

# $O(\log^2 k / \log \log k)$ -Approximation Algorithm for Directed Steiner Tree: A Tight Quasi-Polynomial-Time Algorithm

STOC 2019: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing

Fabrizio Grandoni   Bundit Laekhanukit   Shi Li

Final Presentation, Special Topics on Graph Algorithms, Spring 2021  
B07902024 塗大為   B07902133 彭道耘   B07902134 黃于軒   B07902141 林庭風

June 1st, 2021

# Overview

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results

# Introduction

For a directed graph with  $k$  terminals, the paper [GLL18] presents:

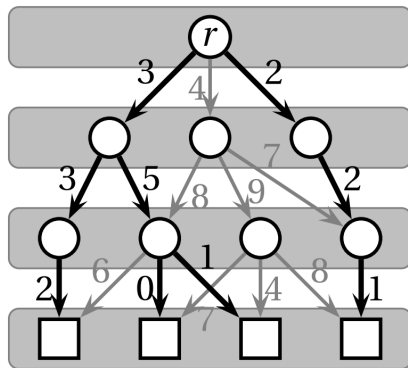
- An  $O(\log^2 k / \log \log k)$ -approximation algorithm for DIRECTED STEINER TREE (DST) in quasi-polynomial time. (Section 3, 4, 5, 6)
- Under certain conjectures,  $O(\log^2 k / \log \log k)$  is the optimal approximation ratio for quasi-polynomial time algorithms. (Section 7)

## Definition 1

Given a weighted directed graph  $G$ , a root  $r \in V(G)$  and a set of terminals  $K \subseteq V(G) \setminus \{r\}$ , a directed steiner tree  $T$  is an aborescence (directed tree) rooted at  $r$  that contains all the terminals.

The goal of **DIRECTED STEINER TREE** is to find such  $T$  with minimum cost. Throughout the presentation, we will additionally assume that the edge weights in the input graph satisfy triangle-inequality, without loss of generality.

# Directed Steiner Tree



1  
,

## Definition 2

Quasi-polynomial-time algorithms run in  $O(2^{\text{polylog}(n)})$  time. More precisely,  $\text{QP} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{O(\log^c n)})$ .

## Definition 3

$\text{ZPTIME}(f(n))$  refers to randomized algorithms that always return the correct answer and have randomized running times with expectation  $O(f(n))$ .

## Remark

*It will be shown that, assuming the Projection Game Conjecture (which will be stated later) and  $\text{NP} \not\subseteq \bigcap_{0 < \epsilon < 1} \text{ZPTIME}(2^{n^\epsilon})$ , the optimal approximation ratio for quasi-polynomial time algorithms is  $O(\log^2 k / \log \log k)$ .*



# Sketch of Approach

- 1 DIRECTED STEINER TREE (DST)  $\iff$  DECOMPOSITION TREE
- 2 DECOMPOSITION TREE  $\iff$  LABEL-CONSISTENT SUBTREE (LCST)
- 3 Integer linear programming formulation of LCST
- 4 Sherali-Adams lifting of the corresponding relaxed LP formulation
- 5  $O(\log^2 k / \log \log k)$ -approximation of the ILP instance in  $O(n^{\log^5 k})$

# Sherali-Adams Lifting

A *lift and project* method to solve ILP, originally introduced in [SA90].

- 1 Formulate (relaxed) LP problem, resulting in a relaxed polytope containing the integer polytope.
- 2 Apply Sherali-Adams to "tighten" the polytope.
- 3 It can be shown that, after enough runs of the previous step, the integer polytope is obtained.
- 4 The idea is to run "the right number of" rounds such that the resulting polytope is "somewhat tight" but yet not very difficult to solve.

For the  $R$ -th round, Sherali-Adams adds the variables  $x_S = \prod_{i \in S} x_i$  for every subset  $S \subseteq [n]$  of size not exceeding  $R$  and replaces the original constraints accordingly.

# Sherali-Adams Lifting

For a polytope  $\mathcal{P}$ ,  $\text{SA}(\mathcal{P}, R)$  denotes the polytope tightened by the  $R$ -th round of Sherali-Adams lift. Additionally, the hierarchy makes *conditioning* on an event possible

## Definition 4

Let  $x \in \text{SA}(\mathcal{P}, R)$  for  $R \geq 1$ , for  $x_i > 0$  define  $x' \in \text{SA}(\mathcal{P}, R-1)$  to be  $x$  *conditioned* on  $x_i$  as

$$x'_S = \frac{x_{S \cup \{i\}}}{x_i}$$

for  $S \in \binom{[n]}{R-1}$ .

## Definition 5

Given a rooted tree  $T$  and a vertex  $v \in V(T)$ , we use  $\text{root}(T)$  to denote the root of  $T$  and  $T[v]$  to denote the subtree containing  $v$  and all of its descendants.

## Definition 6

The rooted trees under discussion are out-arborescences, i.e., edges are directed toward the leaves. For a directed edge  $e = (u, v)$ , we define  $\text{head}(e) = u$  and  $\text{tail}(e) = v$ .

- 1 Introduction
- 2 Previous & Related Works**
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results

## Previous & Related Works

# Comparison to Previous Work

- The best polynomial-time approximation algorithm for DST achieves an  $O((1/\epsilon)^3 k^\epsilon)$ -approximation ratio in  $O(n^{1/\epsilon})$ -time, due to Charikar *et al.* [Cha+99]
- The previously best approximation algorithm for DST in quasi-polynomial-time achieves ratio  $O(\log^3 k)$ , due to Charikar *et al.* [Cha+99] as well.
- Recursive greedy algorithm of Chekuri and Pal for GST<sup>2</sup> [CP05]
  - The first one that yields an approximation ratio of  $O(\log^2 k)$  for GST in quasi-polynomial-time.
  - Their algorithm exploits that any optimal solution can be shortcut into a path of length  $k$ , while paying only a factor of 2. (such a path exists in the metric-closure of the input graph).
- Hierarchy based LP-rounding techniques by Rothvoß [Rot12].
  - Handle the dependency rules.

---

<sup>2</sup>GST can be regarded as a special instance of DST. See Definition 28 for more. 

- An  $O(\log^2 k \log n)$ -approximation for GST in polynomial time, where  $k$  is the number of groups by Garg *et al.* [GKR98]
  - Map the input instance into a tree instance by invoking the Probabilistic Metric-Tree Embeddings.
  - Apply an elegant LP-based randomized rounding algorithm.



- $\ell$ -DST and  $\ell$ -GST, survivable network variants of DST and GST.
  - $\ell$  edge-disjoint directed (resp., undirected) paths from the root to each terminal (resp., group).
  - There is no  $2^{\log^{1-\epsilon} n}$ -approximation, for any  $\epsilon > 0$ , unless  $\text{NP} \subseteq \text{DTIME}(2^{\text{polylog}(n)})$ , by Cheriyan *et al.* [Che+14]
  - There is no  $\ell^{1/2-\epsilon}$ -approximation, for any constant  $\epsilon > 0$ , unless  $\text{NP} = \text{ZPP}$ , by Laekhanukit [Lae14]
  - Gupta *et al.* [GKR10] presented a  $\tilde{O}(\log^3 n \log k)$ -approximation algorithm for 2-GST.
  - Chalermsook *et al.* presented an LP-rounding bicriteria approximation algorithm for  $\ell$ -GST that returns a subgraph with cost  $O(\log^2 n \log k)$  times the optimum while guaranteeing a connectivity of at least  $\Omega(\ell / \log n)$ .

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree**
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results

# Decomposition Tree

## Definition 7

A decomposition tree  $\tau$  of  $G$  is a rooted tree where

- $\alpha \in V(\tau)$  is associated with  $\mu_\alpha \in V(G)$
- Leaf  $\beta \in V(\tau)$  is associated with  $e_\beta \in E(G)$  and  $\mu_\beta = \text{head}(e_\beta)$
- For  $\alpha_2$  being a child of  $\alpha$ , there is a child  $\alpha_1$  of  $\alpha$  with
  - $\mu_\alpha = \mu_{\alpha_1}$
  - $\mu_{\alpha_2}$  is *involved* in  $\tau[\alpha_1]$

The cost of  $\tau$  is the sum of costs of the edges corresponding to its leaves.

## Definition 8

A vertex  $v \in V(G)$  is *involved* in  $\tau[\alpha]$  if either

- $\mu_\alpha = v$
- There is a leaf  $\beta \in \tau[\alpha]$  with  $v = \text{tail}(e_\beta)$

# Decomposition Tree

Intuitively, for a subgraph  $G' \subseteq G$ , a decomposition tree of  $G'$  represents the process of recursively dividing  $E(G')$  until the number of edges becomes one. Additionally,

- The decomposition is associated with a *root*  $r$ .  $r$  is the head of at least one edge in the current edge set.
- At the decomposition step, each sub-instance created either has the same root  $r'$  as the current instance, or  $r'$  is the tail of an edge in the edge set of another sub-instance having the same root as the current instance. This ensures that the sub-instance is *reachable* from the current root.

The followings will be shown

- Given a directed steiner tree  $T$ , there exists a decomposition tree  $\tau$  of  $T$  rooted at  $r$  with  $\text{cost}(\tau) \leq \text{cost}(T)$ . All the terminals are *involved* in  $\tau$ . Additionally,  $\tau$  is a full binary tree of height  $O(\log k)$ .
- Given a decomposition tree  $\tau$  rooted at  $r$  of some unknown subgraph in  $G$  that *involves* all terminals, there exists a directed steiner tree  $T$  with  $\text{cost}(T) \leq \text{cost}(\tau)$ .

Therefore, finding the minimum DST can be reduced to finding the minimum decomposition tree  $\tau^*$  rooted at  $r$  in  $G$  *involving* all  $k$  terminals. Moreover, such decomposition tree will be a full binary tree and have height  $O(\log k)$ .

# DST to Decomposition Tree

Let  $T$  be a directed steiner tree. Since triangle-inequality is satisfied, the internal vertices in  $T$  have out-degree at least two and thus  $|V(T)| \leq 2k$ . The following lemma was proved in class

## Lemma 9

*For a tree  $T$  of  $n$  edges, there exists a vertex  $v \in V(T)$  such that  $T$  can be decomposed into two subtrees  $T_1 = T[v]$  and  $T_2 = T \setminus T[v]$  and*

$$\frac{n}{3} \leq E(T_i) \leq \frac{2n}{3}$$

*holds for  $i \in \{1, 2\}$ .*

# DST to Decomposition Tree

$\tau$  can be constructed by recursively decomposing  $T$  into balanced subtrees.  
In particular, by setting  $\tau \leftarrow \text{DECOMPOSE}(T, r)$

---

---

**procedure**  $\text{DECOMPOSE}(T, r)$

$\alpha \leftarrow$  a new vertex with  $\mu_\alpha = r$

**if**  $T = \{(r, v)\}$  **then**

$e_\alpha = v$

**return** the decomposition tree with a single vertex  $\alpha$

**end if**

Let  $v$  be the vertex with  $\frac{|T|}{3} \leq |T[v]| \leq \frac{2|T|}{3}$  by Lemma 9

$\tau_1 \leftarrow \text{DECOMPOSE}(T \setminus T[v], r)$

$\tau_2 \leftarrow \text{DECOMPOSE}(T[v], v)$

Set  $\text{root}(\tau_1)$  and  $\text{root}(\tau_2)$  to be children of  $\alpha$

**return** the decomposition tree rooted at  $\alpha$

**end procedure**

---



# DST to Decomposition Tree

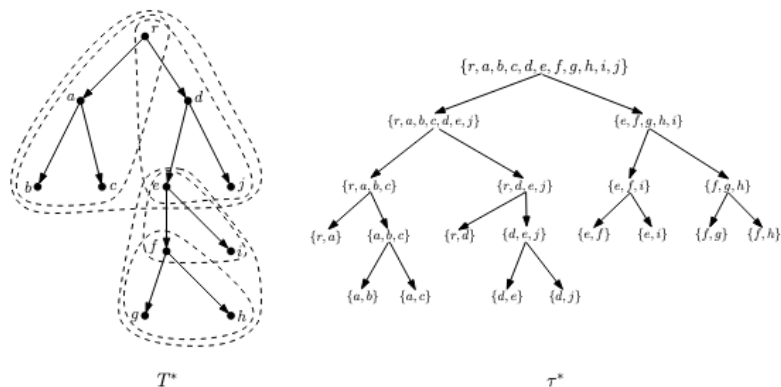


Figure: The steiner tree  $T^*$  and the decomposition tree  $\tau^*$  that it maps to. Each time a balanced subtree is peeled of and the two parts are recursively divided.

3

# DST to Decomposition Tree

Let's verify the outcome indeed has the desired property:

- The cost of  $\tau$  is trivially the same as the cost of  $T$  because each decomposition is disjoint.
- Since  $|V(T)| \leq 2k$  and the size of the edge set reduces by a constant factor at each recursion, the height of  $\tau$  is  $O(\log k)$ .
- Every internal vertex of  $\tau$  has exactly two children, hence  $\tau$  is a fully binary tree.

# Decomposition Tree to DST

Let  $\tau$  be a decomposition tree in  $G$  that involves all terminals with  $\mu_{\text{root}(\tau)} = r$ . We'd like to show

## Lemma 10

*Let  $\alpha \in V(\tau)$  and  $G_\alpha$  be the subgraph induced by edges at the leaves in  $\tau[\alpha]$ . Then, there exists a directed path from  $\mu_\alpha$  to every vertex in  $G$  that is involved in  $\tau[\alpha]$ .*

Let  $E'$  be the edges at the leaves of  $\tau$  and  $G'$  be the subgraph induced by  $E'$ . By Lemma 10, all terminals are reachable from  $\mu_{\text{root}(\tau)} = r$  in  $G'$ . Thus, by removing redundant edges in  $G'$  a directed steiner tree with cost not exceeding  $\text{cost}(\tau)$  is constructed.

# Decomposition Tree to DST

Let's prove the Lemma by induction on the height of  $\alpha$ :

- **$\alpha$  is a leaf**: Only  $\text{head}(e_\alpha)$  and  $\text{tail}(e_\alpha)$  are involved in  $\tau[\alpha]$  and they are both reachable from  $\mu_\alpha = \text{head}(e_\alpha)$  using  $e_\alpha$ .
- **$\alpha$  is an internal vertex**: Let  $v$  be a vertex that is involved in  $\tau[\alpha]$ . There are two possibilities:
  - $v = \mu_\alpha$ : Then it is trivially reachable from itself.
  - $v = \text{tail}(e_\beta)$  for  $\beta \in \tau[\alpha]$ : Let  $\alpha_2$  be the child of  $\alpha$  such that  $\beta \in \tau[\alpha_2]$  and hence  $\beta$  is involved in  $\tau[\alpha_2]$  as well. If  $\mu_{\alpha_2} = \mu_\alpha$ , by inductive hypothesis,  $\beta$  is reachable from  $\mu_\alpha$ .

Otherwise,  $\mu_{\alpha_2}$  is involved in  $\tau[\alpha_1]$  for some child  $\alpha_1$  with  $\mu_{\alpha_1} = \mu_\alpha$ . By inductive hypothesis, there exists a path  $P$  from  $\mu_{\alpha_1}$  to  $\mu_{\alpha_2}$  and a path  $Q$  from  $\mu_{\alpha_2}$  to  $\beta$ . Concatenating  $P$  and  $Q$  raises the desired path.

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree**
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results

# Label-Consistent Subtree

# Label-Consistent Subtree

## Definition 11

Let  $T^0$  be a rooted tree with cost  $c_v$  on vertex  $v$ . Let  $L$  be the set of labels. Each vertex  $v \in V(T^0)$  is associated with two sets, the *service*  $\text{ser}(v) \subseteq L$  and the *demand*  $\text{dem}(v) \subseteq L$ . A subtree  $T$  of  $T^0$  with  $\text{root}(T) = \text{root}(T^0)$  is *label-consistent* if

- For all vertices  $v \in V(T)$  and  $l \in \text{dem}(v)$ , there exists a descendant  $u$  of  $v$  in  $T$  such that  $l \in \text{ser}(u)$ . That is, each demand of  $v$  is supplied by some descendant of  $v$ .

## Definition 12

Let  $K \subseteq L$  be the set of *global labels*. Symmetrically,  $L \setminus K$  is the set of *local labels*. The goal of LABEL-CONSISTENT SUBTREE is to find a *label-consistent* subtree  $T^*$  with minimum cost such that all global labels are supplied.

# Sketch of Reduction

For an instance of DST, we'd like to find a desired decomposition tree by constructing an instance of LCST  $\mathcal{T}^0$  such that

- All possible full binary decomposition trees of height  $O(\log k)$  are embedded in  $\mathcal{T}^0$ .
- To achieve better approximation ratio, each decomposition tree is divided into *twigs* of heights  $O(\log \log k)$ .
- The structure and property of decomposition trees are guaranteed by the label-consistencies of subtrees.



# Sketch of Reduction

In particular,  $\mathcal{T}^0$  will have the following property

- Height  $h = O(\log k / \log \log k)$
- $s := \max_{v \in V(\mathcal{T}^0)} |\text{dem}(v)|$  is  $O(\log k)$
- The number of vertices  $N = n^{O(\log^2 k / \log \log k)}$
- The size of global labels  $|K|$  is the same as the number of terminals  $k$ .
- A solution to LCST can be converted to a decomposition tree  $\tau$  with the same cost that involves all terminals, and vice versa.

With

## Theorem 13

*There is an  $(shN)^{O(sh^2)}$ -time  $O(h \log k)$ -approximation algorithm for the Label-Consistent Subtree problem.*

We get an  $O(h \log k) = O(\log^2 k / \log \log k)$ -approximation algorithm for DST that runs in  $(shN)^{O(sh^2)} = n^{O(\log^5 k)} = 2^{O(\log^6 n)}$ -time.

A decomposition tree is divided into twigs and embedded in the LCTS instance. Specifically

## Definition 14

Let  $g := \lceil \log_2 \log_2 k \rceil$ . A twig  $\eta$  is a full binary tree of height at most  $g$ , where

- Each  $\alpha \in V(\eta)$  is associated with  $\mu_\alpha \in V(G)$
- Each leaf  $\beta \in V(\eta)$  may or may not have an associative  $e_\beta \in E(G)$ , and if it does,  $\mu_\beta = \text{head}(e_\beta)$

We can think of leaves  $\beta$  without  $e_\beta$  as internal vertices in the decomposition tree  $\tau$  to which  $\eta$  corresponds; while leaves with  $e_\beta$  map to leaves in  $\tau$ .

# Construction of $T^0$

Let  $\bar{h} = O(\log k)$  be the upper-bound of  $\text{height}(\tau^*)$ . The height of the twig  $\tau^*$  maps to is then upper-bounded by  $\lceil \bar{h}/g \rceil$ .

$T^0$  is constructed by calling  $\text{CONSTRUCTLCST}(r, 0)$ . The set of global labels is identical to the set of terminals.

- 
- 1: **procedure**  $\text{CONSTRUCTLCST}(u, j)$
  - 2:     Let  $p$  be a new node with  $c_p = 0$  and  $\text{dem}(p) = \{\ell\}$ , where  $\ell$  is a new local label
  - 3:     **if**  $j < \lceil \bar{h}/g \rceil$  **then**
  - 4:         **for** each possible non-singular twig  $\eta$  with  $\mu_{\text{root}(\eta)} = u$  **do**
  - 5:              $\text{ADDCHILD}(p, \ell, \eta)$
  - 6:         **end for**
  - 7:     **end if**
  - 8: **end procedure**
-

# Construction of $\mathcal{T}^0$

- 
- 1: **procedure** ADDCHILD( $p, \ell, \eta$ )
  - 2:     Let  $q$  be a new child of  $p$  with  $c_q = \sum_{\beta \in \eta} c_\beta$ ,  $\text{ser}(q) = \{\ell\}$ ,  
     $\text{dem}(q) = \emptyset$
  - 3:     **for** leaf  $\beta$  of  $\eta$  **do**
  - 4:         **if**  $e_\beta$  is defined **then**
  - 5:             **if**  $\text{tail}(e_\beta) \in K$  **then** Add global label  $\text{tail}(e_\beta)$  to  $\text{ser}(q)$
  - 6:             **end if**
  - 7:         **else**
  - 8:              $\mathbf{T}_\beta^q \leftarrow \text{CONSTRUCTLCST}(\mu_\beta, j + 1)$
  - 9:             Set  $\text{root}(\mathbf{T}_\beta^q)$  to be  $q$ 's child
  - 10:            Create a new local label  $\ell'$  and add it to  $\text{dem}(q)$  and  
     $\text{ser}(\text{root}(\mathbf{T}_\beta^q))$
  - 11:         **end if**
  - 12:     **end for**
  - 13:     ENSURESTRUCTURE( $p, \eta, q$ )
  - 14: **end procedure**

# Construction of $\mathcal{T}^0$

---

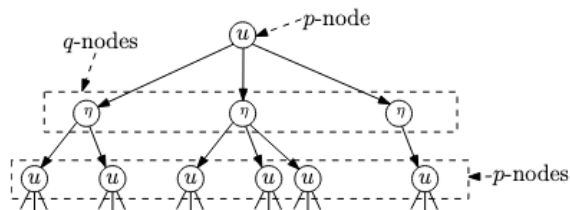
```
1: procedure ENSURESTRUCTURE( $p, \eta, q$ )
2:   for internal node  $\alpha$  of  $\eta$  do
3:     Let  $\alpha_1$  be a child of  $\alpha$  with  $\mu_{\alpha_1} = \alpha_2$  and  $\alpha_2$  be the other child
4:     if  $\mu_{\alpha_2} \neq \mu_\alpha$  and  $\nexists$  leaf  $\beta \in \eta[\alpha_1]$  with  $\text{tail}(e_\beta) = \mu_{\alpha_2}$  then
5:       Create a new local label  $\ell'$  and add it to  $\text{dem}(q)$ 
6:       for leaf  $\beta$  of  $\eta[\alpha_1]$  with  $e_\beta$  undefined,  $q' \in \mathbf{T}_\beta^q$  do
7:         if  $\eta_{q'}$  has a leaf  $\beta'$  with  $\text{tail}(e_{\beta'}) = \mu_{\alpha_2}$  then
8:           Add  $\ell'$  to  $\text{ser}(q')$ 
9:         end if
10:      end for
11:    end if
12:  end for
13: end procedure
```

---

# Construction of $T^0$ - Intuition

- The height (considering the number of twigs) of  $T^0$  is bounded by the  $j < \lceil \bar{h}/g \rceil$  guard.
- Each call to `CONSTRUCTLCST` is a two-level expansion:
  - The first level (L2 in `CONSTRUCTLCST`) corresponds a vertex  $u \in V(G)$  (referred to as  $p$ -nodes)
  - The second level (L2 in `ADDCHILD`) corresponds to a twig rooted at  $u$  (referred to as  $q$ -nodes)
  - We must choose at least one such twig if  $u$  is chosen (enforced by the local label  $\ell$ )

# Construction of $T^0$ - Intuition



**Figure:** Single step of recursion made in `ADDCHILD`. Each  $q$ -node is associated with a twig  $\eta_q$  where  $\mu_{\text{root}(\eta_q)}$  being the vertex  $u_p$  its parent  $p$  corresponds to.

4

<sup>4</sup>GLL18.



# Construction of $\mathcal{T}^0$ - Intuition

- A label-consistent subtree is a valid decomposition tree:
  - If the leaf node  $\beta$  of  $\eta$  does not have a well-defined  $e_\beta$ , we must further expand it, as enforced by the local label  $\ell'$  at L10 in `ADDCHILD`
  - $\alpha_2$  is involved in  $\eta[\alpha_1]$  because the local label  $\ell'$  at L5 in `ENSURESTRUCTURE` is eventually supplied by one of the descendants that points to  $\mu_{\alpha_2}$
- A label-consistent subtree that supplies all global labels map to a decomposition tree in which all terminals are involved since there is a one-to-one correspondence between the global labels and the terminals.

# Properties of $T^0$

- Obviously, the height is by construction  $O(\bar{h}/g) = O(\log k / \log \log k)$
- For a node  $v$  in  $T^0$ 
  - If  $v$  is a  $q$ -node, then  $|\text{dem}(v)|$  is bounded by the number of vertices in a twig, which is  $O(2^{\log \log k}) = O(\log k)$
  - If  $v$  is a  $p$ -node, then  $|\text{dem}(v)| = 1$
- The size of  $T^0$  is dominated by the number of branches of  $p$ -nodes. For  $u \in V(G)$ , the number of twigs rooted at  $u$  is bounded by
  - $(2^g)^2 = 2^{2g}$ : the number of shapes of twigs, times
  - $(n^2)^{2^g} = n^{2 \cdot 2^g}$ : the number of ways to assign  $\mu_*$  and  $e_*$  to vertices in a twig

which is roughly  $\log k \times n^{\log k} = n^{O(\log k)}$ . Thus,

$$N \leq \left( n^{O(\log k)} \right)^{\log k / \log \log k} = n^{O(\log^2 k / \log \log k)}$$

- The set of global labels is the same as the set of terminals

# Decomposition Tree to LCST

Given the optimal decomposition tree  $\tau^*$ , we can locate its corresponding  $T^*$  in  $T^0$  as follows

- Decompose  $\tau$  into twigs at vertices of depths  $ig$  for some  $i$ . That is, if vertex  $v$  is of depth  $ig$ , then the twig it corresponds to contains descendants of  $v$  of depth  $ig, ig + 1, \dots, (i + 1)g$
- Locate  $T^*$  recursively, starting from  $r$ . At vertex  $v$  of depth  $ig$ , add the nodes  $p$  corresponding to  $v$  and  $q$  corresponding to twigs rooted at  $v$  into  $T^*$ . For each descendant  $u$  of  $v$  at depth  $(i + 1)g$ , there will be a child  $p'$  of  $q$  associated to  $u$  and we keep the locating process on  $u$ .

# Decomposition Tree to LCST

In particular, the following can be easily shown by comparing the required property of decomposition tree and the construction of  $T^0$

## Lemma 15

*$T^*$  is a label-consistent subtree of  $T$  with  $\text{cost}(T^*) = \text{cost}(\tau^*)$ . Moreover, all global labels are supplied by  $T^*$ .*

# LCST to Decomposition Tree

Conversely, we need to show the following

## Lemma 16

*Given any feasible solution  $T$  to the LCST instance  $T^0$ , in time  $\text{poly}(|V(T)|)$  we can construct a decomposition tree  $\tau$  with  $\text{cost}(\tau) = \text{cost}(T)$ . Moreover, if a global label  $v \in K$  is supplied by  $T$ , then  $\tau$  involves  $v$ .*

Let  $\mathcal{C}$  be the set of twigs contained in  $T$ . For a  $q$ -node  $q$  in  $T$  and the twig  $\eta_q$  it corresponds to, consider its child  $p$  (a  $p$ -node) and  $p$ 's child  $q'$ .  $\mu_{\text{root}(\eta_{q'})}$  will be  $\mu_\beta$  where  $\beta$  is the leaf in  $\eta_q$  related to  $p$ .  $\tau$  is then constructed by connecting  $\eta_q$  and  $\eta_{q'}$  at  $\mu_\beta$  for all such  $q$  and  $q'$ . The cost and the structure of  $\tau$  can be easily argued by the way we construct  $T^0$  and the placement of local labels.

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree**
- 6 LP Relaxation
- 7 Hardness Results

# Approximation Algorithm for Label-Consistent Subtree

In the following few sections, we will derive the following theorem.

## Theorem 17

*There is an  $(shN)^{O(sh^2)}$ -time  $O(h \log k)$ -approximation algorithm for the Label-Consistent Subtree problem, where  $s := \max_{v \in V(T^0)} |\text{dem}(v)|$ .*



To make the statement clearer and easier, we will, without loss of generality, transform the input LCTS instance to the one satisfying the following properties:

- $\text{dem}(u)$  and  $\text{dem}(v)$  are disjoint for all  $u, v \in V(T^0)$ .
  - Make copies of duplicate labels.
- Demand labels are only located at the internal nodes.
  - Demand labels at leaves are either irrelevant or never supplied
- Service labels are only located at the leaves, and each leaf contains exactly one service label.
  - Attach  $|\text{ser}(v)|$  leaves with cost 0 to  $v$  and distribute the service labels to them.

- With above simplifications, we could redefine the LCST problem. Let  $V^{\text{leaf}}$  and  $V^{\text{int}}$  be the sets of leaves and internal nodes. Let  $\Lambda_v$  be the set of children of  $v$ . Let  $\Lambda_v^{\text{leaf}} = V(T^0[v]) \cap V^{\text{leaf}}$ .
- We need to find a minimum cost subtree  $T$  of  $T_0$  such that  $\text{root}(T) = \text{root}(T^0)$  and for all  $\ell \in K$  there exists  $v \in V(T) \cap V^{\text{leaf}}$  with  $a_v = \ell$ , where  $a_v$  is the unique label in  $\text{ser}(v)$ .

Consider the change in the size and height of  $T^0$  after the transformation. Let  $h'$  and  $N'$  be that of the old  $T_0$ .

- $h \leq h' + 1$ .
- The number of leaves in the new  $T^0$  is at most  $s(h' + 1)N'$ , so  $N = O(sh'N')$ .

For an optimum tree  $T^*$  with cost  $\text{opt}$ , we can assume that

- For every  $\ell \in K$ , there exists exactly one node  $v \in V(T^*) \cap V^{\text{leaf}}$  with  $a_v = \ell$ .
- For every  $\ell \in L \setminus K$ , there is at most one node  $v \in V(T^*) \cap V^{\text{leaf}}$  with  $a_v = \ell$ .

# Randomized Algorithm

The algorithm is based on the following theorem

## Theorem 18

*There is an  $(sN)^{O(sh^2)}$ -time algorithm that outputs a random label-consistent tree  $\tilde{T}$  such that  $\mathbb{E}(c(\tilde{T})) \leq \text{opt}$ , and for every  $\ell \in K$ , we have  $\mathbb{P} \left[ \exists v \in V^{\text{leaf}} \cap V(\tilde{T}) \mid a_v = \ell \right] \geq \frac{1}{h+1}$ .*

We run  $O(h \log k)$  times the algorithm above and let  $T'$  be the union of all  $\tilde{T}$ . The expected cost of  $T'$  is at most  $O(h \log k) \text{opt}$ , and by the union bound, we have

$$\mathbb{P} \left[ \forall \ell \in K, \exists v \in V^{\text{leaf}} \cap V(T') \mid a_v = \ell \right] \geq \frac{1}{2}$$

In expectation, we just need to run the procedure twice.

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation**
- 7 Hardness Results

# LP Relaxation

To construct the algorithm, we first transform the problem into *Linear Programming Problem*. We formulate an LP relaxation to find  $T^*$ .

- $\mathbb{D} := V(T^0) \cup (V(T^0) \times L)$  is the set of variables.  $x_e \in \{0, 1\}$  for all  $e \in \mathbb{D}$ .
  - $u \in V(T^0)$  iff  $u \in V(T^*)$ .
  - $(u, \ell) \in V(T^0) \times L$  iff  $u \in V(T^*)$  and  $\Lambda_u^{\text{leaf}} \cap V(T^*)$  has a node with label  $\ell$ .

The following constraints must hold.


- 1  $x_v \leq x_u, \quad \forall u \in V^{\text{int}}, v \in \Lambda_u.$ 
  - The children can not be chosen if the parent is not
- 2  $x_{(u,\ell)} \leq x_u, \quad \forall u \in V(T_0), \ell \in L.$
- 3  $x_{(u,\ell)} = x_u, \quad u \in V^{\text{int}}, \ell \in \text{dem}(u).$ 
  - If  $u$  is present, labels it demands must be present as well
- 4  $x_{(v,a_v)} = x_v, \quad \forall v \in V^{\text{leaf}}.$
- 5  $x_{(u,\ell)} = \sum_{v \in \Lambda_u} x_{(v,\ell)}, \quad \forall u \in V^{\text{int}}, v \in L.$
- 6  $x_{(v,\ell)} = 0, \quad \forall v \in V^{\text{leaf}}, \ell \neq a_v.$
- 7  $x_{(\text{root}(T^0),\ell)} = 1, \quad \forall \ell \in K.$ 
  - Global labels must be supplied



Let  $\mathcal{P}$  be the polytope containing all vectors  $x \in [0, 1]^{\mathbb{D}}$  satisfying constraints above.

With Sherali-Adams hierarchy, we can find a solution  $x^* \in SA(\mathcal{P}, O(sh^2))$  satisfying the lifted constraints with  $\sum_{v \in V(T^0)} c_v x_v^* \leq \text{opt}$  in time  $|\mathbb{D}|^{O(sh^2)} = (sN)^{O(sh^2)}$ , using any polynomial-time linear programming algorithm.<sup>5</sup>

---

<sup>5</sup>The state-of-the-art algorithm due to [Jia+20] runs in  $O^*(n^{2.055})$  time. 

# Sketch of Randomized Algorithm

In the next slide, we will present the pseudo code of the randomized algorithm. Here we give some intuition of such algorithm.

The solution  $x^*$  of linear programming problem can be regarded as the probability that each events will happen. Therefore, the algorithm tries to randomly choose the nodes recursively based on the solution  $x^*$  from linear programming.

The LP polytope is lifted to the  $O(sh^2)$ -th level so as to repeatedly condition<sup>6</sup> on events throughout the algorithm.

---

<sup>6</sup>Recall Definition 4

# Rounding a Lifted Fractional Solution

---

```
1: procedure SOLVE( $u, L', x$ )
2:    $\tilde{V} \leftarrow \tilde{V} \cup \{u\}$ .
3:   if  $u \in V^{\text{leaf}}$  then return
4:   end if
5:   let  $S_v \leftarrow \emptyset$  for every  $v \in \Lambda_u$ 
6:   for every  $\ell \in L'$  do
7:     randomly choose a child  $v$  of  $u$  with probability  $x_{(v,\ell)}$ 
8:      $S_v \leftarrow S_v \cup \{\ell\}$ 
9:      $x \leftarrow x$  conditioned on the event  $(v, \ell)$ 
10:  end for
11:  for every  $v \in \Lambda_u$ , with probability  $x_v$  do
12:    SOLVE( $v, S_v \cup \text{dem}(v), x$  conditioned on event  $v$ )
13:  end for
14: end procedure
```

---

# Rounding a Lifted Fractional Solution

$\tilde{T}$  can be induced by  $\tilde{V}$  through the following algorithm. Initially,  $\tilde{V} \leftarrow \emptyset$ . Then we call  $\text{SOLVE}(\text{root}(T^0), \text{dem}(\text{root}(T^0)), x^*)$ .

Note that although we had the constraint that  $x_{(\text{root}(T^0), \ell)} = 1$  for  $\ell \in K$ , the solution is fractional and thus there might not be a leaf that "fully" supply the label. Regardless, as claimed before, the probability that a local label will be included by the conditioning procedure is high.

# Analyzing the Algorithm

We shall show some properties first to make sure the algorithm is well-defined.

## Lemma 19

*For every recursion of SOLVE that the algorithm invokes,*

- a) at the beginning of the recursion, we have  $x_u = 1$  and  $x_{(u,\ell)} = 1$  for all  $\ell \in L'$ , and*
- b) the random sampling process is well-defined, that is,  $\sum_{v \in \Lambda_u} x_{(v,\ell)} = 1$  before each step.*

# Analyzing the Algorithm

We prove the lemma recursively.

- $u = \text{root}(T^0)$  follows by the definition of  $x^*$ .
- If (a). holds for some  $u \notin V^{\text{leaf}}$ , then (b). also holds as
$$x_{(u,\ell)} = \sum_{v \in \Lambda_u} x_{(v,\ell)}$$
- If (b). holds for some  $u \notin V^{\text{leaf}}$ . After finishing Loop 6,  $x_{(v,\ell)} = 1, \forall v \in \Lambda_u, \ell \in S_v$ . Let  $x'$  be the polytope passed in sub-recursion at Line 12. We have  $x'v = x'_{(v,\ell)} = 1, \forall \ell \in S_v$ . Also,  $x'_{(v,\ell)} = x'_v = 1, \forall \ell \in \text{dem}(v)$  follows by definition. Therefore, (a). holds for every child  $v \in \Lambda_u$ .

# Analyzing the Algorithm

Next, we shall show that  $\tilde{T}$  is indeed label-consistent.

- $\text{dem}(u) \subseteq L$  by Line 12 and the first call  $\text{SOLVE}(\text{root}(T^0), \text{dem}(\text{root}(T^0)), x^*)$ .
- By Loop 6, each label  $\ell \in \text{dem}(u)$  will be passed down to some leaf node  $v \in \Lambda_u^{\text{leaf}}$ . By 19 (a). we have  $x_{(v,\ell)} = 1$  at the beginning of the recursion. Hence  $\ell = a_v$ .

We define some notations for the convenience of the following discussion.

- Let  $x^{(u,i)}$  be the value of  $x$  after the  $i$ -th iteration of Loop 6 in  $\text{SOLVE}(u, \cdot, \cdot)$ , or the value at the end of loop if it terminates in less than  $i$  iterations.



# Marginal Probabilities

The following lemmas state that the marginal probabilities of events are maintained in the random process.

## Lemma 20

Let  $u \in V(T^0)$ ,  $i \in [sh]$ ,  $x^{\text{old}} = x^{(u,i-1)}$ ,  $x^{\text{new}} = x^{(u,i)}$ . Let  $\mathcal{E}$  be any event determined by the random numbers generated before  $x^{\text{old}}$ . Then, for every  $e \in \mathbb{D}$ , we have

$$\mathbb{E} \left[ x_e^{\text{new}} \mid x_e^{\text{old}}, \mathcal{E} \right] = x_e^{\text{old}}.$$

## Lemma 21

Let  $u \in V^{\text{int}}$ ,  $v \in \Lambda_u$ ,  $x^{\text{old}} = x^{(u,sh)}$ ,  $x^{\text{new}} = x^{(v,0)}$ . Let  $\mathcal{E}$  be any event determined by the random numbers generated before  $x^{\text{old}}$ . Then, for every  $e = v \in V(T^0[u])$  or  $e = (v, \ell)$  for some  $v \in V(T^0[u])$ , we have

$$\mathbb{E} \left[ x_e^{\text{new}} \mid x_e^{\text{old}}, \mathcal{E} \right] = x_e^{\text{old}}.$$

# Marginal Probabilities

The aforementioned lemmas can be proved by the definition of expectation and conditioning in the Sherali-Adams Hierarchy. With these lemmas, we could derive the following corollaries.

## Corollary 1

For every  $v \in V(T^0)$ , we have  $\mathbb{P}[v \in \tilde{V}] = x_v^*$ .

## Corollary 2

$\mathbb{E}[\text{cost}(\tilde{T})] \leq \text{opt}.$

# Bounding Probabilities

To finish the proof, it suffices to show that each label is provided by  $\tilde{T}$  with high probability.

Let  $t_\ell$  be the number of nodes  $v \in \tilde{V} \cap V^{\text{leaf}}$  with  $a_v = \ell$ , then

## Lemma 22

$$\mathbb{E}[t_\ell] = 1.$$

It is easy to prove. By Corollary 1 we have

$$\mathbb{E}[t_\ell] = \sum_{v \in V^{\text{leaf}}: a_v = \ell} \mathbb{P}[v \in \tilde{V}] = \sum_{v \in V^{\text{leaf}}: a_v = \ell} x_v^* = x_{(\text{root}(T^0), \ell)}^* = 1.$$

## Lemma 23

For every  $w \in V^{\text{leaf}}$  with  $a_w = \ell$ , we have  $\mathbb{E} [t_\ell \mid w \in \tilde{V}] \leq h + 1$ .

We give an approach to prove it. Let  $h'$  be the depth of  $w$  in  $T^0$ .

- For  $i = 0, 1, \dots, h' - 1$ , Define  $U_i := \{w' \in V^{\text{leaf}} \setminus \{w\} \mid a_{w'} = \ell, \text{ the depth of LCA of } w, w' \text{ is } i\}$
- If we could prove  $\mathbb{E} [ |U_i \cap \tilde{V}| \mid w \in \tilde{V} ] \leq 1$ , then the result follows by summing up the inequality over all  $i = 0, 1, 2, \dots, h' - 1$ .

# Bounding Probabilities

Now we shall prove  $\mathbb{E} \left[ |U_i \cap \tilde{V}| \mid w \in \tilde{V} \right] \leq 1$  for all  $0 \leq i < h'$ . Let  $u_i$  be the ancestor of  $w$  with depth  $i$ , and  $(S_v)_{v \in \Lambda_u}$  be the vector before Loop 11 in  $\text{SOLVE}(u, \cdot, \cdot)$ . The following equation holds.

$$\begin{aligned} \mathbb{P} \left[ w' \in \tilde{V} \mid (S_v)_{v \in \Lambda_u, X^{(u,sh)}, w' \in \tilde{V}} \right] &= \mathbb{P} \left[ w' \in \tilde{V} \mid (S_v)_{v \in \Lambda_u, X^{(u,sh)}} \right] \\ &= \mathbb{E} \left[ X_{w'}^{(w',0)} \mid (S_v)_{v \in \Lambda_u, X^{(u,sh)}} \right] = X_{w'}^{(u,sh)} \end{aligned}$$

The last equality follows from Lemma 20 and Lemma 21.

Summing up all  $w' \in U_i$  we have

$$\mathbb{E} \left[ |U_i \cap \tilde{V}| \right] = \sum_{w' \in U_i} X_{w'}^{(u,sh)} = \sum_{w' \in U_i} X_{(w',\ell)}^{(u,sh)} \leq X_{(u,\ell)}^{(u,sh)} \leq 1.$$

Lastly, we will state the following lemma.

## Lemma 24

*For every  $\ell \in K$ , we have  $\mathbb{E}[t_\ell \mid t_\ell \geq 1] \leq h + 1$ .*

With the lemma above, we could simply derive that

$$1 = \mathbb{E}[t_\ell] = \mathbb{E}[t_\ell \mid t_\ell \geq 1] \cdot \mathbb{P}[t_\ell \geq 1] \Rightarrow \mathbb{P}[t_\ell \geq 1] \geq \frac{1}{h + 1}.$$

This finishes the proof of Theorem 18.

# Bounding Probabilities

Finally, we give a derivation of Lemma 24. In the following,  $w, w'$  in summations are over all nodes in  $V^{\text{leaf}}$  with label  $\ell$ .

$$\begin{aligned}\mathbb{E}[t_\ell \mid t_\ell \geq 1]^2 &\leq \mathbb{E}[t_\ell^2 \mid t_\ell \geq 1] = \sum_{w, w'} \mathbb{P}[w, w' \in \tilde{V} \mid t_\ell \geq 1] \\ &= \sum_w \left( \mathbb{P}[w \in \tilde{V} \mid t_\ell \geq 1] \sum_{w'} \mathbb{P}[w' \in \tilde{V} \mid w \in \tilde{V}, t_\ell \geq 1] \right) \\ &= \sum_w \mathbb{P}[w \in \tilde{V} \mid t_\ell \geq 1] \mathbb{E}[t_\ell \mid w \in \tilde{V}] \\ &\leq (h+1) \sum_w \mathbb{P}[w \in \tilde{V} \mid t_\ell \geq 1] \\ &= (h+1) \mathbb{E}[t_\ell \mid t_\ell \geq 1] \Rightarrow \mathbb{E}[t_\ell \mid t_\ell \geq 1] \leq h+1.\end{aligned}$$

- 1 Introduction
- 2 Previous & Related Works
- 3 Decomposition Tree
- 4 Label-Consistent Subtree
- 5 Approximation Algorithm for Label-Consistent Subtree
- 6 LP Relaxation
- 7 Hardness Results**



# Hardness Results

## Definition 25

Quasi-polynomial-time algorithms run in  $O(2^{\text{polylog}(n)})$  time. More precisely,  $\text{QP} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{O(\log^c n)})$ .

## Definition 26

$\text{ZPTIME}(f(n))$  refers to randomized algorithms that always return the correct answer and have randomized running times with expectation  $O(f(n))$ .

## Remark

*It will be shown that, assuming the Projection Game Conjecture (which will be stated later) and  $\text{NP} \not\subseteq \bigcap_{0 < \epsilon < 1} \text{ZPTIME}(2^{n^\epsilon})$ , the optimal approximation ratio for quasi-polynomial time algorithms is  $O(\log^2 k / \log \log k)$ .*

## Remark

*[HK03] shows that, by assuming  $\text{NP} \not\subseteq \text{ZPTIME}(n^{\text{polylog}(n)})$ , an approximation ratio  $O(\log^{2-\epsilon} k)$  is infeasible for (quasi-)polynomial time algorithms.*

## Definition 27

Projection Game Let  $G = (U, W; E)$  be a bipartite graph and  $\Sigma$  a set of labels. We associate each edge  $(u, w) \in E$  with a projection  $\pi_{uw} : \Sigma \rightarrow \Sigma$ . We are to find an labeling  $f : (U \cup W) \rightarrow \Sigma$  that associates each vertex to a label. A labeling  $f$  is said to *cover* an edge  $(u, w)$  if  $\pi_{uw}(f(u)) = f(w)$ . Our goal is then to find a labeling that covers the most edges.

## Definition 28

Group Steiner Tree (GST) Given a (weighted) undirected graph  $G$  with  $n$  vertices, a root vertex  $r \in V(G)$ , and some vertex subsets  $S_1, \dots, S_k \subseteq V(G)$  referred to as *groups*, the goal of GST is to find a minimal cost subgraph that contains a path from the root to at least a vertex in each group.

## Remark

*GST can be reduced to DST by first making the edges bi-directed, then add, for each group  $S_i$ , a terminal  $t_i$ , together with zero-cost edges from each vertex in  $S_i$  to  $t_i$ .*

# (Randomized) Exponential Time Hypothesis

## Definition 29

(Randomized) Exponential Time Hypothesis There exists some  $\delta > 0$  such that 3-SAT cannot be solved in (randomized)  $O(2^{\delta n})$ .

## Remark

*The paper uses the slightly weaker formulation  $\text{NP} \not\subseteq \bigcap_{0 < \epsilon < 1} \text{ZPTIME}(2^{n^\epsilon})$ , though we still assume ETH for simplicity. (The general idea is the same.)*

## Definition 30

**Projection Game Conjecture** There exists a  $c > 0$  such that, for every  $0 < \epsilon \leq c$ , any SAT instance  $\phi$  of size  $n$  can be efficiently reduced to a Projection Game on a  $(n^{\text{poly}(\epsilon)}\text{-regular bipartite})$  graph with  $n^{1+o(n)}$  vertices with  $|\Sigma| = O(n^{\text{poly}(\epsilon)})$ . Furthermore, the game satisfies the following.

- If  $\phi$  is satisfiable, then there is a labeling covering every edge.
- Otherwise, there exists no labeling covering more than a  $n^{-\epsilon}$  fraction of the edges.

By comparing the coefficients, we can derive

## Corollary 31

*Assuming the previous hypotheses, for any  $0 \leq \epsilon < c$ , there is no  $O(2^{n^\epsilon})$ -time algorithm that outputs an  $n^\epsilon$ -approximation of the Projection Game.*



## Theorem 32

Consider an instance  $\psi$  of the Projection Game on a  $\Delta$ -regular  $n$ -vertex bipartite graph  $G$  with the label set  $\Sigma$  of size  $\sigma$ . For any parameter  $1 \leq h \leq O(\log^2 n)$ , there is a randomized reduction from  $\psi$  to a GST on a tree  $T$  with  $k$  groups such that  $|V(T)| = (\sigma n)^h$  and  $k = \Delta^h$  satisfying the following with high probability.

- If there is a labeling covering all edges of  $G$ , then there is a feasible solution to the GST with cost  $h^2$ .
- If there is no labeling covering  $\gamma$  fraction of edges, there is no feasible solution to the GST with cost  $< \min(\gamma^{-\frac{1}{2}} h, \Omega(h \log k))$ .

The proof can be found in [HK03].

By choosing the parameters carefully, we obtain

## Theorem 33

*Assuming Corollary 31, there exists no quasi-polynomial time algorithm that approximates GST (and therefore DST) to a ratio of  $o(\log^2 k / \log \log k)$  or  $o(\log^2 N / \log \log N)$ .*



Moses Charikar et al. “Approximation Algorithms for Directed Steiner Problems”. In: *Journal of Algorithms* 33.1 (1999), pp. 73–91. ISSN: 0196-6774. DOI: <https://doi.org/10.1006/jagm.1999.1042>. URL: <https://www.sciencedirect.com/science/article/pii/S0196677499910428>.



Joseph Cheriyan et al. “Approximating Rooted Steiner Networks”. In: *ACM Trans. Algorithms* 11.2 (Oct. 2014). ISSN: 1549-6325. DOI: [10.1145/2650183](https://doi.org/10.1145/2650183). URL: <https://doi.org/10.1145/2650183>.



Chandra Chekuri and M. Pal. “A recursive greedy algorithm for walks in directed graphs”. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 2005, pp. 245–253. DOI: [10.1109/SFCS.2005.9](https://doi.org/10.1109/SFCS.2005.9).



Anupam Gupta, Ravishankar Krishnaswamy, and R. Ravi. “Tree Embeddings for Two-Edge-Connected Network Design”. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '10. Austin, Texas: Society for Industrial and Applied Mathematics, 2010, pp. 1521–1538. ISBN: 9780898716986.



Naveen Garg, Goran Konjevod, and R. Ravi. “A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem”. In: *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '98. San Francisco, California, USA: Society for Industrial and Applied Mathematics, 1998, pp. 253–259. ISBN: 0898714109.



Fabrizio Grandoni, Bundit Laekhanukit, and Shi Li.  *$O(\log^2 k / \log \log k)$ -Approximation Algorithm for Directed Steiner Tree: A Tight Quasi-Polynomial-Time Algorithm*. 2018. arXiv: 1811.03020 [cs.DS].



Eran Halperin and Robert Krauthgamer. “Polylogarithmic Inapproximability”. In: *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '03. San Diego, CA, USA: Association for Computing Machinery, 2003, pp. 585–594. ISBN: 1581136749. DOI: 10.1145/780542.780628. URL: <https://doi.org/10.1145/780542.780628>.



Shunhua Jiang et al. *Faster Dynamic Matrix Inverse for Faster LPs*. 2020. arXiv: 2004.07470 [cs.DS].



Bundit Laekhanukit. “Parameters of Two-Prover-One-Round Game and the Hardness of Connectivity Problems”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '14. Portland, Oregon: Society for Industrial and Applied Mathematics, 2014, pp. 1626–1643. ISBN: 9781611973389.



Thomas Rothvoß. *Directed Steiner Tree and the Lasserre Hierarchy*. 2012. arXiv: 1111.5473 [cs.DS].



Hanif Sherali and Warren Adams. “A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems”. In: *SIAM J. Discrete Math.* 3 (May 1990), pp. 411–430. DOI: 10.1137/0403036.