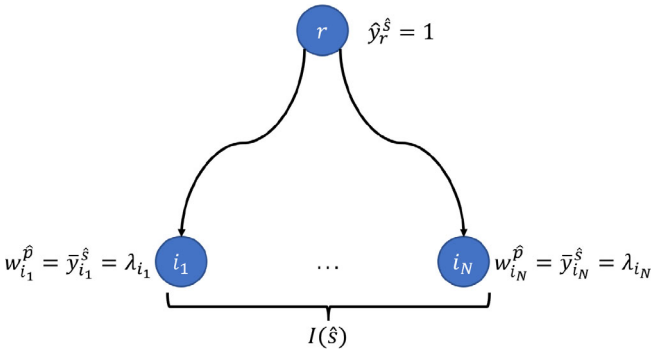

FIGURE 3 Solution illustration [Color figure can be viewed at wileyonlinelibrary.com]

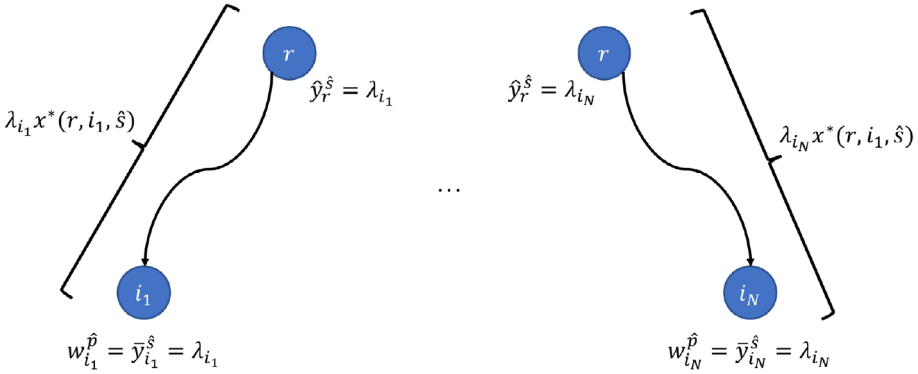

FIGURE 4 Solution decomposition illustration [Color figure can be viewed at wileyonlinelibrary.com]

Corollary 4. *Let $c > 0$ be the cost vector of a Steiner tree problem instance. Let $x^l \in \text{Argmin}\{c^T x : x \in Q_l^{LP}\}$, and $v^l = c^T x^l$ for all $l \in \mathcal{L}_b$. Then, $\phi(x^{l^*})$ is a minimum cost Steiner tree, where $l^* \in \text{Argmin}\{v^l : l \in \mathcal{L}_b\}$.*

Proof. Note that, since the cost is positive, then the optimal solution for each subproblem is acyclic. By Proposition 1, we know that $\phi(x^{l^*})$ is either an (r, l^*) structured Steiner tree, or $\phi(x^{l^*})$ contains a Steiner tree, with a different structure. But, since the cost vector is nonnegative, then $\phi(x^{l^*})$ has to be a Steiner tree, otherwise, we can take any Steiner tree contained in the support of $\phi(x^{l^*})$, which will have a lower cost since it uses a subset of the edges used by $\phi(x^{l^*})$. Therefore, $\phi(x^{l^*})$ is a minimum cost Steiner tree. ∎

Note that using a standard disjunctive programming argument, we can write an extended formulation for the entire problem based on the formulations for each laminar family [1, 9].

An important remark is that, although we have done all the analysis to solve the undirected Steiner tree problem, this approach can also be used to solve the directed Steiner tree problem, which is a more difficult problem to solve. For the directed version of the problem, the main difference is that we are given the directed graph $D = (V, A)$ and the root node r. The rest of the analysis remains the same.

3.3 | Problem size

The downside of our proposed formulation is the number of laminar families, which grows very fast as the number of terminals increases. In this section, we provide an upper bound on the number of laminar families that we have to consider, and we show that this upper bound is tight.

In the following proposition, we use the notion of a *full binary* tree. Note that a tree T is called a *full binary* tree if every node in T has either zero or two children.

Proposition 5. *To solve the Steiner tree problem to optimality, it suffices to consider only laminar families in $\mathcal{L}_b$ whose tree representation corresponds to a full binary tree.*

Proof. Let l_1 be a laminar family in $\mathcal{L}_b$ and let $T(l_1)$ be the tree representation of l_1, such that $T(l_1)$ is not a full binary tree. Therefore, there must be a node in $T(l_1)$ that has at least three child nodes, which implies that there must

be a set in l_1 that is partitioned into at least three sets; let $\hat{s}$ be such set. Suppose that the *child* sets of $\hat{s}$ in l_1 are s_1, s_2, …, s_t with $t \geq 3$. For simplicity, assume $t = 3$. Now, let l_2 be a laminar family in $\mathcal{L}_b$ such that $S(l_2) = S(l_1) \cup s'$ where $s' = s_1 \cup s_2$. This means that all partitions in l_1 and l_2 are the same, but in l_1, $\hat{s}$ is partitioned into s_1, s_2, s_3, and in l_2, $\hat{s}$ is partitioned into s' and s_3, and s' is partitioned into s_1 and s_2. Now, let $x_1 = (f_1, w_1, \hat{y}_1, \bar{y}_1)$ be an optimal solution to $\mathcal{Z}_{l_1}$. We claim that we can construct a feasible solution to $\mathcal{Z}_{l_2}$ using x_1. Let i^* be the node where $\hat{s}$ is split in x_1. Then we define $x_2 = (f_2, w_2, \hat{y}_2, \bar{y}_2) \in Q^{LP}_{l_2}$ as follows

$$(f_2)^s_a = \begin{cases} (f_1)^s_a & \forall a \in A, \ \ s \in S(l_1) \\ 0 & \forall a \in A, \ \ s = s' \end{cases}$$

$$(\hat{y}_2)^s_i = \begin{cases} (\hat{y}_1)^s_i & \forall i \in V, \ \ s \in S(l_1) \\ 1 & i = i^*, \ \ s = s' \\ 0 & \forall i \in V \backslash \{i^*\}, \ \ s = s' \end{cases}$$

$$(\bar{y}_2)^s_i = \begin{cases} (\bar{y}_1)^s_i & \forall i \in V, \ \ s \in S(l_1) \\ 1 & i = i^*, \ \ s = s' \\ 0 & \forall i \in V \backslash \{i^*\}, \ \ s = s' \end{cases}$$

Note that the values of the w_2 vector will be determined by the values of $\hat{y}_2$ and $\bar{y}_2$. Moreover, x_2 is a feasible solution to $\mathcal{Z}_{l_2}$, whose cost is the same as the cost of x_1. Hence, $\mathcal{Z}_{l_2} \leq \mathcal{Z}_{l_1}$. The same argument holds for $t > 3$ as well. Consequently, we only need laminar families where each nonsingleton set is partitioned into two proper subsets, which correspond to laminar families whose tree representation is a full binary tree. ∎

Using Proposition 5, we can reduce the number of laminar families by considering only laminar families whose tree representations are full binary trees. For simplicity of notation, we denote by $\mathcal{L}_b$ the reduced set of laminar families, when we have $b + 1$ terminals. We are interested in expressing $|\mathcal{L}_b|$ as a function of b. Note that $|\mathcal{L}_b|$ is equivalent to the number of full binary labeled trees with b leaves. It is known that (see [4] and [[32], Chapter 5.2.6])

$$|\mathcal{L}_b| = (2b-3)!! = \frac{(2(b-1))!}{2^{b-1}(b-1)!}$$

Note, however, that $|\mathcal{L}_b|$ is not always achieved, since there are graphs that do not yield some laminar families of $\mathcal{L}_b$, which is why $\frac{(2(b-1))!}{2^{b-1}(b-1)!}$ is an upper bound on the number of laminar families to consider. Now, recall Stirling's formula, which gives lower and upper bounds for $n!$. For any $n \in \mathbb{Z}^+$ we have

$$\sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n} \leq n! \leq e n^{n+\frac{1}{2}} e^{-n}$$

Then,

$$\begin{aligned} |\mathcal{L}_b| &= \frac{(2(b-1))!}{2^{b-1}(b-1)!} \\ &\leq \frac{e(2(b-1))^{2(b-1)+\frac{1}{2}} e^{-2(b-1)}}{2^{b-1}\sqrt{2\pi}(b-1)^{(b-1)+\frac{1}{2}} e^{-(b-1)}} \\ &= \frac{e 2^{(b-1)} (b-1)^{(b-1)} e^{-(b-1)}}{\sqrt{\pi}} \\ &= \frac{e}{\sqrt{\pi}} \left(\frac{2(b-1)}{e} \right)^{(b-1)} \end{aligned}$$

We conclude that $|\mathcal{L}_b|$ grows as $\mathcal{O}\left(\left(\frac{2b}{e} \right)^b \right)$. Although this is a big number, it is small compared to the total number of laminar families. In fact, it can be shown that as b increases, the ratio between the number of laminar families corresponding to full binary trees and the total number of admissible laminar families, goes to 0.

Now, we analyze the size of each subproblem $LP(\mathcal{Z}_l)$ for $l \in \mathcal{L}_b$. By Proposition 5, we use laminar families whose tree representation $T(l)$ is a full binary tree. An important property of full binary trees is that the number of internal nodes is $L - 1$, where L is the number of leaves. In our case, there are b leaves, which implies $b - 1$ internal nodes, and $2b - 1$ nodes in the entire tree. This implies that $|S(l)| = 2b - 1$ for all $l \in \mathcal{L}_b$. Moreover, the number of nonsingleton sets corresponds to the number of internal nodes in $T(l)$ which is $b - 1$, and also is the number of partitions in l. Finally, since we only consider l such that $T(l)$ is a full binary tree, the number of *child* sets of each partition is exactly 2. Using these facts, the number of variables and

TABLE 1 Number of terminals, maximum number of laminar families, and total execution time if each subproblem takes 1 second to solve

Number of terminals	$\lvert\mathcal{L}_b\rvert$	Running time
3	1	1 s
4	3	3 s
5	15	15 s
6	105	1.75 m
7	945	15.75 m
8	10 395	2.89 h
9	135 135	37.54 h
10	2 027 025	23.46 d
11	34 459 425	1.09 y

constraints are the same for all $l \in \mathcal{L}_b$, and the number of variables is $\mathcal{O}(nb + mb)$ and the number of constraints is $\mathcal{O}(nb + b)$, where $n = |V|$, $m = |A|$ and $b + 1 = |R|$. Therefore, $LP(\mathcal{Z}_l)$ is of polynomial size in the input data. Consequently, we have that each subproblem is polynomially solvable, but the number of subproblems grows as $\mathcal{O}\left(\left(\frac{2b}{e}\right)^b\right)$, which implies that the problem is fixed-parameter tractable with respect to the number of terminal nodes. This last remark is consistent with previous results [11, 15].

4 | COMPUTATIONAL RESULTS

We provide computational results to evaluate the performance of our proposed formulation. These results are presented just to validate our model. At the current state of the proposed approach, the results are not competitive with the state of the art solvers for the unidirected and directed Steiner tree problems [10, 12, 16, 20, 22, 27, 28, 34].

We use instances from the SteinLib library [23]. Table 1 shows the number of laminar families as a function of the number of terminal nodes, and it shows how long it will take to solve the entire problem if each subproblem takes one second to execute, and they are solved sequentially.

As we can see from Table 1, the expected time to run instances with nine or more terminals will be very large, even if the execution time for each subproblem is 1 second. Thus, we have only run experiments on instances with at most eight terminal nodes.

Our code is implemented in *Python* using *Gurobi* 7.5.1 as a solver. All tests were performed on a laptop with an *Intel Core i7* (3.3 GHz) processor, and 16 GB of memory, using the *macOS Sierra* operating system. As we previously discussed, we can solve each laminar family subproblem independently, and therefore, they can be solved in parallel, but we implemented our code such that we run each subproblem sequentially.

Table 2 summarizes the results. The first two columns correspond to the test set and instance name, the third column shows the number of nodes, the fourth column shows the number of arcs, the fifth and the sixth columns present the number of terminal nodes and total number of laminar families, respectively. Finally, the last two columns present the average execution time for each subproblem, and the total execution time for all the models, respectively. All the reported times are in seconds.

In order to show the potential of the proposed approach, we present results on larger instances, which are not meant to be solved to optimality, but to obtain a good solution within seconds. As discussed, the main drawback of the proposed approach is that, in order to prove optimality, one must solve the entire set of tree structures. Nevertheless, we may want to solve the problem for a restricted number of structures, and take the best solution among them. This process will not guarantee optimality, but may be useful to have good quality solutions within seconds. The idea is to determine good candidate structures to be solved. There are many ways to do this, and some ideas are discussed in Section 5, but we decided to propose a simple way to solve the problem for only one candidate structure, and compare its value with the value of an optimal solution.

The process to construct the structure to solve is the following. First, we arbitrarily select a root node r. Second, we create a complete graph G' whose nodes are the terminal nodes excluding r. For every edge $e' = (t', t'')$, its cost is the length of the shortest path between t' and t'' in the original graph G. Then, we construct a minimum spanning tree T' in G'. Finally, we run Algorithm 1 to construct a laminar family using T' as input.

TABLE 2 Instance details and execution times in seconds

Test set	Instance	$\|V\|$	$\|A\|$	$\|R\|$	$\|\mathcal{L}_b\|$	Subproblem time (s)	Entire problem time (s)
LIN	*lin01*	53	160	4	3	0.003	0.01
LIN	*lin02*	55	164	6	105	0.005	0.48
LIN	*lin04*	157	532	6	105	0.021	2.18
LIN	*lin07*	307	1052	6	105	0.060	6.26
I160	*i160-033*	160	640	7	945	0.016	15.43
I160	*i160-043*	160	5088	7	945	0.151	142.63
I160	*i160-045*	160	5088	7	945	0.164	154.74
LIN	*lin03*	57	168	8	10 395	0.007	71.26
PUC	*cc3-4p*	64	576	8	10 395	0.018	190.89
PUC	*cc3-4u*	64	576	8	10 395	0.019	192.65
I320	*i320-011*	320	3690	8	10 395	0.197	1806.00
I320	*i320-043*	320	20 416	8	10 395	0.910	9468.00

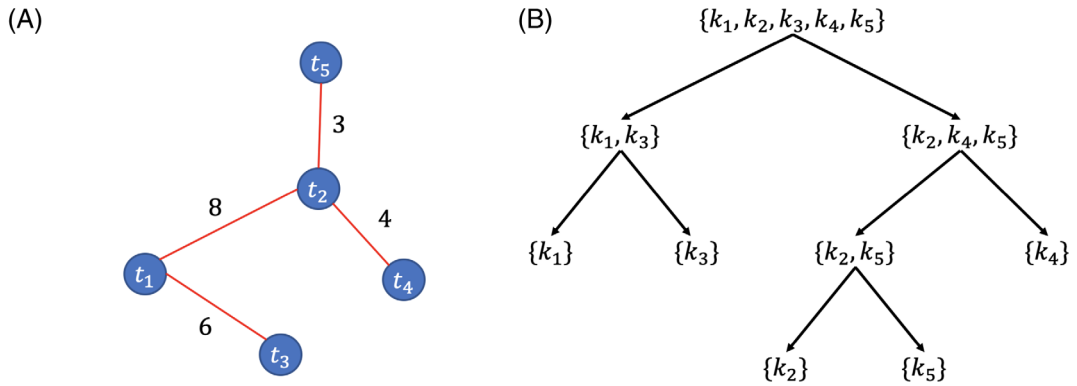

FIGURE 5 Explanatory example of Algorithm 1. (A) Minimum spanning tree T' of graph G'. (B) Tree representation of laminar family constructed from T' [Color figure can be viewed at wileyonlinelibrary.com]

Algorithm 1. Algorithm to create candidate laminar family l.

Require: Tree T', we assume T' is a set of edges ordered in increasing order of their lengths.
1: Set $S(l) \leftarrow \{\{k_1\}, \{k_2\}, \ldots, \{k_b\}\}$
2: **for** $e' \in T'$
3: Let $e' = (t', t'')$, and let k' and k'' be the commodities of terminals t' and t'', respectively.
4: Let s' be largest set in $S(l)$ containing k'
5: Let s'' be largest set in $S(l)$ containing k''
6: $\hat{s} \leftarrow s' \cup s''$
7: $S(l) \leftarrow S(l) \cup \hat{s}$
return $S(l)$.

Algorithm 1 constructs a laminar family l based on the single linkage clustering algorithm. It first starts with the collection $S(l)$ of all the singletons. Then, it goes over the set of all edges of T', which are ordered in increasing order of their lengths. For each edge $e' = (t', t'')$, we take the commodities k' and k'' of terminals t' and t'', respectively. Then, we take sets s' and s'', which are the largest sets in $S(l)$ that contain k' and k'', respectively. Finally, we create a set $\hat{s}$ to be the union of s' and s'', and we add it to $S(l)$. After we visit all edges, we have that $S(l)$ is a collection of sets that forms an admissible laminar family. Figure 5 shows an example to understand the laminar construction algorithm. Figure 5a shows tree T' in G', and Figure 5b shows the tree representation of the laminar family constructed using Algorithm 1.

We take four set of instances from the SteinLib that are difficult to solve since they are built to defy preprocessing. The instances chosen were I080, I160, I320, and I640. We only take instances with at least 17 terminal nodes whose optimal value is known. In total, we solve 220 instances. Figure 6 shows the performance profile of the proposed approach for each one of the set of instances. We show the performance profile of each set separately and also show the performance profile of all 4 set of instances, which we refer as *iSeries* in Figure 6. To explain the graph, take, for instance, the point (10, 0.36) of the red curve. This means that for the test set I080, 36% of the instances have an error of at most 10%.

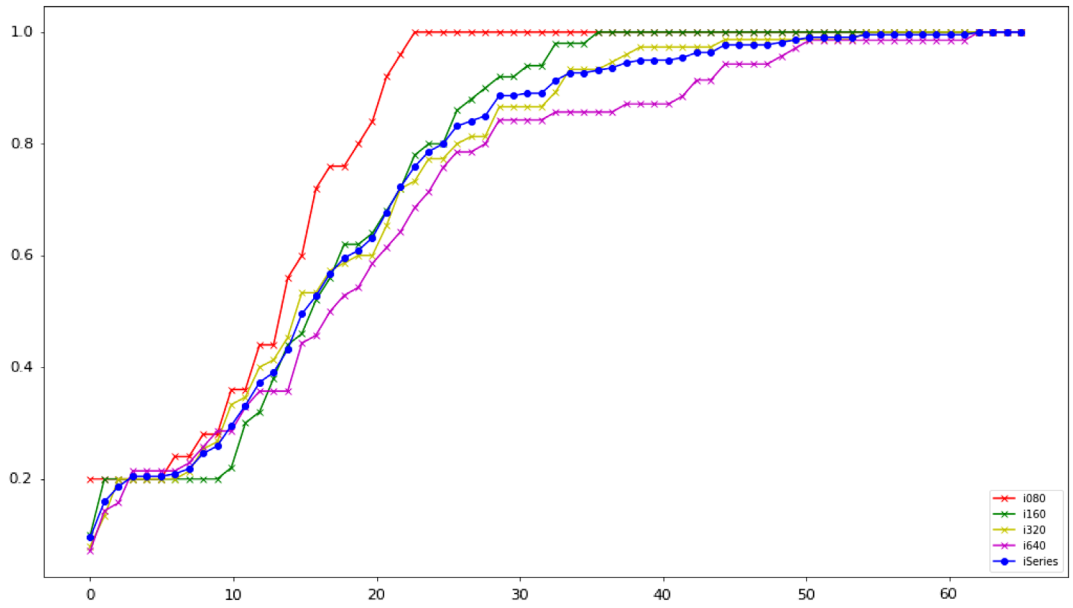

FIGURE 6 Performance profile for studied instances [Color figure can be viewed at wileyonlinelibrary.com]

From Figure 6, we see that the solution quality of the proposed approach is fairly good, considering that we only solve the problem for one laminar family. If we consider all the instances studied, 80% of those instances are within a factor of 1.25 of the optimum value, and all of the instances are within a factor of 1.62 of the optimum value.

This approach could be improved, but we wanted to show the potential of our approach to construct good quality solutions. Note that all of the instances solved with this approach took less than 5 seconds to solve.

5 | CONCLUSIONS AND FUTURE WORK

We propose an LP based approach to the Steiner tree problem. The proposed approach consists of solving a set of independent IPs. The best solution among the set of IPs corresponds to an optimal Steiner tree. Each IP is polynomial in the size of the underlying graph, and we prove that the LP relaxation of each IP is integral, so each IP can be solved as a linear program. The main issue is that the number of IPs to solve grows as $\mathcal{O}\left(\left(\frac{2b}{e}\right)^b\right)$ where $b+1$ is the number of terminal nodes. Consequently, we are able to solve the Steiner tree problem by solving a polynomial number of LPs, when the number of terminals is fixed. This is consistent with previous results [11, 15].

Future research might be directed in trying to develop tools to deal with the large number of laminar families that need to be considered. We are now working to develop an algorithm to solve each subproblem efficiently. The structure of the optimal solution for each problem is well characterized. For each subproblem, the problem reduces to finding the splitting nodes, and then taking the union of shortest paths. Furthermore, we can use this algorithm to construct a local search algorithm. Suppose we have a solution for a given laminar family $l \in \mathcal{L}_b$. There are laminar families in $\mathcal{L}_b$ that have a similar topology to that of l. Then, the idea is to develop an algorithm that searches for an improving laminar family in the neighborhood of l.

Another idea is to use column generation to solve this problem. We can start with a small set of the laminar families, and then try to add new laminar families to the problem in a smart way. The main difficulty with this approach is to develop an efficient pricing problem to determine which laminar families must be added to the problem.

Finally, a different line of research is to study our proposed approach in special graphs. There are important results in the literature for special graphs, in particular there are results that provide an upper bound on the integrality gap for well-known formulations. In these type of graphs, we may find a bound on how bad the optimal solution of the subproblem is, for the "worst" structure compared with the optimal solution for the entire problem. Moreover, there are results that show that for special cases of planar graphs, there is an algorithm to solve the problem in polynomial time [2]. These results may imply that for those type of graphs, the number of tree structures to consider is limited.

ORCID

Matias Siebert https://orcid.org/0000-0002-6570-0492

REFERENCES

[1] E. Balas, *Disjunctive programming: Properties of the convex hull of feasible points*, Discrete Appl. Math. **89** (1998), 3–44.
[2] M. Bern and D. Bienstock, *Polynomially solvable special cases of the Steiner problem in planar networks*, Ann. Oper. Res. **33** (1991), 403–418.
[3] M. Bern and P. Plassmann, *The Steiner problem with edge lengths 1 and 2*, Inform. Process. Lett. **32** (1989), 171–176.
[4] L.J. Billera, S.P. Holmes, and K. Vogtmann, *Geometry of the space of phylogenetic trees*, Adv. Appl. Math. **27** (2001), 733–767.
[5] A. Borchers and D.Z. Du, *The k-Steiner ratio in graphs*, SIAM J. Comput. **26** (1997), 857–869.
[6] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità, *Steiner tree approximation via iterative randomized rounding*, J. ACM **60** (2013), 6.
[7] D. Chakrabarty, J. Könemann, and D. Pritchard, "*Hypergraphic LP relaxations for Steiner trees*," *International Conference on Integer Programming and Combinatorial Optimization*, 2010, pp. 383–396, Springer, Berlin, Heidelberg.
[8] M. Chlebík and J. Chlebíková, *The Steiner tree problem on graphs: Inapproximability results*, Theoret. Comput. Sci. **406** (2008), 207–214.
[9] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*, Springer, Berlin, 2014.
[10] M.P. de Aragão, E. Uchoa, and R.F. Werneck, *Dual heuristics on the exact solution of large Steiner problems*, Electron. Notes Discr. Math. **7** (2001), 150–153.
[11] S. Dreyfus and R.A. Wagner, *The Steiner problem in graphs*, Networks **1** (1971), 195–207.
[12] C. Duin and S. Voß, *Efficient path and vertex exchange in Steiner tree algorithms*, Networks **29** (1997), 89–105.
[13] J. Edmonds, *Optimum branchings*, J. Res. Natl. Bur. Stand. **71** (1967), 233–240.
[14] A.E. Feldmann, J. Könemann, N. Olver, and L. Sanità, *On the equivalence of the bidirected and hypergraphic relaxations for Steiner tree*, Math. Program. **160** (2016), 379–406.
[15] A.E. Feldmann and D. Marx, *The complexity landscape of fixed-parameter directed Steiner network problems*, arXiv preprint arXiv: 1707.06808, 2017.
[16] Z.H. Fu and J.K. Hao, *Swap-vertex based neighborhood for Steiner tree problems*, Math. Program. Comput. **9** (2017), 297–320.
[17] M.X. Goemans, *The Steiner tree polytope and related polyhedra*, Math. Program. **63** (1994), 157–182.
[18] M.X. Goemans and Y.S. Myung, *A catalog of Steiner tree formulations*, Networks **23** (1993), 19–28.
[19] M.X. Goemans, N. Olver, T. Rothvoß, and R. Zenklusen, "*Matroids and integrality gaps for hypergraphic Steiner tree relaxations*," *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, ACM, New York, 2012, pp. 1161–1176.
[20] S. Hougardy, J. Silvanus, and J. Vygen, *Dijkstra meets Steiner: A fast exact goal-oriented Steiner tree algorithm*, Math. Program. Comput. **9** (2017), 135–202.
[21] R.M. Karp, "*Reducibility among combinatorial problems*," *Complexity of Computer Computations*, Springer, Boston, MA, 2018, pp. 85–103.
[22] T. Koch and A. Martin, *Solving Steiner tree problems in graphs to optimality*, Networks **32** (1998), 207–232.
[23] T. Koch, A. Martin, and S. Voß, "*SteinLib: An updated library on Steiner tree problems in graphs*," *Steiner Trees in Industry*, 2001, pp. 285–325, Springer, MA.
[24] J. Könemann, D. Pritchard, and K. Tan, *A partition-based relaxation for Steiner trees*, Math. Program. **127** (2011), 345–370.
[25] R.K. Martin, R.L. Rardin, and B.A. Campbell, *Polyhedral characterization of discrete dynamic programming*, Oper. Res. **38** (1990), 127–138.
[26] T. Polzin and S.V. Daneshmand, *A comparison of Steiner tree relaxations*, Discrete Appl. Math. **112** (2001), 241–261.
[27] T. Polzin and S.V. Daneshmand, *Improved algorithms for the Steiner problem in networks*, Discrete Appl. Math. **112** (2001), 263–300.
[28] T. Polzin and S.V. Daneshmand, "*Extending reduction techniques for the Steiner tree problem*," *European Symposium on Algorithms*, 2002, pp. 795–807, Springer, Berlin, Heidelberg.
[29] T. Polzin and S.V. Daneshmand, *On Steiner trees and minimum spanning trees in hypergraphs*, Oper. Res. Lett. **31** (2003), 12–20.
[30] S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem, In Proceedings of 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 742–751
[31] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, vol. **24**, Springer Science & Business Media, Berlin, 2003.
[32] R.P. Stanley, *Enumerative Combinatorics*, vol. **2**, Cambridge University Press, Cambridge, 1999.
[33] D.M. Warme and J.S. Salowe, *Spanning trees in hypergraphs with applications to Steiner trees*, University of Virginia, Charlottesville, VA, 1998.
[34] D. Watel and M.A. Weisser, *A practical greedy approximation for the directed Steiner tree problem*, J. Comb. Optim. **32** (2016), 1327–1370.
[35] R.T. Wong, *A dual ascent approach for Steiner tree problems on a directed graph*, Math. Program. **28** (1984), 271–287.

How to cite this article: Siebert M, Ahmed S, Nemhauser G. A linear programming based approach to the Steiner tree problem with a fixed number of terminals. *Networks.* 2020;75:124–136. https://doi.org/10.1002/net.21913