A Logarithmic Integrality Gap Bound for Directed Steiner Tree in Quasi-bipartite Graphs*

Zachary Friggstad[1], Jochen Könemann[2], and Mohammad Shadravan[3]

1 **Department of Computing Science, University of Alberta, Edmonton, Canada**
zacharyf@ualberta.ca
2 **Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada**
jochen@uwaterloo.ca
3 **Department of Industrial Engineering and Operations Research, Columbia University, New York, USA**
ms4961@columbia.edu

Abstract

We demonstrate that the integrality gap of the natural cut-based LP relaxation for the directed Steiner tree problem is $O(\log k)$ in quasi-bipartite graphs with k terminals. Such instances can be seen to generalize set cover, so the integrality gap analysis is tight up to a constant factor. A novel aspect of our approach is that we use the primal-dual method; a technique that is rarely used in designing approximation algorithms for network design problems in directed graphs.

1998 ACM Subject Classification G.1.6 Optimization, G.2.2 Graph Theory, I.1.2 Artificial Intelligence

Keywords and phrases Approximation algorithm, Primal-Dual algorithm, Directed Steiner tree

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.3

1 Introduction

In an instance of the *directed Steiner tree* (DST) problem, we are given a directed graph $G = (V, E)$, non-negative costs c_e for all $e \in E$, *terminal* nodes $X \subseteq V$, and a root $r \in V$. The remaining nodes in $V - (X \cup \{r\})$ are the *Steiner nodes*. The goal is to find the cheapest collection of edges $F \subseteq E$ such that for every terminal $t \in X$ there is an r, t-path using only edges in F. Throughout, we let n denote $|V|$ and k denote $|X|$.

If $X \cup \{r\} = V$, then the problem is simply the *minimum-cost arborescence* problem which can be solved efficiently [5]. However, the general case is well-known to be NP-hard. In fact, the problem can be seen to generalize the *set-cover* and *group Steiner tree* problems. The latter cannot be approximated within $O(\log^{2-\epsilon}(n))$ for any constant $\epsilon > 0$ unless NP $\subseteq$ DTIME$(n^{\text{polylog}(n)})$ [11].

For a DST instance G, let OPT_G denote the value of the optimum solution for this instance Say that an instance $G = (V, E)$ of DST with terminals X is ℓ-*layered* if V can be partitioned as $V_0, V_1, \ldots, V_\ell$ where $V_0 = \{r\}, V_\ell = X$ and every edge $uv \in E$ has $u \in V_i$

* This work was in part completed while the first author was a postdoctoral fellow and the third author was a graduate student at the University of Waterloo. The work of all three authors is supported by NSERC's Discovery grant program. The second author greatfully acknowledges the support of the Hausdorff Institute and the Institute for Discrete Mathematics in Bonn, Germany.

and $v \in V_{i+1}$ for some $0 \leq i < \ell$. Zelikovsky showed for any DST instance G and integer $\ell \geq 1$ that we can compute an ℓ-layered DST instance H in poly(n, ℓ) time such that $OPT_G \leq OPT_H \leq \ell \cdot k^{1/\ell} \cdot OPT_G$ and that a DST solution in H can be efficiently mapped to a DST solution in G with the same cost [2, 17].

Charikar et al. [3] exploited this fact and presented an $O(\ell^2 \cdot k^{1/\ell} \cdot \log k)$-approximation with running time poly(n, k^ℓ) for any integer $\ell \geq 1$. In particular, this can be used to obtain an $O(\log^3 k)$-approximation in quasi-polynomial time and a polynomial-time $O(k^\epsilon)$-approximation for any constant $\epsilon > 0$. Finding a polynomial-time polylogarithmic approximation remains an important open problem.

For a set of nodes S, we let $\delta^{in}(S) = \{uv \in V : u \notin S \text{ and } v \in S\}$ be the set of edges entering S. The following is a natural linear programming (LP) relaxation for directed Steiner tree.

$$\min \quad \sum_{e \in E} c_e x_e \qquad \textbf{(DST-Primal)}$$

$$\text{s.t.} \quad x(\delta^{\text{in}}(S)) \geq 1 \quad \forall\, S \subseteq V - r, S \cap X \neq \emptyset \qquad (1)$$

$$x_e \geq 0 \quad \forall e \in E$$

This LP is called a *relaxation* because of the natural correspondence between feasible solutions to a DST instance G and feasible $\{0, 1\}$-integer solutions to the corresponding LP (**DST-Primal**). Thus, if we let OPT_{LP} denote the value of an optimum (possibly fractional) solution to LP (**DST-Primal**) then we have $OPT_{LP} \leq OPT_G$. For a particular instance G we say the *integrality gap* is OPT_G / OPT_{LP}; we are interested in placing the smallest possible upper bound on this quantity.

Interestingly, if $|X| = 1$ (the shortest path problem) or $X \cup \{r\} = V$ (the minimum-cost arborescence problem), the extreme points of (**DST-Primal**) are integral so the integrality gap is 1 ([13] and [5], respectively). However, in the general case Zosin and Khuller showed that (**DST-Primal**) is not useful for finding polylog(k)-approximation algorithms for DST [18]. The authors showed that the integrality gap of (**DST-Primal**) relaxation can, unfortunately, be as bad as $\Omega(\sqrt{k})$, even in instances where G is a 4-layered graph. In their examples, the number of nodes n is exponential in k so the integrality gap may still be $O(\log^c n)$ for some constant c.

On the other hand, Rothvoss recently showed that applying $O(l)$ rounds of the semidefinite programming Lasserre hierarchy to (the flow-based extended formulation of) (**DST-Primal**) yields an SDP with integrality gap $O(\ell \cdot \log k)$ for ℓ-layered instances [15]. Subsequently, Friggstad et al. [7] showed similar results for the weaker Sherali-Adams and Lovász-Schrijver linear programming hierarchies.

In this paper we consider the class of *quasi-bipartite* DST instances. An instance of DST is quasi-bipartite if the Steiner nodes $V \setminus (X \cup \{r\})$ form an independent set (i.e., no directed edge has both endpoints in $V \setminus (X \cup \{r\})$). Such instances still capture the set cover problem, and thus do not admit an $(1 - \epsilon) \ln k$-approximation for any constant $\epsilon > 0$ unless P = NP [4, 6]. Furthermore, it is straightforward to adapt known integrality gap constructions for set cover (e.g. [16]) to show that the integrality gap of (**DST-Primal**) can be as bad as $(1 - o(1)) \cdot \ln k$ in some instances. Hibi and Fujito [12] give an $O(\log k)$-approximation for quasi-bipartite instances of DST, but do not provide any integrality gap bounds.

Quasi-bipartite instances have been well-studied in the context of *undirected* Steiner trees. The class of graphs was first introduced by Rajagopalan and Vazirani [14] who studied the integrality gap of (**DST-Primal**) for the *bidirected* map of the given undirected Steiner

tree instances. Currently, the best approximation for quasi-bipartite instances of undirected Steiner tree is $\frac{73}{60}$ by Goemans et al. [8] who also bound the integrality gap of the bidirected cut relaxation by the same quantity. This is the same LP relaxation as (**DST-Primal**), applied to the directed graph obtained by replacing each undirected edge $\{u, v\}$ with the two directed edges uv and vu. This is a slight improvement over a prior $(\frac{73}{60} + \epsilon)$-approximation for any constant $\epsilon > 0$ by Byrka et al. [1].

The best approximation for general instances of undirected Steiner tree is $\ln(4) + \epsilon$ for any constant $\epsilon > 0$ [1]. However, the best known upper bound on the integrality gap of the bidirected cut relaxation for non-quasi-bipartite instances is only 2; it is an open problem to determine if this integrality gap is a constant-factor better than 2.

1.1 Our contributions

Our main result is the following. Let $H_n = \sum_{i=1}^{n} 1/i = O(\log n)$ be the nth harmonic number.

▶ **Theorem 1.** *The integrality gap of LP (**DST-Primal**) is at most $2\,H_k = O(\log k)$ in quasi-bipartite graphs with k terminals. Furthermore, a Steiner tree with cost at most $2\,H_k \cdot OPT_{LP}$ can be constructed in polynomial time.*

As noted above, Theorem 1 is asymptotically tight since any of the well-known $\Omega(\log k)$ integrality gap constructions for set cover instances with k items translate directly to an integrality gap lower bound for (**DST-Primal**), using the usual reduction from set cover to 2-layered quasi-bipartite instances of directed Steiner tree.

This integrality gap bound asymptotically matches the approximation guarantee proven by Hibi and Fujito for quasi-bipartite DST instaces [12]. We remark that their approach is unlikely to give any integrality gap bounds for (**DST-Primal**) because they iteratively choose *low-density full Steiner trees* in the same spirit as [3] and give an $O(\ell \cdot \log k)$-approximation for finding the optimum DST solution T that does not contain a path with $\geq \ell$ Steiner nodes $V \setminus (X \cup \{r\})$. In particular, their approach will also find an $O(\log k)$-approximation to the optimum DST solution in 4-layered graphs and we know the integrality gap in some 4-layered instances is $\Omega(\sqrt{k})$ [18].

We prove Theorem 1 by constructing a directed Steiner tree in an iterative manner. An iteration starts with a *partial* Steiner tree (see Definition 2 below), which consists of multiple directed components containing the terminals in X. Then a set of arcs are purchased to *augment* this partial solution to one with fewer directed components. These arcs are discovered through a primal-dual moat growing procedure; a feasible solution for the dual (**DST-Primal**) is constructed and the cost of the purchased arcs can be bounded using this dual solution.

While the primal-dual technique has been very successful for *undirected* network design problems (e.g., see [9]), far fewer success stories are known in *directed* domains. Examples include a primal-dual interpretation of Dijkstra's shortest path algorithm (e.g., see Chapter 5.4 of [13]), and Edmonds' [5] algorithm for minimum-cost arborescences. In both cases, the special structure of the problem is instrumental in the primal-dual construction. One issue arising in the implementation of primal-dual approaches for directed network design problems appears to be a certain *overlap* in the *moat* structure maintained by these algorithms. We are able to handle this difficulty here by exploiting the quasi-bipartite nature of our instances.

2 The integrality gap bound

2.1 Preliminaries and definitions

We now present an algorithmic proof of Theorem 1. As we will follow a primal-dual strategy, we first present the LP dual of (**DST-Primal**).

$$\max \quad \sum_{S} y_S \qquad \text{(DST-Dual)}$$

$$\text{s.t.} \quad \sum_{S: e \in \delta^{\text{in}}(S)} y_S \leq c_e \ \forall e \in E \qquad (2)$$

$$y \geq 0$$

In (**DST-Dual**), the sums range only over sets of nodes S such that $S \subseteq V - r$ and $S \cap X \neq \emptyset$.

Our algorithm builds up partial solutions, which are defined as follows.

▶ **Definition 2.** A *partial Steiner tree* is a tuple $\mathcal{T} = (\{B_i, h_i, F_i\}_{i=0}^{\ell}, \bar{B})$ where, for each $0 \leq i \leq \ell$, B_i is a subset of nodes, $h_i \in B_i$, and F_i is a subset of edges with endpoints only in B_i such that the following hold.

- The sets $B_0, B_1, \ldots, B_\ell, \bar{B}$ form a partition V.
- $\bar{B} \subseteq V - X - r$ (i.e. $\bar{B}$ is a subset of Steiner nodes).
- $h_0 = r$ and $h_i \in X$ for each $1 \leq i \leq \ell$.
- For every $0 \leq i \leq \ell$ and every $v \in B_i$, F_i contains an h_i, v-path.

We say that $\bar{B}$ is the set of *free Steiner nodes* in $\mathcal{T}$ and that h_i is the *head* of B_i for each $0 \leq i \leq \ell$. The edges of $\mathcal{T}$, denoted by $E(\mathcal{T})$, are simply $\cup_{i=0}^{\ell} F_i$. We say that $B_0, \ldots, B_\ell$ are the *components* of $\mathcal{T}$ where B_0 is the *root component* and $B_1, \ldots, B_\ell$ are the *non-root components*.

Figure 1 illustrates a partial Steiner tree. Note that if $\mathcal{T}$ is a partial Steiner tree with $\ell = 0$ non-root components, then $E(\mathcal{T})$ is in fact a feasible DST solution.

Finally, for a subset of edges F we let $\text{cost}(F) = \sum_{e \in F} c_e$.

2.2 High-level approach

Our algorithm builds up partial Steiner trees in an iterative manner while ensuring that the cost does not increase by a significant amount between iterations. Specifically, we prove the following lemma in Section 3. Recall that OPT_{LP} refers to the optimum solution value for (**DST-Primal**).

▶ **Lemma 3.** *Given a partial Steiner tree $\mathcal{T}$ with $\ell \geq 1$ non-root components, there is a polynomial-time algorithm that finds a partial Steiner tree $\mathcal{T}'$ with $\ell' < \ell$ non-root components such that*

$$\text{cost}(E(\mathcal{T}')) \leq \text{cost}(E(\mathcal{T})) + 2 \cdot OPT_{LP} \cdot \frac{\ell - \ell'}{\ell}.$$

Theorem 1 follows from Lemma 3 in a standard way.

Proof of Theorem 1. Initialize a partial Steiner tree $\mathcal{T}_k$ with k non-root components as follows. Let $\bar{B}$ be the set of all Steiner nodes, $B_0 = \{r\}$, and $F_0 = \emptyset$. Furthermore, label the terminals as $t_1, \ldots, t_k \in X$ and for each $1 \leq i \leq k$ let $B_i = \{t_i\}, h_i = t_i$ and $F_i = \emptyset$. Note that $\text{cost}(E(\mathcal{T}_k)) = 0$.

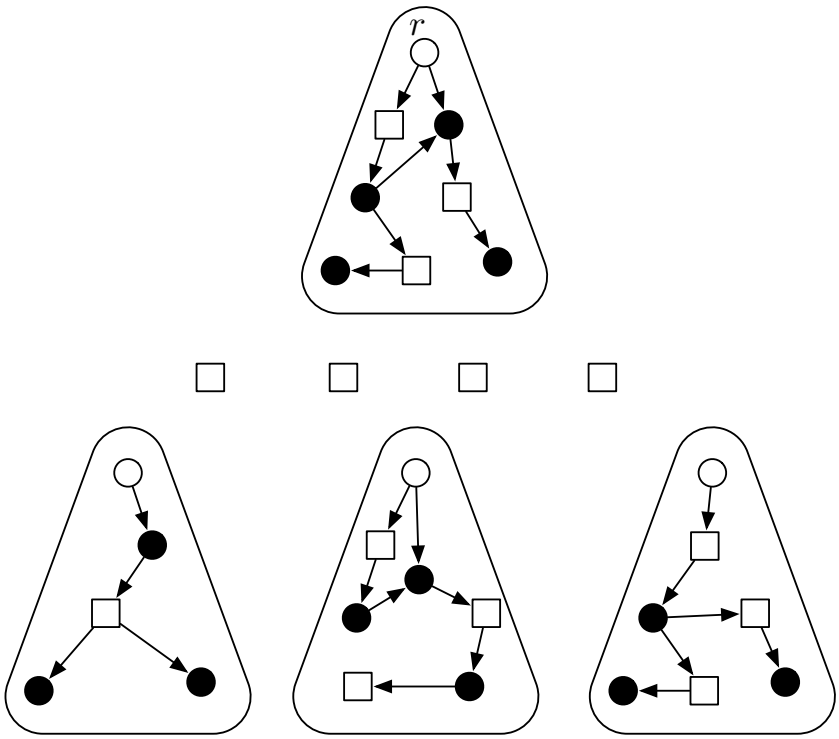

Figure 1 A partial Steiner tree with $\ell = 3$ non-root components (the root is pictured at the top). The only edges shown are those in some F_i. The white circles are the heads of the various sets B_i and the black circles are terminals that are not heads of any components. The squares outside of the components are the free Steiner nodes $\bar{B}$. Note, in particular, that each head can reach every node in its respective component. We do not require each F_i to be a minimal set of edges with this property.

Iterate Lemma 3 to obtain a sequence of partial Steiner trees $\mathcal{T}_{\ell_0}, \mathcal{T}_{\ell_1}, \mathcal{T}_{\ell_2}, \ldots, \mathcal{T}_{\ell_a}$ where $\mathcal{T}_{\ell_i}$ has ℓ_i non-root components such that $k = \ell_0 > \ell_1 > \ldots > \ell_a = 0$ and

$$\text{cost}(E(\mathcal{T}_{i+1})) \leq \text{cost}(E(\mathcal{T}_i)) + 2 \cdot OPT_{LP} \cdot \frac{\ell_i - \ell_{i+1}}{\ell_i}$$

for each $0 \leq i < a$. Return $E(\mathcal{T}_{\ell_a})$ as the final Steiner tree.

That $E(\mathcal{T}_a)$ can be found efficiently follows simply because we are iterating the efficient algorithm from Lemma 3 at most k times. The cost of this Steiner tree can be bounded as follows.

$$\begin{aligned}
\text{cost}(E(\mathcal{T}_a)) &\leq 2 \cdot OPT_{LP} \cdot \sum_{i=0}^{a-1} \frac{\ell_i - \ell_{i+1}}{\ell_i} = 2 \cdot OPT_{LP} \cdot \sum_{i=0}^{a-1} \sum_{j=\ell_{i+1}+1}^{\ell_i} \frac{1}{\ell_i} \\
&\leq 2 \cdot OPT_{LP} \cdot \sum_{i=0}^{a-1} \sum_{j=\ell_{i+1}+1}^{\ell_i} \frac{1}{j} = 2 \cdot OPT_{LP} \cdot \sum_{j=1}^{k} \frac{1}{k} \\
&= 2 \cdot OPT_{LP} \cdot H_k.
\end{aligned}$$

◀

The idea presented above resembles one proposed by Guha et al. [10] for bounding the integrality gap of a natural relaxation for *undirected* node-weighted Steiner tree by $O(\log k)$ [10]. Like our approach, Guha et al. also build a solution incrementally. In each *phase* of the algorithm, the authors reduce the number of connected components of a partial solution by adding vertices whose cost is charged carefully to the value of a dual LP solution that the algorithm constructs simultaneously.

3 A primal-dual proof of Lemma 3

Consider a given partial Steiner tree $\mathcal{T} = (\{B_i, h_i, F_i\}_{i=0}^{\ell}, \bar{B})$ with $\ell \geq 1$ non-root components. Lemma 3 promises a partial Steiner tree $\mathcal{T}'$ with $\ell' < \ell$ non-root components with $\text{cost}(E(\mathcal{T}')) \leq \text{cost}(E(\mathcal{T})) + 2 \cdot OPT_{LP} \cdot \frac{\ell - \ell'}{\ell}$. In this section we will present an algorithm that *augments* forest $\mathcal{T}$ in the sense that it computes a set of edges to *add* to $\mathcal{T}$. The proof presented here is constructive: we will design a *primal-dual* algorithm that maintains a feasible dual solution for (**DST-Dual**), and uses the structure of this solution to guide the process of adding edges to $\mathcal{T}$.

3.1 The algorithm

For any two nodes $u, v \in V$, let $d(u, v)$ be the cost of the cheapest u, v-path in G. More generally, for a subset $\emptyset \subsetneq S \subseteq V$ and a node $v \in V$ we let $d(S, v) = \min_{u \in S} d(u, v)$. We will assume that for every $0 \leq i \leq \ell$ and $1 \leq j \leq \ell, j \neq i$ that $d(B_i, h_j) > 0$ as otherwise, we could merge B_i and B_j by adding the 0-cost B_i, h_j-path to $\mathcal{T}$.

The usual conventions of primal-dual algorithms will be adopted. We think of such an algorithm as a continuous process that increases the value of some dual variables over time. At time $t = 0$, all dual variables are initialized to a value of 0. At any point in time, exactly ℓ dual variables will be raised at a rate of one unit per time unit. We will use Δ for the time at which the algorithm terminates. As is customary, we will say that an edge e *goes tight* if the dual constraint for e becomes tight as the dual variables are being increased. When an edge goes tight, we will perform some updates to the various sets being maintained by the algorithm. Again, the standard convention applies that if multiple edges go tight at the same time, then we process them in any order.

Algorithm 1 describes the main subroutine that augments the partial Steiner tree $\mathcal{T}$ to one with fewer components. It maintains a collection of *moats* $M_i \subseteq V - \{r\}$ and edges F_i' for each $1 \leq i \leq \ell$, while ensuring that the dual solution y it grows remains feasible. Mainly to aid notation, our algorithm will maintain a so called *virtual body* β_i for all $0 \leq i \leq \ell$ such that $B_i \subseteq \beta_i \subseteq B_i \cup \bar{B}$. We will ensure that each $v \in \bar{B} \cap \beta_i$ has a *mate* $u \in B_i$ such that the edge uv has cost no more than Δ. For notational convenience, we will let $\beta_0 = B_0$ be the virtual body of the root component. The algorithm will not grow a moat around the root since dual variables do not exist for sets containing the root.

Our algorithm will ensure that moats are pairwise *terminal-disjoint*. In fact, we ensure that any two moats may only intersect in $\bar{B}$. Terminal-disjointness together with the quasi-bipartite structure of the input graph will allow us to charge the cost of arcs added in the augmentation process to the duals grown.

An intuitive overview of our process is the following. At any time $t \geq 0$, the moats M_i will consist of all nodes v with $d(v, h_i) \leq t$. The moats M_i will be grown until, at some time Δ, for at least one pair i, j with $i \neq j$, there is a tight path connecting β_j to h_i. At this point the algorithm stops, and adds a carefully chosen collection of tight arcs to the partial Steiner tree that merges B_j and B_i (and potentially other components). Crucially, the cost

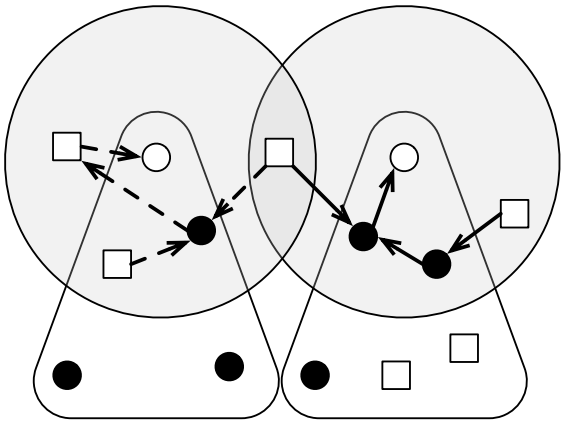

Figure 2 The moats around the two partial Steiner trees are depicted by the gray circles. The dashed edges are those bought by the first moat and the solid edges are those bought by the second moat. Note the moats only intersect in $\overline{B}$ (in particular, v is the only lying in both moats). Also, u lies in the virtual body for the left partial Steiner tree and the dashed arc entering u is coming from its mate. The edges F_i from the original partial Steiner trees are not shown.
Observe that if any edge entering v goes tight then it must be from either r or some terminal (because G is quasi-bipartite). This would allow us to merge at least one partial Steiner tree into the body of another.

of the added arcs will be *charged* to the value of the dual solution grown around the merged components.

Due the structure of quasi-bipartite graphs, we are able to ensure that in each step of the algorithm the active moats pay for at most one arc that is ultimately bought to form $\mathcal{T}'$. Also, if $\mathcal{T}'$ has $\ell' < \ell$ non-root components then each arc was paid for by moats around at most $\ell - \ell' + 1 \leq 2(\ell - \ell')$ different heads. So, the total cost of all purchased arcs is at most $2(\ell - \ell') \cdot \Delta$. Finally, the total dual grown is $\ell \cdot \Delta$, which is $\leq OPT_{LP}$ due to feasibility, so the cost of the edges bought can be bounded by $2\frac{\ell-\ell'}{\ell} \cdot OPT_{LP}$.

3.2 Algorithm and invariants

Now we will be more precise. The primal-dual procedure is presented in Algorithm 1. The following invariants will be maintained at any time $0 \leq t \leq \Delta$ during the execution of Algorithm 1.

1. For each $1 \leq i \leq \ell$, $h_i \in M_i$ and $M_i \subseteq V - \{r\}$ (so there is a variable y_{M_i} in the dual).
2. For each $1 \leq i \leq \ell$, $M_i = \{v \in V : d(v, h_i) < t\} \cup S$ where $S \subseteq \{v \in V : d(v, h_i) = t\}$.
3. $M_i \cap M_j \subseteq \bar{B}$ and both $\beta_i \cap \beta_j = M_i \cap \beta_j = \emptyset$ for distinct $0 \leq i, j \leq \ell$.
4. For each $1 \leq i \leq \ell$ we have $B_i \subseteq \beta_i \subseteq B_i \cup \bar{B}$. Furthermore, for each each $v \in \beta_i - B_i$ there is a *mate* $u \in B_i$ such that $uv \in E$ and $c_{uv} \leq t$.
5. y is feasible for LP (**DST-Dual**) with value exactly $\ell \cdot t$.

These concepts are illustrated in Figure 2.

3.3 Invariant analysis

▶ **Lemma 4.** *Invariants 1–5 are maintained by Algorithm 1 until the condition in the* ***if*** *statement in Step* (5) *is true. Furthermore, the algorithm terminates in* $O(n \cdot k)$ *iterations.*

Algorithm 1 Dual Growing Procedure

1: $M_i \leftarrow \{v \in V : d(v, h_i) = 0\}, 1 \leq i \leq \ell$
2: $\beta_i \leftarrow B_i$ for $0 \leq i \leq \ell$
3: $y \leftarrow \mathbf{0}$
4: Raise $y_{M_{i'}}$ uniformly for each moat $M_{i'}$ until some edge uv goes tight
5: **if** $u \in \beta_j$ for some $0 \leq j \leq \ell$ **and** $v \in M_{i'}$ for some $i' \neq j$ **then**
6: **return** the partial Steiner tree $\mathcal{T}'$ described in Lemma 6.
7: **else**
8: Let M_i be the unique moat with $uv \in \delta^{in}(M_i)$ ▷ cf. Proposition 5
9: $M_i \leftarrow M_i \cup \{u\}$
10: **if** $u \in \beta_i$ **then**
11: $\beta_i \leftarrow \beta_i \cup \{v\}$
12: **go to** Step (4)

Proof. Clearly the invariants are true after the initialization steps (at time $t = 0$), given that $d(B_i, h_j) > 0$ for any $i \neq j$. To see why Algorithm 1 terminates in a polynomial number of iterations, note that each iteration increases the size some moat by 1 and does not decrease the size of any moats. So after at most kn iterations some moat will grow to include the virtual body of another moat, at which point the algorithm stops.

Assume now that the invariants are true at some point just before Step (4) is executed and that the condition in Step (5) is false after Step (4) finishes. We will show that the invariants continue to hold just before the next iteration starts. We let uv denote the edge that went tight that is considered in Step (4). We also let t denote the total time the algorithm has executed (i.e. grown moats) up to this point.

Before proceeding with our proof, we exhibit the following useful fact. In what follows, let $M_j^{t'}$ be the moat around h_j at any time $t' \leq t$ during the algorithm. This proposition demonstrates how we control the overlap of the moats by exploiting the quasi-bipartite structure.

▶ **Proposition 5.** *If $uv \in \delta^{in}(M_i)$, then $uv \notin \delta^{in}(M_j^{t'})$ for any $j \neq i$, and for any $t' \leq t$.*

Proof. Suppose, for the sake of contradiction, that $uv \in \delta^{in}(M_j^{t'})$ for some $j \neq i$ and $t' \leq t$. Since $M_j^{t'}$ is a subset of M_j, the moat containing h_j at time t, we must have $v \in M_j \cap M_i$. Invariant 3 now implies that $v \in \bar{B}$. Since G is quasi-bipartite, then $u \in X$. Therefore $u \in B_{j'}$ for some j'. Since $j' \neq i$ or $j' \neq j$, then the terminating condition in Step (5) would have been satisfied as $u \in \beta_{j'}$. A contradiction. ◀

Following Proposition 5, we let i be the unique index such that $uv \in \delta^{in}(M_i)$ as in Step (8).

Invariant 1

First note that M_i never loses vertices during the algorithm's execution, and it therefore always contains head vertex h_i. Also, vertex u is not part of B_0 as otherwise the algorithm would have terminated in Step (5). Hence $M_i \cup \{u\}$ also does not contain the root node r.

Invariant 2

This is just a reinterpretation of Dijkstra's algorithm in the primal-dual framework (e.g. Chapter 5.4 of [13]), coupled with the fact that no edge considered in Step (4) in some iteration crosses more than one moat at any given time (Proposition 5).

Invariant 3

Suppose $(M_i \cup \{u\}) \cap M_j \not\subseteq \bar{B}$ for some $i \neq j$. This implies $u \in M_j \setminus \bar{B}$, and hence $u \in B_j \subseteq \beta_j$. Thus, the termination condition in Step (5) was satisfied and the algorithm should have terminated; contradiction.

If v is not added to β_i, and thus β_i remains unchanged, $\beta_i \cap \beta_j = M_j \cap \beta_i = \emptyset$ continues to hold for $j \neq i$. We also must have that $(M_i \cup \{u\}) \cap \beta_j = \emptyset$ for $i \neq j$, as otherwise $u \in \beta_j$ and this would violate the termination condition in Step (5).

Now suppose that v is added to β_i. Then for $j \neq i$ we still have $(\beta_i \cup \{v\}) \cap \beta_j = \emptyset$ as otherwise $v \in \beta_j$ which contradicts $v \in M_i$ and the fact that Invariant 3 holds at the start of this iteration. We also have that $M_j \cap (\beta_i \cup \{v\}) = \emptyset$ as otherwise $v \in M_j$. But this would mean that $u \in M_j$ as well by Proposition 5. We established above that $(M_i \cup \{u\}) \cap M_j \subseteq \overline{B}$. However, $\{u, v\} \subseteq (M_i \cup \{u\}) \cap M_j \subseteq \overline{B}$ contradicts the fact that G is quasi-bipartite.

Invariant 4

That $B_i \subseteq \beta_i$ is clear simply because we only add nodes to the sets β_i. Suppose now that v is added to β_i. In this case, $v \notin B_i$ as $B_i \subseteq \beta_i$ from the start. We claim that v can also not be part of B_j for some $j \neq i$, since otherwise $\emptyset \neq B_j \cap M_i \subseteq \beta_j \cap M_i$, contradicting Invariant 3. Hence $v \in \bar{B}$. Note that the quasi-bipartiteness of G implies that $u \in X$, and hence $u \in B_i$. Proposition 5 finally implies that only the moats crossed by uv are moats around i, so since the algorithm only grows one moat around i at any time we have $c_{uv} \leq t$, and this completes the proof of Invariant 4.

Invariant 5

The Step (4) stops the first time a constraint becomes tight, so feasibility is maintained. In each step, the algorithm grows precisely ℓ moats simultaneously. Because the objective function of (**DST-Dual**) is simply the sum of the dual variables, then the value of the dual is just ℓ times the total time spent growing dual variables. ◀

3.4 Augmenting $\mathcal{T}$

To complete the final detail in the description of the algorithm, we now show how to construct the partial Steiner tree after Step (5) has been reached. Lemma 4 shows that Invariants 1 through 5 hold just before Step (4) in the final iteration. Say the final iteration executes for δ time units and that uv is the edge that goes tight and was considered in Step (5).

▶ **Lemma 6.** *When Step* (6) *is reached in Algorithm 1, we can efficiently find a partial Steiner tree $\mathcal{T}'$ with $\ell' < \ell$ non-root components such that $cost(E(\mathcal{T}')) \leq cost(E(pt)) + 2\frac{\ell-\ell'}{\ell} \cdot OPT_{LP}$.*

Proof. Let j be the unique index such that $u \in \beta_j$ at time Δ. There is exactly one such j because $\beta_i \cap \beta_j = \emptyset$ for $i \neq j$ is ensured by the invariants. Next, let $J = \{i' \neq j : v \in M_{i'}\}$ and note that J consists of all indices i' (except, perhaps, j) such that $uv \in \delta^{in}(M_{i'})$. By the termination condition, $J \neq \emptyset$. Vertex u lies in β_j by definition. If $u \notin B_j$ then we let w be the mate of u as defined in Invariant 4. Otherwise, if $w \in B_j$, we let $w = u$.

For notational convenience, we let P_j be the path consisting of the single edge wu (or just the trivial path with no edges if $w = u$). In either case, say cost of P_j is $\Delta - \epsilon_j$ where $\epsilon_j \geq 0$ (cf. Invariant 4). For each $i' \in J$, let $P_{i'}$ be a shortest $v, h_{i'}$-path. Invariant 2 implies that

$$c(P_{i'}) = \Delta - \epsilon_{i'}, \tag{3}$$

for some $\epsilon_{i'} \geq 0$. Observe also that the tightness of uv at time Δ and the definition of J imply that

$$\sum_{i' \in J \cup \{j\}} \epsilon_{i'} \geq c_{uv}. \tag{4}$$

In fact, precisely a $\epsilon_{i'}$-value of the dual variables for $i' \neq j$ contribute to c_{uv}; the contribution of j's variables to c_{uv} is at most $\epsilon_{i'}$.

Construct a partial Steiner tree $\mathcal{T}'$ obtained from $\mathcal{T}$ and Algorithm 1 as follows.

- The sets $B_{j'}, F_{j'}$ and head $h_{j'}$ are unchanged for all $j' \notin J \cup \{j\}$.
- Replace the components $\{B_{i'}\}_{i' \in J \cup \{j\}}$ with a component $B := \bigcup_{i' \in J \cup \{j\}} (B_{i'} \cup V(P_{i'}))$ having head $h := h_j$. The edges of this component in $\mathcal{T}'$ are $F := \bigcup_{i' \in J \cup \{j\}} (F_{i'} \cup E(P_{i'})) \cup \{uv\}$.
- The free Steiner nodes $\bar{B}'$ of $\mathcal{T}'$ are the Steiner nodes not contained in any of these components.

Namely, $\bar{B}'$ consists of those nodes in $\bar{B}$ that are not contained on any path $P_{i'}, i' \in J \cup \{j\}$.

We show that Steiner tree $\mathcal{T}'$ as constructed above satisfies the conditions stated in Lemma 3. We first verify that $\mathcal{T}'$ as constructed above is indeed a valid partial Steiner tree. Clearly the new sets $\bar{B}', \{B_i\}_{i \notin J+j}$ and B partition V and $\bar{B}'$ is a subset of Steiner nodes.

Note that if $0 \in J \cup \{j\}$ in the above construction, then $j = 0$ because no moat contains r. Thus, if B_0 is replaced when B is constructed, then r is the head of this new component.

Next, consider any $b \in B$. If $b \in B_j$ then there is an h_j, b-path in $F_j \subseteq F$. If $b \in B_{i'}, i' \neq j$ then it can be reached from h_j in (B, F) as follows. Follow the h_j, w-path in F_j, then the w, u path P_j, cross the edge uv, follow $P_{i'}$ to reach $h_{i'}$, and finally follow the $h_{i'}, b$-path in $F_{i'}$. Finally, if $b \notin B_{i'}$ for any $i' \in J + j$ then b lies on some path $P_{i'}$, in which case it can be reached in a similar way.

It is also clear that $E(\mathcal{T}) \subseteq E(\mathcal{T}')$ and that the number of non-root components in $\mathcal{T}'$ is $\ell - |J| < \ell$. Also, $\text{cost}(E(\mathcal{T}')) - \text{cost}(E(\mathcal{T}))$ is at most the cost of the the paths $\{P_{i'}\}_{i' \in J+i}$ plus c_{uv}.

It now easily follows from (3) and (4) that

$$\sum_{i' \in J \cup \{j\}} \text{cost}(E(P_{i'})) + c_{uv} \leq \sum_{i' \in J \cup \{j\}} (\Delta - \epsilon_{i'}) + c_{uv} \leq (|J| + 1)\Delta \leq \frac{|J| + 1}{\ell} \cdot OPT_{LP}.$$

The last bound follows because the feasible dual we have grown has value $\ell \cdot \Delta \leq OPT_{LP}$. Let $\ell' = \ell - |J|$ be the number of nonroot components in $\mathcal{T}'$. Conclude by observing $|J| + 1 = \ell - \ell' + 1 \leq 2(\ell - \ell')$. ◀

To wrap things up, executing Algorithm 1 and constructing the partial Steiner tree as in Lemma 6 yields the partial Steiner tree that is promised by Lemma 3.

4 Conclusion

We have shown that the integrality gap of LP relaxation (**DST-Primal**) is $O(\log k)$ in quasi-bipartite instances of directed Steiner tree. The gap is known to be $\Omega(\sqrt{k})$ in 4-layered

instances [18] and $O(\log k)$ in 3-layered instances [7]. Since quasi-bipartite graphs are a generalization 2-layered instances, it is natural to ask if there is a generalization of 3-layered instances which has an $O(\log k)$ or even $o(\sqrt{k})$ integrality gap.

One possible generalization of 3-layered graphs would be when the subgraph of G induced by the Steiner nodes does not have a node with both positive indegree and positive outdegree. None of the known results on directed Steiner tree suggest such instances have a bad gap.

Even when restricted to 3-layered graphs, a straightforward adaptation of our algorithm that grow moats around the partial Steiner tree heads until some partial Steiner trees absorbs another fails to grow a sufficiently large dual to pay for the augmentation within any reasonable factor. A new idea is needed.

References

1. J. Byrka, F. Grandoni, T. Rothvoss, , and L. Sanita. Steiner tree approximation via iterative randomized rounding. *Journal of the ACM*, 60(1), 2013.
2. G. Calinescu and G. Zelikovsky. The polymatroid steiner problems. *J. Combinatorial Optimization*, 9(3):281–294, 2005.
3. M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, , and M. Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999.
4. I. Dinur and D. Steurer. Analytic approach to parallel repetition. In *In proceedings of STOC*, 2014.
5. J. Edmonds. Optimum branchings. *J. Res. Natl. Bur. Stand.*, 71:233–240, 1967.
6. U. Feige. A threshold of $\ln n$ for approximating set-cover. *Journal of the ACM*, 45(4):634–652, 1998.
7. Z. Friggstad, A. Louis, Y. K. Ko, J. Könemann, M. Shadravan, and M. Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *In proceedings of IPCO*, 2014.
8. M. X. Goemans, N. Olver, T. Rothvoss, and R. Zenklusen. Matroids and integrality gaps for hypergraphic steiner tree relaxations. In *In proceedings of STOC*, 2012.
9. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
10. S. Guha, A. Moss, J. Naor, and B. Scheiber. Efficient recover from power outage. In *In proceedings of STOC*, 1999.
11. E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *In proceedings of STOC*, 2003.
12. T. Hibi and T. Fujito. Multi-rooted greedy approximation of directed steiner trees with applications. In *In proceedings of WG*, 2012.
13. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
14. S. Rajagopalan and V. V. Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. In *In proceedings of SODA*, 1999.
15. T. Rothvoss. Directed steiner tree and the lasserre hierarchy. Technical report, CORR abs/1111.5473, 2011.
16. V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.
17. A. Zelikovsky. A series of approximation algorthms for the acyclic directed steiner tree problem. *Algorithmica*, 18:99–110, 1997.
18. L. Zosin and S. Khuller. On directed steiner trees. In *In proceedings of SODA*, 2002.