On Extended Formulations For Parameterized Steiner Trees

Andreas Emil Feldmann ✉ 📷
Department of Applied Mathematics, Charles University, Prague, Czech Republic

Ashutosh Rai ✉ 📷
Department of Mathematics, IIT Delhi, India

─── **Abstract** ───

We present a novel linear program (LP) for the STEINER TREE problem, where a set of terminal vertices needs to be connected by a minimum weight tree in a graph $G = (V, E)$ with non-negative edge weights. This well-studied problem is NP-hard and therefore does not have a compact extended formulation (describing the convex hull of all Steiner trees) of polynomial size, unless P=NP. On the other hand, STEINER TREE is fixed-parameter tractable (FPT) when parameterized by the number k of terminals, and can be solved in $O(3^k|V| + 2^k|V|^2)$ time via the Dreyfus-Wagner algorithm. A natural question thus is whether the STEINER TREE problem admits an extended formulation of comparable size.

We first answer this in the negative by proving a lower bound on the extension complexity of the STEINER TREE polytope, which, for some constant $c > 0$, implies that no extended formulation of size $f(k)2^{cn}$ exists for any function f. However, we are able to circumvent this lower bound due to the fact that the edge weights are non-negative: we prove that STEINER TREE admits an integral LP with $O(3^k|E|)$ variables and constraints. The size of our LP matches the runtime of the Dreyfus-Wagner algorithm, and our poof gives a polyhedral perspective on this classic algorithm. Our proof is simple, and additionally improves on a previous result by Siebert et al. [2018], who gave an integral LP of size $O((2k/e)^k)|V|^{O(1)}$.

1 Introduction

A central topic in combinatorial optimization is to determine whether a polynomially solvable optimization problem admits a *compact extended formulation*, i.e., if it admits a polytope of polynomial size describing the convex hull of all feasible solutions. The existence of such a polytope means that the corresponding problem can be solved efficiently using linear programming (LP) solvers. This has been a very fruitful research direction from its beginnings all the way to the present day, of which we give a brief overview below (see the surveys [10, 18]) to set the stage for our contribution on the NP-hard STEINER TREE problem.

If $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is an n-dimensional polytope then a d-dimensional polytope $Q \subseteq \mathbb{R}^d$ is an *extension* of P if P is the projection of Q onto the variables of P, i.e., $P = \{x \mid \exists y : (x, y) \in Q\}$. The *size* of a polytope is the number of its facets (i.e., the number

16th International Symposium on Parameterized and Exact Computation (IPEC 2021).
Editors: Petr A. Golovach and Meirav Zehavi; Article No. 18; pp. 18:1–18:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of inequalities in its description), and its *extension complexity* is the minimum size of any of its extensions. An *extended formulation* of an optimization problem is an extension of the polytope given by the convex hull of all characteristic vectors of feasible solutions to the problem. An extended formulation is said to be *compact* if its size is polynomially bounded in the input size. In particular, once a compact extended formulation has been found, we may optimize over it using an LP solver and then project the LP solution to a solution of the problem. Given a modern LP solver this can be used to obtain very efficient algorithms to solve the problem.

The well known Weyl-Minkowski Theorem [28] states that every convex hull of a finite set of vectors (of solution vectors in particular) can be described as a polytope, i.e., every optimization problem with a finite number of solutions admits an extended formulation, which however may not be compact. There are problems for which compact extended formulations exist, for example spanning trees [23] and perfect matchings in planar graphs [3]. On the other hand, there are some polynomially solvable problems that do not admit compact extended formulations, notably for matchings in general graphs: Rothvoß [26] proved that the matching polytope has exponential extension complexity.

For NP-hard problems we do not expect to find compact extended formulations, since this would imply P=NP given that LPs can be solved in polynomial time. However, in the past few decades the development of the theory of fixed-parameter tractability (cf. [11]) has given us a more fine-grained view, where we differentiate between NP-hard problems that are more tractable than others: it can be shown that for some NP-hard problems the expected super-polynomial runtime overhead can be restricted to a *parameter* $k \in \mathbb{N}$, which describes some property of the input, while the runtime remains polynomial in the input size n. Formally, a problem is called *fixed-parameter tractable (FPT)* if it can be solved by an algorithm with runtime $f(k)n^{O(1)}$ for any input of size n and parameter k, where $f : \mathbb{N} \to \mathbb{N}$ is some computable function. Analogous to the complexity class P, which captures all problems that are polynomially solvable, the class FPT contains all problems that are FPT, and can thus be considered more tractable than other NP-hard problems for which FPT algorithms do not exist.

In light of the research on compact extended formulation for problems in P, a very natural question becomes: what problems in FPT admit extended formulations of size $f(k)n^{O(1)}$? Or even more ambitious: suppose that a problem in FPT can be solved in time $O(g(k)n^c)$ for some specific function g and constant c (say $g(k) = 2^k$ and $c = 1$). Is there an extended formulation for which the size matches the running time $O(g(k)n^c)$ of the algorithm? This question has for instance been studied by Buchanan [4] for the VERTEX COVER problem. He gave an extended formulation of size $O(1.47^k + kn)$ where k is the solution size, which does not yet match the currently best running time of $O(1.2738^k + kn)$ for the problem [8]. For the parameterization by the treewidth t of the input graph, an extended formulation of size $O(2^t n)$ exists [5], which in this case also matches the fastest algorithm for this parameter [11]. A generalization of this results is by Kolman et al. [20], who give extended formulations for CSPs parameterized by the treewidth.

Given a graph $G = (V, E)$ and a set of *terminals* $T \subseteq V$, a *Steiner tree* of G is an inclusion-wise minimal connected subgraph of G containing all the terminals (it has to be a tree where all the leaves are terminals). In this paper, we consider the STEINER TREE problem, where we are given an undirected graph G with non-negative edge weights $w : E \to \mathbb{R}^+$ and a set of terminals $T \subseteq V$, and the aim is to find a minimum weight Steiner tree for the terminal set T in G. This problem is known to be NP-hard [14], and has a lot of applications including VLSI routing [9, 17], phylogenetic tree construction [16], and network

routing [22]. A well-studied parameter for this problem is the number of terminals $k = |T|$, for which the classic Dreyfus-Wagner algorithm [12] solves the problem in $O(3^k n + 2^k n^2)$ time. In light of the above discussion, we wish to compare this runtime to the extension complexity of the STEINER TREE polytope, defined as

$$ST(G, T) := \text{conv.hull } \{x^{E(H)} \in \{0, 1\}^{|E(G)|} \mid H \text{ is a Steiner tree for } T \text{ in } G\},$$

where for a subgraph H of G, $x^{E(H)}$ represents the characteristic vector of length $|E(G)|$ for the set $E(H)$, and conv.hull(S) denotes the convex hull of a set S of vectors.

Our first result is a lower bound showing that the extension complexity of the STEINER TREE polytope is exponential, and thus, in contrast to the VERTEX COVER problem, in general we cannot hope to find an extended formulation of $f(k)n^{O(1)}$ size. In fact this is true even if the number k of terminals is constant.

▶ **Theorem 1.** *There exists a constant $c > 0$ such that for any function f, there exists a graph G on n vertices such that the extension complexity of the STEINER TREE polytope $ST(G, T)$ is at least $f(k)2^{cn}$, where $k = |T|$.[1] In particular, the extension complexity of $ST(G, T)$ is $2^{\Omega(n)}$ for some graph G with n vertices even when $|T| = 2$.*

An extended formulation asymptotically matching this exponential lower bound can be obtained for STEINER TREE via the matroid polytope [21]. However, in contrast to Theorem 1 we give an *integral* LP for STEINER TREE, i.e., an LP for which each extreme point is a $(0, 1)$-vector: our main result is that for any graph G and terminal set T, there is an LP of $f(k)n^{O(1)}$ size that optimizes over a polyhedron, which contains an extension of $ST(G, T)$, and every optimum solution to the LP given by *non-negative* edge weights projects to a point of $ST(G, T)$. In other words, we find a polyhedron that describes the lower envelope of the STEINER TREE polytope by projecting to all the characteristic vectors of optimum Steiner trees for non-negative edge weights. Thus, it suffices to solve this LP to solve the STEINER TREE problem given its definition, and so we are able to circumvent the lower bound of Theorem 1.

▶ **Theorem 2.** *For any STEINER TREE instance on a graph $G = (V, E)$ with n vertices, m edges, and k terminals $T \subseteq V$, there is a $3^k m$-dimensional polyhedron Q given by $2^k n + 3^k m$ constraints, such that $ST(G, T) \subseteq \{x \in \mathbb{R}^{|E|} \mid \exists y : (x, y) \in Q\}$ and optimizing over Q with any non-negative edge weight function $w : E \to \mathbb{R}^+$ gives a point of Q that projects to a point of $ST(G, T)$.*

Siebert et al. [27] showed that STEINER TREE can be solved using $O((2k/e)^k)$ polynomial-sized integral LPs. Each solution vector also projects to a Steiner tree. It is possible to write all these LPs into one integral LP (by a result of Balas [1, 2], see Theorem 10), which then has size $(2k/e)^k n^{O(1)} = 2^{O(k \log k)} n^{O(1)}$. Theorem 2 improves on this result by making the size single exponential. Moreover, the size of our LP matches the runtime of the Dreyfus-Wagner algorithm, and thus Theorem 2 provides an alternative way of solving the STEINER TREE problem in comparable runtime via linear programming. In another work, Martin et al. [24] show how to obtain an LP having size linear in the running time of any dynamic programming algorithm. Even though their result can be used to solve the STEINER TREE problem due to the Dreyfus-Wagner dynamic program [12], their LP is not an extension of the STEINER

---

[1] We remark that compared to parameterized runtime lower bounds, this lower bound is unconditional and therefore the function f is not restricted to be computable.

TREE polytope, i.e., it does not project down to $ST(G,T)$. Instead their LP describes the states of the dynamic program and thus gives an indirect way to encode a solution via an LP. On the other hand, our LP solutions describe the vertices of $ST(G,T)$ via a direct projection.

1.1   Our techniques

On a high level our LP of Theorem 2 takes its inspiration from flow based formulations. A well-known approach to obtain an LP relaxation used for approximation algorithms for STEINER TREE is to first construct the corresponding bidirected graph, in which each undirected edge uv is replaced by the two directed edges uv and vu and setting their weights to the weight of the original undirected edge. Then one of the terminals is declared the *root*, and the constraints of the LP relaxation say that each terminal sends a unit flow to the root. Typically, the LP is formulated using cut constraints, which by the celebrated MaxFlow-MinCut theorem imply that the required flows exist. Roughly speaking, for our integral LP, we formulate the problem by sending a "labelled" unit flow from each terminal to the root. The flows are labelled by pairs of terminal subset $T_1, T_2 \subseteq T$ on each edge e, with the meaning that flows from these two sets converge at e. Because we work with these labelled flows, in contrast to usual flow formulations, we do not obtain an equivalent cut formulation. We present our LP in section 2

Since we use directed edges to formulate flows, in fact we prove Theorem 2 for the more general DIRECTED STEINER TREE problem, where we are given a directed graph $G = (V, E)$ (of which bidirectedness is a special case) with non-negative edge weights $w : E \rightarrow \mathbb{R}^+$, a set of *terminals* $T \subseteq V$, and a *root* $r \in V$. The aim is to find an arborescence[2] $A \subseteq G$ of minimum weight, such that there is a path from t to r in A for each $t \in T$.

To prove Theorem 2, in section 3 we develop a primal-dual algorithm for our LP. Interestingly, it turns out that this algorithm can be interpreted as the Dreyfus-Wagner algorithm, and thus we give a polyhedral perspective on this classic algorithm. After proving Theorem 2 we turn to Theorem 1 in section 4. Towards that, we show that the the STEINER TREE polytope contains the HAMILTONIAN PATH polytope as a face, for which exponential lower bounds on the extension complexity can be derived from known lower bounds for HAMILTONIAN CYCLE.

1.2   Related results

Karp [19] mentioned the STEINER TREE problem in his seminal list of NP-hard problems already in 1972, and the problem has since been widely studied. The best approximation algorithm known today is by Byrka et al. [6], and achieves an approximation ratio of $\ln(4) + \epsilon$. Their result uses a hypergraphic LP relaxation, which has also been studied in [7]. Further LPs for STEINER TREE can be found in [15]. Faster FPT algorithms than the one by Dreyfus and Wagner also exist: for arbitrary non-negative edge weights STEINER TREE can be solved in $(2 + \epsilon)^k n^{O_\epsilon(1)}$ time [13], and the unweighted problem can even be solved in $2^k n^{O(1)}$ time [25].

---

[2]   a directed graph with exactly one sink of out-degree 0, for which the underlying undirected graph is a tree.

2   The linear program

In this section we describe our linear program and prove its correctness. Throughout this paper we will assume that the root r of the instance is not contained in the terminal set T. For technical reasons, we also have to assume that each non-terminal (so-called *Steiner vertices*) has at most three neighbours, and the root and each terminal has one neighbour. This can be achieved using a standard preprocessing procedure (cf. Appendix A). We call such a preprocessed instance *reduced*.

For any non-empty terminal subset $T' \subseteq T$ we define the *reachability set* $R_{T'} \subseteq V \cup E$ as those vertices $u \in V$ and edges $uv \in E$ for which every terminal $t \in T'$ has a path to u in G. Our *Parameterized Directed Steiner Tree* (PDST) LP has one variable $x_e^{\{T_1,T_2\}}$ for every unordered pair of sets $T_1, T_2 \subseteq T$ that are disjoint, i.e., $T_1 \cap T_2 = \emptyset$, and not both of T_1 and T_2 are empty, i.e., $T_1 \cup T_2 \neq \emptyset$, and for every directed edge $e \in R_{T_1 \cup T_2}$ reachable from both T_1 and T_2. In the following, $\delta^-(v) \subseteq E$ denotes all incoming edges to a vertex v, while $\delta^+(v) \subseteq E$ denotes its outgoing edges.

$$\min \sum_{\substack{T_1 \cup T_2 \neq \emptyset, \\ T_1 \cap T_2 = \emptyset \\ e \in R_{T_1 \cup T_2}}} x_e^{\{T_1,T_2\}} w(e) \quad \text{such that} \tag{PDST}$$

$$\sum_{\substack{T_1 \cup T_2 = T' \\ T_1 \cap T_2 = \emptyset \\ e \in \delta^-(v) \cap R_{T'}}} x_e^{\{T_1,T_2\}} = \sum_{\substack{T'' \subseteq T \setminus T' \\ e \in \delta^+(v) \cap R_{T' \cup T''}}} x_e^{\{T',T''\}} \qquad \left\{ \begin{array}{l} \forall T' \subseteq T : T' \neq \emptyset, \text{ and} \\ \forall v \in R_{T'} \setminus (T \cup \{r\}) \end{array} \right. \tag{1}$$

$$x_e^{\{\{t\},\emptyset\}} = 1 \qquad \left\{ \begin{array}{l} \forall t \in T, \{e\} = \delta^+(t) \end{array} \right. \tag{2}$$

$$\sum_{\substack{T_1 \cup T_2 = T' \\ T_1 \cap T_2 = \emptyset}} x_e^{\{T_1,T_2\}} = 0 \qquad \left\{ \begin{array}{l} \forall t \in T, \{e\} = \delta^+(t), \text{ and} \\ \forall T' \subseteq T : \emptyset \neq T' \neq \{t\} \end{array} \right. \tag{3}$$

$$\sum_{\substack{T_1 \cup T_2 = T \\ T_1 \cap T_2 = \emptyset}} x_e^{\{T_1,T_2\}} = 1 \qquad \left\{ \begin{array}{l} \{e\} = \delta^-(r) \end{array} \right. \tag{4}$$

$$\sum_{\substack{T_1 \cup T_2 = T' \\ T_1 \cap T_2 = \emptyset}} x_e^{\{T_1,T_2\}} = 0 \qquad \left\{ \begin{array}{l} \{e\} = \delta^-(r), \text{ and} \\ \forall T' \subseteq T : \emptyset \neq T' \neq T \end{array} \right. \tag{5}$$

$$x_e^{\{T_1,T_2\}} \geq 0 \qquad \left\{ \begin{array}{l} \forall T_1, T_2 \subseteq T : T_1 \cap T_2 = \emptyset, \\ T_1 \cup T_2 \neq \emptyset, \text{ and} \\ \forall e \in R_{T_1 \cup T_2} \end{array} \right. \tag{6}$$

Before we prove the correctness of the LP, we try to give some intuition about how it works. As usual, we want an integral solution for the LP to correspond to a solution for the original instance, where exactly one variable corresponding to every edge in the solution is 1. The idea is to look at the solution in the graph as a flow from the terminals to the root. Each terminal pushes a flow of value 1, but as we go closer to the root, the number of edges carrying the flow decrease, while flows from many terminals combine on these edges, but still we would want their corresponding variables to be 1. To this end, we index the variables for the edges with pairs of subsets of T. If there are flows from two subsets of terminals T' and T'' flowing through an edge uv, then we want $x_{uv}^{\{T',T''\}}$ to be set to 1 by a

solution to the LP. From preprocessing, it is guaranteed that in any Steiner arborescence, each vertex has in-degree at most 2. To get the flow for set T' at uv, it must be the case that $T' = T_1 \cup T_2$ for some $T_1, T_2 \subseteq T$ such that $T_1 \cap T_2 = \emptyset$, and T_1 and T_2 had combined to form T' before u. This condition is captured by Constraint (1), which is the crucial constraint of the LP. This process of forming a flow from T' might have happened long before reaching an incoming edge of u and then taking a path to u, so we allow one of T_1 and T_2 to be empty. Constraint (2) ensures that terminals send a flow of 1. Constraint (4) makes sure that there is a flow from all the terminals to the root. Constraint (3) makes sure that a terminal does not send flows for sets other than the singleton set containing itself, and finally Constraint (5) ensures that no other flow from a subset of terminals reaches it, except from the full terminal set. Constraint (5) will not be needed for the correctness of the LP in this section, but will simplify the dual LP given in the next section.

Now we are ready to prove the correctness of the LP, for which we need to argue that there is an integral solution to the LP of cost at most W if and only if there is a Steiner arborescence in the input graph of cost at most W. We begin with the following lemma.

▶ **Lemma 3.** *Let $I := (G, T, r, w)$ be a reduced* DIRECTED STEINER TREE *instance, and let* (PDST) *be the corresponding LP for it. If instance I has a solution of weight at most W, then* (PDST) *has an integral solution of weight at most W.*

Proof. Let A be a minimal solution to I of weight at most W such that every terminal $t \in T$ is a leaf in A and every internal vertex of A has degree at most 3 in A. From minimality of A, we know that A is an arborescence, so for every vertex $v \in V(A) \setminus \{r\}$, we have that $|\delta^+(v) \cap E(A)| = 1$. To get an integral feasible solution for (PDST), we first put $x_e^{\{T_1, T_2\}} := 0$ for all $e \notin E(A) \cap R_{T_1 \cup T_2}$, for all $T_1, T_2 \subseteq T$. We know that every vertex $v \in V(A) \setminus (T \cup \{r\})$, $\deg^+(v) + \deg^-(v) \leq 3$. Now we define a function $T(e)$ for all edges $e = uv \in E(A)$ to be the set of all the terminals in the subtree of A rooted at the vertex u. For every edge $e = uv \in E(A)$ such that $e \neq tv$ for some $t \in T$, u has either one or two incoming edges. If u has only one incoming edge $e' = wu$, then we put $x_e^{\{T(e'), \emptyset\}} := 1$, and $x_e^{\{T_1, T_2\}} := 0$ for all other $T_1, T_2 \subseteq T$. If u has two incoming edges in A, then let $e_1 = w_1 u$ and $e_2 := w_2 u$ be those two edges. We put $x_e^{\{T(e_1), T(e_2)\}} := 1$ and $x_e^{\{T_1, T_2\}} := 0$ for all other $T_1, T_2 \subseteq T$. For $e \in E(A)$ such that $e = tv$ for some $t \in T$, we put $x_e^{\{\{t\}, \emptyset\}} := 1$. For every other variable $x_e^{\{T_1, T_2\}}$, we put $x_e^{\{T_1, T_2\}} := 0$. It is easy to see that this procedure decides to set $x_e^{\{T_1, T_2\}}$ to 1 only if $e \in R_{T_1 \cup T_2}$, and hence the the variable $x_e^{\{T_1, T_2\}}$ exists, and the assignment is valid.

Since for all the edges $e \in E(A)$, exactly one of the variables $x_e^{\{T_1, T_2\}}$ is set to 1 and all other variables are set to 0, it follows that the weight of this assignment for (PDST) is same as weight of A, which is at most W. So all we need to show is that this assignment is feasible for (PDST). It is easy to see that the Constraints (4), (5), and (6) are satisfied by this assignment. To see that Constraint (1) is satisfied, we observe that for any $v \notin V(A)$, all the variables corresponding to edges incident on it are set to 0, and hence (1) is satisfied. For $v \in V(A) \setminus (T \cup \{r\})$, it has exactly one outgoing edge $e = vw$ in A, such that $x_e^{\{T', T''\}} := 1$ for a unique pair $T', T'' \subseteq T$ with $T' \cup T'' \neq \emptyset$ and $T' \cap T'' = \emptyset$. If v has only one incoming edge $e' = uv$ in A, then we have that $T'' = \emptyset$, and T' is the set of terminals in the subtree rooted at u. The assignment puts $x_{e'}^{\{T_1, T_2\}} := 1$ for some $T_1, T_2 \subseteq T'$ such that $T_1 \cap T_2 = \emptyset$ and $T_1 \cup T_2 = T'$, and hence constraint (1) is satisfied. In the other case, let $e_1 = u_1 v$ and $e_2 = u_2 v$ be the two incoming edges to v in A. Let A_1 and A_2 be subtrees of A rooted at u_1 and u_2 respectively and let T_1 and T_2 be set of terminals in A_1 and A_2. The above assignment puts $x_{e_1}^{\{T_3, T_4\}} = 1$ for some T_3 and T_4 such that T_3 and T_4 are disjoint subsets of T_1 with $T_3 \cup T_4 \neq \emptyset$. Since $x_{e_1}^{\{T_3, T_4\}}$ is the only variable corresponding to e_1 put as 1, we can

see that the Constraint (1) is satisfied for v and T'. Similarly we can show that Constraint (1) is satisfied for v and T'' also. To see that Constraints (2) and (3) are satisfied, we only need to look at the edge $tv \in E(A)$ for $t \in T$. For that, the assignment puts $x_{tv}^{\{\{t\},\emptyset\}} = 1$, and all other variables corresponding to the edge tv are set to 0, so Constraints (2) and (3) are also satisfied. ◀

Next we prove the other direction needed to show the correctness of the LP. This step critically relies on the fact that all edge weights are non-negative.

▶ **Lemma 4.** *Let $I := (G, T, r, w)$ be a reduced* DIRECTED STEINER TREE *instance, and let* (PDST) *be the corresponding LP for it. If* (PDST) *has an integral solution of weight at most W then I has a solution of weight at most W.*

Proof. It suffices to prove the lemma for an integral solution x to (PDST) of minimum weight. Consider the support of x in $G = (V, E)$, i.e., the set $F \subseteq E$ of edges for which $x_e^{\{T_1, T_2\}} > 0$ for some $T_1, T_2 \subseteq T$. We claim that for any terminal $t \in T$ there exists a path from t to the root r in the graph spanned by F in G. Since x is integral, any strictly positive variable has value at least 1. Using the fact that all edge weights are non-negative, this implies that the objective function value of (PDST) is at least the weight of the union of these paths (where each edge is counted only once even if it appears in several paths). As these paths form a feasible solution to the DIRECTED STEINER TREE instance, the lemma follows.

To show the existence of a path from some t to r spanned by edges in F, we show that there is a walk from t to r given by a finite sequence of edges $\{e_i\}_{i \geq 1}$, such that for any $i \geq 1$ the head of e_i is the tail of e_{i+1}, and there exists a set $T_1^i \subseteq T$ with $t \in T_1^i$ and $x_{e_i}^{\{T_1^i, T_2^i\}} > 0$ for some $T_2^i \subseteq T$. By Constraint (2), we have $x_e^{\{\{t\},\emptyset\}} = 1$ for $\{e\} = \delta^+(t)$ and so we may set $e_1 = e$, $T_1^1 = \{t\}$, and $T_2^1 = \emptyset$ for $i = 1$. Now let $i \geq 1$ and assume that we have identified an edge $e_i = uv$ with the properties required for the walk. Note that $v \notin T$, since in the reduced instance G any terminal has in-degree 0. So if $v \neq r$ then $v \in V \setminus (T \cup \{r\})$ and Constraint (1) implies that, if $x_{e_i}^{\{T_1^i, T_2^i\}} > 0$ then there must be an edge $e \in \delta^+(v)$ with $x_e^{\{T', T''\}} > 0$ for $T' = T_1^i \cup T_2^i$ and for some $T'' \subseteq T$. Hence we may set $e_{i+1} = e$, $T_1^{i+1} = T'$, and $T_2^{i+1} = T''$.

This way we obtain a walk of potentially infinite length. Note however that if it is finite then it must end in r, since this is the only condition under which we would not be able to identify the next edge e_i for the sequence in the above argument. To see that the length of the walk is finite, note that $T_1^j \subseteq T_1^i$ for all $i > j$, i.e., the sets T_1^i form a sequence with non-decreasing sizes. Assuming the walk has infinite length, there must thus be some index $j \geq 1$ for which $T_1^i = T_1^j$ for all $i > j$, since every set $T_1^i \subseteq T$ has size at most k. By construction we have $T_1^{i+1} = T_1^i \cup T_2^i$, and so $T_1^i = T_1^j = T_1^{i+1}$ implies $T_2^i = \emptyset$ for all $i > j$. Since there is a finite number of edges in G, the edges e_i with $i > j$ must contain a simple cycle C. We now subtract 1 from each variable $x_e^{\{T_1^j, \emptyset\}}$ where $e \in E(C)$, and claim that the resulting solution x' is a feasible integral solution for (PDST), which however would contradict that x has minimum weight. As C is a simple cycle we subtract 1 from both the left- and right-hand side of Constraint (1) for every vertex v lying on C and $T' = T_1^j$. Constraint (6) is also still valid, since every decreased variable lies in the support of the integral solution x. All other constraints are unchanged, leading to the required contradiction. ◀

Combining Theorem 3 and 4 gives us the following theorem, which proves the correctness of the linear program (PDST). In particular, it implies the first part of Theorem 2, i.e., $ST(G, T) \subseteq \{x \mid \exists y : (x, y) \in Q\}$ where Q is the polyhedron defined by the constraints of (PDST).

▶ **Theorem 5.** *Let $I := (G, T, r, w)$ be a reduced DIRECTED STEINER TREE instance, and let* (PDST) *be the corresponding LP for it.* (PDST) *has an integral solution of weight at most W if and only if I has a solution of weight at most W.*

3 Integrality via primal-dual algorithm

In this section we will show that (PDST) is integral. For this we develop a primal-dual algorithm to compute an optimum solution to (PDST), which is also integral. The dual to (PDST) is captured by the following LP, which has a variable $z_v^{T'}$ for every non-empty set $T' \subseteq T$ and vertex $v \in R_{T'}$ reachable from T' corresponding to each of the Constraints (1), (2), (3), (4), and (5). To simplify the notation, we additionally define a variable $z_u^\emptyset$ for every vertex $u \in V \setminus T$ and empty terminal set, and set all these to zero. Note that these variables only occur on the right-hand side of Constraint (7) of the dual LP (PDST*) below.

$$\max z_r^T + \sum_{t \in T} z_t^{\{t\}} \text{ such that} \tag{PDST*}$$

$$z_v^{T_1 \cup T_2} \le z_u^{T_1} + z_u^{T_2} + w(uv) \quad \begin{cases} \forall T_1, T_2 \subseteq T : T_1 \cap T_2 = \emptyset, T_1 \cup T_2 \ne \emptyset \\ \forall uv \in R_{T_1 \cup T_2} : u \notin T \end{cases} \tag{7}$$

$$z_t^{T'} + z_v^{T'} \le w(tv) \quad \begin{cases} \forall T' \subseteq T : T' \ne \emptyset, \text{ and} \\ \forall t \in T, v \in R_{T'}, tv \in E \end{cases} \tag{8}$$

$$z_u^\emptyset = 0 \quad \begin{cases} \forall u \in V \setminus T \end{cases} \tag{9}$$

We now describe the algorithm, which uses (PDST*) to construct an integral solution to (PDST). In particular, the algorithm maintains a feasible dual solution to (PDST*) from which in the end it extracts a feasible solution to (PDST).

Initially the algorithm sets all variables $z_v^{T'} = 0$ in (PDST*). Clearly this dual solution is feasible for (PDST*), as all edge weights are non-negative. The algorithm considers all possible non-empty subsets of terminals $T' \subseteq T$ in non-decreasing order of their sizes (breaking ties arbitrarily) to change variables $z_v^{T'}$ where $v \in R_{T'}$. Starting with singleton terminal sets, if $d(u, v)$ denotes the shortest-path distance from u to v in G, for every $t \in T$ and $v \in R_{\{t\}}$ the algorithm sets

$$z_v^{\{t\}} = d(t, v). \tag{10}$$

After this the algorithm considers each $T' \subseteq T$ with $|T'| \ge 2$. For every $v \in R_{T'}$ it sets the variables according to the following recurrence.

$$z_v^{T'} = \min_{\substack{u \in R_{T'} \\ T_1 \cup T_2 = T' \\ T_1 \cap T_2 = \emptyset \\ T_1, T_2 \ne \emptyset}} \{z_u^{T_1} + z_u^{T_2} + d(u, v)\}. \tag{11}$$

Note that here the sets T_1 and T_2 are proper subsets of T' and have thus been considered by the algorithm in a previous step. Also note that u may be equal to v in which case $d(u, v) = 0$. As a consequence, every variable $z_v^{T'}$ is set to a finite value. In particular, if a vertex $u \in R_{T'}$ does not have a path to v then $d(u, v) = \infty$ and so the minimum of the right-hand side of (11) is always obtained by some vertex u that has a path to v.

We remark that setting the variables as shown above can be seen as a step of the dynamic program of the Dreyfus-Wagner algorithm (where the table entries are given by all dual variables).[3] On the other hand, this can also be seen as setting the dual variables to a highest possible value without violating constraints of (PDST*) (as is typical in primal-dual algorithms), where we witness the maximality of the dual values by arguing that certain corresponding constraints of (PDST*) are *tight*, meaning that they are fulfilled with equality. This is formalized by the following two lemmas, where we first show that the dual solution is feasible.

▶ **Lemma 6.** *Given a non-empty terminal subset $T' \subseteq T$, after the algorithm sets all dual variables $z_v^{T'}$ with $v \in R_{T'}$ according to either (10) or (11), the dual solution is feasible for* (PDST*).

Proof. Neither (10) nor (11) change the value of any variable $z_u^\emptyset$, and as these are set to 0 initially, Constraint (9) is never violated. For any $t \in T$ and $T' \subseteq T$ such that $|T'| \geq 2$, the algorithm never sets $z_t^{T'}$ to a non-zero value because t is a leaf and cannot be reached by all the terminals in T'. Constraint (8) can only be violated for $T' \subseteq T$ of size at least 2 if the algorithm sets $z_v^{T'}$ to a non-zero value according to (11). This is not possible because v is the only neighbour of t and so there does not exist any $u \in R_{T'}$ different from v. So, Constraint (8) can only be violated if the algorithm sets a dual variable $z_v^{\{t\}}$ according to (10), since $|T'| \geq 2$ in (11). However, $z_t^{\{t\}} = d(t,t) = 0$, and so a variable $z_v^{\{t\}}$ is set to its maximum possible value without violating Constraint (8), i.e., it is tight.

For Constraint (7) and an edge uv, note that if $T_1 \neq \emptyset$ and $T_2 \neq \emptyset$ then (11) sets $z_v^{T_1 \cup T_2}$ to a value of at most $z_u^{T_1} + z_u^{T_2} + d(u,v)$, since the shortest path from u to v is considered in the right-hand side of (11). Because $d(u,v) \leq w(uv)$, Constraint (7) is not violated in this case. Furthermore, if one of the two sets, say T_2, is empty, then $z_u^{T_2} = 0$ by Constraint (9) and so Constraint (7) is given by $z_v^{T_1} \leq z_u^{T_1} + w(uv)$. According to (11), $z_u^{T_1}$ is set to some value $z_q^{T_1'} + z_q^{T_2'} + d(q,u)$ for some vertex q and sets T_1', T_2', which together form T_1. The shortest path from q to v is at most as long as the shortest path from q to u plus the edge uv, i.e., $d(q,v) \leq d(q,u) + w(uv)$. Thus according to (11), $z_v^{T_1}$ is set to a value of at most $z_q^{T_1'} + z_q^{T_2'} + d(q,v) \leq z_q^{T_1'} + z_q^{T_2'} + d(q,u) + w(uv) = z_u^{T_1} + w(uv)$, which concludes the proof. ◀

To identify the tight constraints after running the algorithm, we say that Constraint (7) of (PDST*) is *tight for* $(uv, \{T_1, T_2\})$ if the corresponding inequality is tight, i.e., $z_v^{T_1 \cup T_2} = z_u^{T_1} + z_u^{T_2} + w(uv)$, and Constraint (8) of (PDST*) is *tight for* (tv, T') if $z_t^{T'} + z_v^{T'} = w(tv)$ for the corresponding inequality. Given $t \in T$ and $v \in R_{\{t\}}$ so that $z_v^{\{t\}}$ is set according to (10), we say that a $t \to v$ path P in G is *tight for* $\{t\}$ if

- for the first edge tv' of P, Constraint (8) is tight for $(tv', \{t\})$, and
- for every edge $u'v'$ of P, Constraint (7) is tight for $(u'v', \{\{t\}, \emptyset\})$.

Given $|T'| \geq 2$ and $v \in R_{T'}$ so that $z_v^{T'}$ is set according to (11) where the minimum of the right-hand side of (11) is obtained by $u \in R_{T'}$ and $T_1, T_2 \subseteq T'$, we say that a $u \to v$ path P in G is *tight for* $\{T_1, T_2\}$ if

- for the first edge uv' of P, Constraint (7) is tight for $(uv', \{T_1, T_2\})$, and
- for every edge $u'v'$ of P, Constraint (7) is tight for $(u'v', \{T', \emptyset\})$.

---

[3] Note that the dynamic program of Dreyfus and Wagner is typically set up slightly differently, such that the table entries are initially set to ∞. This implies that for any vertex v that is not reachable from a terminal set T', the corresponding entry for v and T' will be set to ∞ when filling the table. Here we instead initialize the dual variables to 0 in order to obtain a feasible dual solution. We then counteract any effects that this has on the recursion by only considering vertices in the reachability set when changing variables.

▶ **Lemma 7.** *Given a terminal $t \in T$, after the algorithm sets all dual variables $z_v^{\{t\}}$ with $v \in R_{\{t\}}$ according to (10), any shortest $t \to v$ path for $v \in R_{\{t\}}$ is tight for $\{t\}$. Given a terminal subset $T' \subseteq T$ with $|T'| \geq 2$, after the algorithm sets all dual variables $z_v^{T'}$ with $v \in R_{T'}$ according to (11) such that $z_v^{T'} = z_u^{T_1} + z_u^{T_2} + d(u, v)$ for some $u \in R_{T'}$ and $T_1, T_2 \subseteq T'$, any shortest $u \to v$ path for $v \in R_{T'}$ is tight for $\{T_1, T_2\}$.*

Proof. Consider a terminal $t \in T$, $v \in R_{\{t\}}$, and any shortest $t \to v$ path P. We saw in the proof of Lemma 6 above that for the first edge tv' of P, Constraint (8) is tight for $(tv', \{t\})$. Now consider any edge $u'v'$ of P. According to (10), $z_{v'}^{\{t\}} = d(t, v')$ and $z_{u'}^{\{t\}} = d(t, u')$. As u' and v' lie on the shortest path P, u' also lies on a shortest $t \to v'$ path and so $d(t, v') = d(t, u') + w(u'v')$. Hence we get $z_{v'}^{\{t\}} = z_{u'}^{\{t\}} + w(u'v')$, i.e., Constraint (7) is tight for $(u'v', \{\{t\}, \emptyset\})$ using that $z_{u'}^{\emptyset} = 0$ according to Constraint (9).

Given a terminal set T' with $|T'| \geq 2$ and $v \in R_{T'}$, let $z_v^{T'}$ be set according to (11) so that $z_v^{T'} = z_u^{T_1} + z_u^{T_2} + d(u, v)$ for some $u \in R_{T'}$ and $T_1, T_2 \subseteq T'$. Consider any vertex p of any shortest $u \to v$ path P. According to (11), $z_p^{T'} = z_q^{T_1'} + z_q^{T_2'} + d(q, p)$ for some $q \in R_{T'}$ and $T_1', T_2' \subseteq T'$, which together form T'. As $z_u^{T_1} + z_u^{T_2} + d(u, p)$ is considered by (11) when setting $z_p^{T'}$, we have $z_q^{T_1'} + z_q^{T_2'} + d(q, p) \leq z_u^{T_1} + z_u^{T_2} + d(u, p)$. Since p lies on the path P leading to v, there exists a path from q to v via p, and $d(q, v) \leq d(q, p) + d(p, v)$. Consequently,

$$
\begin{aligned}
z_q^{T_1'} + z_q^{T_2'} + d(q, v) &\leq z_q^{T_1'} + z_q^{T_2'} + d(q, p) + d(p, v) \\
&\leq z_u^{T_1} + z_u^{T_2} + d(u, p) + d(p, v) \\
&= z_u^{T_1} + z_u^{T_2} + d(u, v),
\end{aligned}
$$

where the last equality uses the fact that p lies on the shortest $u \to v$ path P. The value $z_q^{T_1'} + z_q^{T_2'} + d(q, v)$ is considered when setting $z_v^{T'}$ according to (11). But since $z_u^{T_1} + z_u^{T_2} + d(u, v)$ minimizes the right-hand side of (11) when setting $z_v^{T'}$, the latter inequalities are in fact equalities. In particular, after subtracting $d(p, v)$ from the above, the second (in)equality implies $z_q^{T_1'} + z_q^{T_2'} + d(q, p) = z_u^{T_1} + z_u^{T_2} + d(u, p)$. As $z_p^{T'} = z_q^{T_1'} + z_q^{T_2'} + d(q, p)$ this means that in fact $z_p^{T'} = z_u^{T_1} + z_u^{T_2} + d(u, p)$.

Now consider the first edge uv' of the shortest $u \to v$ path P, for which $w(uv') = d(u, v')$. By setting $p = v'$, from our previous conclusion we obtain $z_{v'}^{T'} = z_u^{T_1} + z_u^{T_2} + w(uv')$, i.e., Constraint (7) is tight for $(uv', \{T_1, T_2\})$. Finally, consider any edge $u'v'$ of P. Since p was an arbitrary vertex of P in the above argument, we can conclude that both $z_{v'}^{T'} = z_u^{T_1} + z_u^{T_2} + d(u, v')$ and $z_{u'}^{T'} = z_u^{T_1} + z_u^{T_2} + d(u, u')$. From $d(u, v') = d(u, u') + w(u'v')$ we thus get $z_{v'}^{T'} = z_{u'}^{T'} + w(u'v')$, i.e., Constraint (7) is tight for $(u'v', \{T', \emptyset\})$. ◀

We now use the tight paths identified by Theorem 7 to extract a feasible integral primal solution to (PDST) from the dual solution computed by the algorithm. This can be done by the following recursive procedure, for which we initially set all variables $x_e^{\{T_1, T_2\}} = 0$. Each recursive call is given by a function $\textsc{primSol}(v, T')$ on some vertex v and terminal set T' with $v \in R_{T'}$, and we begin the recursion with a call $\textsc{primSol}(r, T)$ on the root r and the full terminal set T. Note that w.l.o.g., we may assume that every terminal can reach the root, i.e., $r \in R_T$. We will maintain the invariant that $v \in R_{T'}$ during each recursive call.

Given a vertex v and a terminal set T', the function $\textsc{primSol}(v, T')$ does the following. If $|T'| \geq 2$ then let $P_{T'}$ be any tight shortest $u \to v$ path for $\{T_1, T_2\}$ given by Theorem 7 (using that $v \in R_{T'}$) for the vertex $u \in R_{T'}$ and sets $T_1, T_2 \subseteq T'$ for which the minimum is obtained on the right-hand side of (11) when setting $z_v^{T'}$. If f denotes the first edge of $P_{T'}$, then in (PDST) the function sets $x_f^{\{T_1, T_2\}} = 1$, and $x_e^{\{T', \emptyset\}} = 1$ for every edge $e \neq f$ of $P_{T'}$. After this the function makes a recursive call $\textsc{primSol}(u, T_1)$ on u and T_1, and thereafter a call $\textsc{primSol}(u, T_2)$ on u and T_2. Note that in particular $u \in R_{T_1}$ and $u \in R_{T_2}$ and thus the invariant $v \in R_{T'}$ is given for each recursive call.

Finally, if $T' = \{t\}$ for some terminal $t \in T$, then let $P_{\{t\}}$ be any tight shortest $t \to v$ path for $\{t\}$ given by Lemma x7 (using that $v \in R_{\{t\}}$). The function PRIMSOL$(v, \{t\})$ sets $x_e^{\{\{t\}, \emptyset\}} = 1$ in (PDST) for every edge e of $P_{\{t\}}$. Here the recursion ends.

Note that, since (PDST) only has equality constraints, using tight paths to set variables of the primal solution to 1 means that the primal and dual solutions are *complementary slack*, i.e., for every non-zero variable of the primal (dual) LP the corresponding constraint in the dual (primal) LP is tight. It is well-known (cf. [21]) that feasible primal and dual solutions are complementary slack if and only if both solutions are optimal for their LPs. As we prove in the next lemma, the above procedure computes a feasible integral primal solution. By Theorem 6 also the dual solution is feasible, and so the primal and dual solutions are optimal for (PDST) and (PDST*), respectively. In particular, the primal solution is an optimal integral $(0, 1)$-solution to (PDST), proving the second part of Theorem 2.

▶ **Lemma 8.** *The above recursive procedure computes a feasible integral primal solution to* (PDST) *for any reduced DIRECTED STEINER TREE instance.*

Proof. First observe that for any terminal subset T', (11) always minimizes over proper subsets $T_1, T_2 \subset T'$ partitioning T'. Since the recursive calls are on the same sets T_1 and T_2, this implies that at most one recursive call is made on any set T', and in particular, the path $P_{T'}$ is well-defined. Moreover, for any T' the procedure only sets variables $x_e^{\{T_1, T_2\}}$ to non-zero values if $T_1 \cup T_2 = T'$ (where possibly $T_1 = T'$ and $T_2 = \emptyset$). As a consequence, for every non-zero variable $x_e^{\{T_1, T_2\}}$ with $T_1 \cup T_2 = T'$, e lies on the path $P_{T'}$.

Now consider Constraint (1) of (PDST) for some vertex $v \in V \setminus (T \cup \{r\})$ and non-empty terminal subset $T' \subseteq T$. If any variable $x_e^{\{T_1, T_2\}}$ contributes a non-zero value to the left-hand side of this constraint, then e lies on $P_{T'}$, since $T_1 \cup T_2 = T'$ by definition of Constraint (1). Note that there can be only one non-zero variable on the left-hand side of the constraint, as there is only one recursive call on T'. Furthermore, v can only be the last vertex or an internal vertex of $P_{T'}$, since $e \in \delta^-(v)$ for any left-hand side variable.

If v is the last vertex of the tight path $P_{T'}$ for $\{T_1, T_2\}$, then the procedure did a recursive call PRIMSOL(v, T') on T' and v. Since $v \neq r$, this call was made due to a previous call on a set $T' \cup T''$ for some non-empty set T'', and v is the first vertex of the tight path $P_{T' \cup T''}$ for $\{T', T''\}$. During this call the function sets the variable $x_f^{\{T', T''\}} = 1$ for the the first edge $f \in \delta^+(v)$ of $P_{T' \cup T''}$. This variable thus contributes a non-zero value to the right-hand side of the constraint for v and T'. As v is the last vertex of $P_{T'}$ and only the unique recursive call on T' sets variables $x_e^{\{T', \emptyset\}}$ to non-zero values, all such variables with $e \in \delta^+(v)$ on the right-hand side of the constraint are set to zero. Furthermore, for each non-zero variable $x_e^{\{T', T''\}}$ with $T'' \neq \emptyset$ and $e \in \delta^+(v)$, there is a recursive call PRIMSOL(v, T') on v and T': the procedure will only set such a variable to a non-zero value if both T' and T'' are non-empty and e is the first edge of the tight path for $\{T', T''\}$. As each recursive call is on a unique set T', this means that there can only be one non-zero variable on the right-hand side of the constraint. Since all non-zero variables are set to 1, if the left-hand side has one variable set to 1, then so does the right-hand side. Note that the above arguments also imply the reverse: if the right-hand side has a variable $x_f^{\{T', T''\}} = 1$ with $f \in \delta^+(v)$, then the recursive call PRIMSOL(v, T') sets a variable $x_e^{\{T_1, T_2\}} = 1$ with $e \in \delta^-(v)$, and there can be only one such non-zero variable on each of the right- and left-hand sides. Therefore the constraint is valid if v is the last vertex of $P_{T'}$.

If v is an internal vertex of $P_{T'}$, the procedure sets $x_e^{\{T', \emptyset\}}$ for the edge $e \in \delta^+(v)$ lying on $P_{T'}$ to a non-zero value. As only the unique call on T' sets variables $x_e^{\{T', \emptyset\}}$, no other such variable with $e \in \delta^+(v)$ is set to a non-zero value. Now consider a variable $x_f^{\{T', T''\}}$

with $f \in \delta^+(v)$ and $T'' \neq \emptyset$. Since both T' and T'' are non-empty, if $x_f^{\{T',T''\}}$ is non-zero then f must be the first edge of a tight path $P_{T' \cup T''}$ and $x_f^{\{T',T''\}}$ is set during a recursive call on $T' \cup T''$. However, since $|T' \cup T''| \geq 2$, this would mean during this call a recursive call PRIMSOL(v, T') was made on v and T', implying that v is the last vertex of $P_{T'}$ – a contradiction. Hence as before, if the left-hand side of the constraint has one variable set to 1, then so does the right-hand side, and the reverse is also true. Thus Constraint (1) is valid.

For the other constraints of (PDST) it is easy to see that they are valid: for Constraint (2) the procedure sets $x_e^{\{\{t\},\emptyset\}} = 1$ for $\{e\} \in \delta^+(t)$, as the recursion begins with the full terminal set T and so must end in each terminal $t \in T$, and the tight path $P_{\{t\}}$ for $\{t\}$ always has t as the first vertex. Since all variables are initially set to zero, for Constraint (3) it suffices to note that no tight path $P_{T'}$ with $T' \neq \{t\}$ can contain terminal t, as the DIRECTED STEINER TREE instance is reduced and so every terminal has only one neighbour. Constraint (4) is valid since the recursion begins at the root r with the full terminal set T, so that some variable $x_e^{\{T_1,T_2\}}$ with $T_1 \cup T_2 = T$ is set to 1. As the recursive call on T is unique, exactly one such variable is non-zero. For Constraint (5) it again suffices to note that no tight path $P_{T'}$ with $T' \neq T$ can contain r, since also the root only has one neighbour. Finally, Constraint (6) is obviously valid, since the procedure sets all variables to either 0 or 1. ◄

4   Extension complexity of the Steiner tree polytope

In this section we prove Theorem 1, i.e., that the extension complexity of the Steiner tree polytope $ST(G,T)$ is exponential. For that, we will need the following two additional polytopes. One is for the HAMILTONIAN PATH problem, where we are given an undirected graph $G = (V,E)$ and two vertices $u, v \in V$ and we need to decide whether there exists a path P in G between u and v such that P visits all vertices of G. A related problem is the HAMILTONIAN CYCLE problem, where we need to decide whether an undirected graph G has a cycle that visits all the vertices of G.

$$HP(G,u,v) := \text{conv.hull } \{x^{E(P)} \in \{0,1\}^{|E(G)|} \mid P \text{ is a Hamiltonian } u \to v \text{ path in } G\}$$
$$HC(G) := \text{conv.hull } \{x^{E(C)} \in \{0,1\}^{|E(G)|} \mid C \text{ is a Hamiltonian cycle in } G\}$$

Now, we first show a lemma that relates the extension complexity of $HP(G,u,v)$ and $ST(G,T)$ for any graph G. In fact, we will show the lemma for the case when the number of terminals for the STEINER TREE instance is 2. In the following we denote the extension complexity of any polytope Q by $xc(Q)$.

▶ **Lemma 9.** *For any graph G, $xc(HP(G,u,v)) \leq xc(ST(G,\{u,v\}))$.*

Proof. Observe that $\sum_{e \in E(G)} x_e \leq n-1$ is a valid inequality for $ST(G,T)$, as any Steiner tree has at most $n-1$ edges, and thus $ST(G,T) \cap \{x \mid \sum_{e \in E} x_e = n-1\}$ is a face of $ST(G,T)$. We want to show that $HP(G,u,v) = ST(G,\{u,v\}) \cap \{x \mid \sum_{e \in E} x_e = n-1\}$. This would mean that the polytope $HP(G,u,v)$ is a face of the polytope $ST(G,\{u,v\})$, which would imply that the extension complexity of $HP(G,u,v)$ is at most the extension complexity of $ST(G,\{u,v\})$, and thus prove the lemma.

Now, to show the claim, it is enough to show that the extreme points of the left-hand side belong to the polytope on the right-hand side and vice versa. We look at an extreme point $x \in HP(G,u,v)$, that is, there exists a Hamiltonian path P between u and v such that $x = x^{E(P)}$. Clearly, P has $n-1$ edges, so we have that $\sum_{e \in E} x_e = n-1$. On the other hand, since P is a path between u and v, it is a minimal connected subgraph containing u and v and hence $x \in ST(G,\{u,v\})$.

For the other direction, let us take an extreme point of the polytope $ST(G, \{u, v\}) \cap \{x \mid \sum_{e \in E} x_e = n - 1\}$. As observed earlier, for all the extreme points x of $ST(G, \{u, v\})$, $\sum_{e \in E} x_e \leq n - 1$ is a valid inequality. This means that the polytope $ST(G, \{u, v\})$ lies on one side of the hyperplane $\{x \mid \sum_{e \in E} x_e = n - 1\}$. This gives us that all the extreme points of $ST(G, \{u, v\}) \cap \{x \mid \sum_{e \in E} x_e = n - 1\}$ are also extreme points of $ST(G, \{u, v\})$. So, to prove the other direction, it is enough to look at the extreme points of $ST(G, \{u, v\})$. Now, let x be an extreme point of $ST(G, \{u, v\})$, which also belongs to $\{x \mid \sum_{e \in E} x_e = n - 1\}$. This means that $x = x^{E(H)}$ for some Steiner tree H for $(G, \{u, v\})$. We also know that $\sum_{e \in E} x_e = n - 1$. Since H is a minimal subgraph connecting u and v having $n - 1$ edges, it has to be a Hamiltonian path between u and v, and hence $x \in HP(G, u, v)$. This finishes the proof of the lemma. ◄

Now we state the following lemma by Balas [1, 2], which bounds the extension complexity of union of several polytopes.

▶ **Theorem 10** ([1, 2]). *Consider q polytopes $P^i \subseteq \mathbb{R}^n$, $i = 1, \ldots, q$ and write $P := \text{conv.hull}(\cup_{i \in [q]} P^i)$. Then, $xc(P) \leq q + \sum_{i \in [q]} xc(P^i)$.*

We use this theorem in the following to relate the extension complexities of Hamiltonian cycles of the complete graph K_n and Hamiltonian paths.

▶ **Lemma 11.** *Let K_n be a complete graph with n vertices and u and v be two arbitrary vertices of K_n. Then $xc(HC(K_n)) \leq \binom{n}{2} + \binom{n}{2} \cdot xc(HP(K_n - uv, u, v))$, where $K_n - uv$ is the graph K_n with the edge uv removed.*

Proof. We will show $HC(K_n)$ to be convex hull of the union of $\binom{n}{2}$ polytopes, which have their extension complexity same as that of $HP(K_n - uv, u, v)$. We take $Q_{uv} := HP(K_n - uv, u, v)$ for all $uv \in E(K_n)$. Observe that the dimension of Q_{uv} is $\binom{n}{2} - 1$. For each $x \in Q_{uv}$, we make a $y \in P_{uv}$ of length $\binom{n}{2}$ by adding the co-ordinate for the edge uv and giving it value 1. Since P_{uv} is just an embedding of the $(\binom{n}{2} - 1)$-dimensional polytope Q_{uv} into $\binom{n}{2}$-dimensional space by fixing the additional co-ordinate to be 1, the number of facets of P_{uv} and Q_{uv} remain the same, which is at most $xc(HP(K_n - uv, u, v))$. Observe that for all edges uv and $u'v'$ in $E(K_n)$, the polytopes $HP(K_n - uv, u, v)$ and $HP(K_n - u'v', u', v')$ are isomorphic to each other and hence have the same extension complexity. So the polytopes P_{uv} and $P_{u'v'}$ also have the same extension complexity for all $uv, u'v' \in E(K')$.

Now we just need to show that $HC(K_n) = \text{conv.hull}(\cup_{uv \in E(K_n)} P_{uv})$, and the proof of the lemma will follow by Theorem 10. Let us look at an extreme point $x \in HC(K_n)$. Clearly, $x = x^{E(C)}$ for some Hamiltonian cycle C of K_n. Let $e = uv$ be an arbitrary edge in C. We claim that $x \in P_{uv}$. For that, first observe that C contains a Hamiltonian path between u and v, which is the cycle C with edge uv removed. So the characteristic vector of C is the same as the characteristic vector of $C - uv$ in $K_n - uv$ along with 1 at the co-ordinate for the edge uv, so $x \in P_{uv}$. For the other side, let x be an extreme point in $\cup_{uv \in E(K_n)} P_{uv}$ and hence in some P_{uv}. From the construction of P_{uv}, x has the coordinates for $E(P)$ and the edge uv as 1, where P is some Hamiltonian path between u and v in $K_n - uv$. This shows that x is a characteristic vector of a Hamiltonian cycle in K_n and hence $x \in HC(K_n)$. ◄

To show Theorem 1 with the help of the lemmas proved above, we use the following result about the extension complexity of Hamiltonian cycles in complete graphs.

▶ **Theorem 12** ([26]). $xc(HC(K_n)) = 2^{\Omega(n)}$.

Proof of Theorem 1. For the sake of contradiction, let us assume that Theorem 1 is not true. This means that for all constants $c > 0$, there exists a function f such that for all n vertex graphs G, $xc(ST(G, T)) < f(k)2^{cn}$, where $k = |T|$. Now, Lemma 9 and Lemma 11 imply that $xc(HC(K_n)) \leq \binom{n}{2} + \binom{n}{2}xc(HP(K_n - uv, u, v)) \leq \binom{n}{2} + \binom{n}{2}xc(ST(K_n - uv, \{u, v\}))$. We know that $xc(ST(K_n - uv, \{u, v\})) < f(2)2^{cn}$ for some function f and all constants c. This would mean that $xc(HC(K_n)) < d2^{cn}$ for all constants c, where $d = f(2)$ is another constant, which contradicts Theorem 12. This finishes the proof of Theorem 1. ◀

References

1 Egon Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486, 1985.

2 Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, 1998.

3 Francisco Barahona. On cuts and matchings in planar graphs. *Mathematical Programming*, 60(1):53–68, June 1993. `doi:10.1007/BF01580600`.

4 Austin Buchanan. Extended formulations for vertex cover. *Operations Research Letters*, 44(3):374–378, 2016.

5 Austin Loyd Buchanan. *Parameterized Approaches for Large-Scale Optimization Problems*. PhD thesis, 2015.

6 J. Byrka, F. Grandoni, T. Rothvoß, and Laura Sanità. An improved LP-based approximation for Steiner tree. In *Proc. 42nd STOC*, pages 583–592, 2010. `doi:10.1145/1806689.1806769`.

7 Deeparnab Chakrabarty, Jochen Könemann, and David Pritchard. Hypergraphic lp relaxations for steiner trees. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 383–396. Springer, 2010.

8 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010.

9 Jun-Dong Cho. *Steiner Tree Problems in VLSI Layout Designs*, pages 101–173. Springer US, Boston, MA, 2001.

10 Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

12 S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

13 Bernhard Fuchs, Walter Kern, D Molle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007.

14 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

15 Michel X Goemans and Young-Soo Myung. A catalog of steiner tree formulations. *Networks*, 23(1):19–28, 1993.

16 F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem*. ISSN. Elsevier Science, 1992.

17 A.B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. The Springer International in Engineering and Computer Science. Springer US, 2013.

18 V Kaibel. Extended formulations in combinatorial optimization. *Optima 85*, page 14. URL: `http://www.mathopt.org/Optima-Issues/optima85.pdf`.

19 R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

20   Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. Extension complexity, mso logic, and treewidth. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.

21   Bernhard Korte and Jens Vygen. *Combinatorial optimization*, volume 2. Springer.

22   B.H. Korte. *Paths, flows, and VLSI-layout*. Algorithms and combinatorics. Springer-Verlag, 1990.

23   R. Kipp Martin. Using separation algorithms to generate mixed integer model reformulations. *Oper. Res. Lett.*, 10(3):119–128, 1991. `doi:10.1016/0167-6377(91)90028-N`.

24   R. Kipp Martin, Ronald L. Rardin, and Brian A. Campbell. Polyhedral characterization of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990. URL: `http://www.jstor.org/stable/171304`.

25   Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013.

26   Thomas Rothvoss. The matching polytope has exponential extension complexity. *J. ACM*, 64(6):41:1–41:19, 2017.

27   Matias Siebert, Shabbir Ahmed, and George Nemhauser. A linear programming based approach to the steiner tree problem with a fixed number of terminals. *Networks*, n/a(n/a).

28   François Vanderbeck and Laurence A. Wolsey. *Reformulation and Decomposition of Integer Programs*, pages 431–502. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

A   Preprocessing

In this section, given an instance (G, T, r, w) of Directed Steiner Tree, we will transform it into an equivalent instance (G', T', r', w'), where G' is a directed graph and G has a solution of weight W if and only if there exists a solution for G' of weight W where all the terminals are leaves with out-degree 1, the root is a leaf with in-degree 1 and for every other vertex v, we have that $\deg^+(v) + \deg^-(v) \leq 3$. Formally, we want to prove the following lemma.

▶ **Lemma 13.** *Given an instance (G, T, r, w) of Directed Steiner Tree, in polynomial time, we can transform it into another Directed Steiner Tree instance (G', T', r', w') such that (G, T, r, w) has a solution A of weight W if and only if (G', T', r', w') has a solution A' of weight W where all the terminals are leaves with out-degree 1, the root is a leaf with in-degree 1, and for all $v \in V(G') \setminus (T' \cup \{r'\})$, we have $\deg^+(v) + \deg^-(v) \leq 3$.*

Proof. We will prove the lemma in two stages. First we transform (G, T, r, w) to (G_1, T_1, r_1, w_1) where G_1 is the graph obtained by making a copy t' of every terminal $t \in T$, and adding the edge (t', t) with zero weight. Let T^* be the set of these new vertices. We also add a vertex r^* as a new root, and add the edge (r, r^*) with zero weight. Then we take $T_1 := T^*$, $r_1 := r^*$, and w_1 is obtained by adding the new zero edge weights to w. It is easy to see that (G, T, r, w) has a solution of weight W if and only if (G_1, T_1, r_1, w_1) has a solution of weight W where all the terminals are leaves with out-degree 1 and the root is a leaf with in-degree 1.

   Given (G_1, T_1, r_1, w_1) of Directed Steiner Tree as described above, we make an instance (G_2, T_2, r_2, w_2) of Directed Steiner Tree by the following transformation. For every vertex $v \in V(G_1) \setminus (T_1 \cup \{r_1\})$ with in-degree $\deg^-(v)$ and out-degree $\deg^+(v)$ such that $\deg^+(v) + \deg^-(v) \geq 4$, we replace it with a cycle of length $d := \deg^+(v) + \deg^-(v)$ defined as $C_v = v_1 v_2 \ldots v_d v_1$ where $v_1, v_2, \ldots, v_d$ are new vertices. Then for every neighbour u of v, if $uv \in E(G_1)$ (or $vu \in E(G_1)$), we find a unique v_i of v, a unique copy u_j of u, and add the edge $v_i u_j$ (or $u_j v_i$). Then we take $T_2 := T_1$ and $r_2 := r_1$ This way, the graph G_2 has maximum degree 3 and all the terminals are leaves in G_2 with out-degree 1 and the root is a

leaf with in-degree 1. For every edge e in the cycle C_v, we put $w_2(e) := 0$. For every other edge $u_i v_j \in E(G_2)$ such that u_i and v_j are in cycles C_u and C_v corresponding to $u \in V(G_1)$ and $v \in V(G_1)$, we put $w_2(u_i v_j) := w_1(uv)$. Given a solution of weight W of (G_2, T_2, r_2, w_2), we can transform it into a solution of weight W for (G_1, T_1, r_1, w_1) by contracting the zero weight edges of the cycles in the solution. On the other hand, given a solution of weight W for (G_1, T_1, r_1, w_1), for an edge uv in the solution, we pick the edge between unique copies u_i and v_j of u and v corresponding to the edge uv and connect all these edges by picking as many edges from C_u and C_v as we want at no extra cost. Observe that the terminals and the root are leaves in G_2, so they will be leaves in any solution as well. This shows that (G_1, T_1, r_1, w_1) has a solution of weight W where all the terminals are leaves with out-degree 1 and the root is also a leaf with in-degree 1 if and only if (G_2, T_2, r_2, w_2) has a solution of weight W where all the terminals are leaves with out-degree 1 and the root is also a leaf with in-degree 1 and all the vertices $v \in V(G_2) \setminus (T_2 \cup \{r_2\})$, we have that $\deg^+(v) + \deg^-(v) \leq 3$. Finally taking $(G', T', r', w') := (G_2, T_2, r_2, w_2)$ finishes the proof of the lemma. ◀