Query Complexity of the Metric Steiner Tree Problem*

Yu Chen† Sanjeev Khanna‡ Zihan Tan§

Abstract

In the metric Steiner Tree problem, we are given an $n \times n$ metric w on a set V of vertices along with a set $T \subseteq V$ of k terminals, and the goal is to find a tree of minimum cost that contains all terminals in T. This is a well-known NP-hard problem and much of the previous work has focused on understanding its polynomial-time approximability. In this work, we initiate a study of the query complexity of the metric Steiner Tree problem. Specifically, if we desire an α-approximate estimate of the metric Steiner Tree cost, how many entries need to be queried in the metric w? For the related minimum spanning tree (MST) problem, this question is well-understood. For any fixed $\varepsilon > 0$, one can estimate the MST cost to within a $(1+\varepsilon)$-factor using only $\tilde{O}(n)$ queries, and this is known to be essentially tight. Can one obtain a similar result for Steiner Tree cost? Note that a $(2+\varepsilon)$-approximate estimate of Steiner Tree cost can be obtained with $\tilde{O}(k)$ queries by simply applying the MST cost estimation algorithm on the metric induced by the terminals.

Our first result shows that the Steiner Tree problem behaves in a fundamentally different manner from MST: any (randomized) algorithm that estimates the Steiner Tree cost to within a $(5/3-\varepsilon)$-factor requires $\Omega(n^2)$ queries, even if k is a constant. This lower bound is in sharp contrast to an upper bound of $O(nk)$ queries for computing a $(5/3)$-approximate Steiner Tree, which follows from previous work by Du and Zelikovsky.

Our second main result, and the main technical contribution of this work, is a *sublinear* query algorithm for estimating the Steiner Tree cost to within a *strictly better-than*-2 factor. We give an algorithm that achieves this goal, with a query complexity of $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$; since $k \leq n$, the algorithm performs at most $\tilde{O}(n^{13/7}) = o(n^2)$ queries in the worst-case. Our estimation algorithm reduces this task to that of designing a sublinear query algorithm for a suitable set cover problem. We complement this result by showing an $\tilde{\Omega}(n + k^{6/5})$ query lower bound for any algorithm that estimates Steiner Tree cost to a strictly better than 2 factor. Thus $\tilde{\Omega}(n^{6/5})$ queries are needed to just beat 2-approximation when $k = \Omega(n)$; a sharp contrast to MST cost estimation where a $(1+o(1))$-approximate estimate of cost is achievable with only $\tilde{O}(n)$ queries.

1 Introduction

In the Steiner Tree problem, we are given a weighted (undirected) graph G and a subset T of vertices in G called *terminals*, and the goal is to compute a minimum weight connected subgraph of G (a Steiner Tree) that spans all terminals in T. This is one of the most fundamental NP-hard problems [16], and has been studied extensively over the past several decades from the perspective of approximation algorithms [17, 27, 13, 18, 20, 15, 14] (see also [21] for a compendium of its variants). The current best known approximation ratio is $\ln 4 + \varepsilon < 1.39$ achieved by [15] (see also [14, 23]), and it has been shown [22] that approximating to within a factor better than 96/95 is NP-hard.

In this paper, we study the query complexity of the Steiner Tree problem. In particular, we consider an equivalent variant called the *metric Steiner Tree* problem, where the input consists of a *metric* w on a set V of n points (equivalently, a weighted complete graph on V) and a subset $T \subseteq V$ of k points called *terminals*. We are allowed to perform weight queries between vertices in V[1], and the goal is to design an algorithm for either computing a Steiner Tree with minimum cost or estimating the cost of an optimal Steiner Tree, using as few queries as possible.

*The full version of the paper can be accessed at https://arxiv.org/abs/2211.03893

†EPFL, Lausanne, Switzerland. Email: yu.chen@epfl.ch. Supported by ERC Starting Grant 759471. Work done while the author was a graduate student at University of Pennsylvania.

‡University of Pennsylvania, Philadelphia, PA, USA. Email: sanjeev@cis.upenn.edu. Supported in part by NSF awards CCF-1763514, CCF-1934876, and CCF-2008305.

§Rutgers University, NJ, USA. Email: zihantan1993@gmail.com. Supported by a grant to DIMACS from the Simons Foundation (820931). Work done while the author was a graduate student at University of Chicago.

[1]This is also known as the Dense Graph Model [8]. In the other model, the Bounded-Degree Graph Model [10] (where the max-degree of the input graph is bounded by d), it is easy to show that estimating the minimum Steiner Tree cost, or even the minimum spanning tree cost, in an n-vertex graph within any non-trivial factor requires $\Omega(nd)$ queries.

It is well-known that a minimum weight spanning tree of the metric induced by the terminals gives a 2-approximate Steiner Tree [17]. Moreover, the metric Minimum Spanning Tree (MST) cost can be estimated to within factor $(1+\varepsilon)$ be performing $\tilde{O}(n/\varepsilon^{O(1)})$ queries [25, 24]. Therefore, the minimum metric Steiner Tree cost can be estimated within factor $(2+\varepsilon)$ with only $\tilde{O}(k/\varepsilon^{O(1)})$ queries. However, the query complexity of obtaining a better-than-2 estimation of the minimum metric Steiner Tree cost remains wide open, and so is the query complexity of computing such a Steiner Tree. We note that this is the interesting regime of the metric Steiner Tree problem: approximating or estimating the cost to a factor better than 2 crucially requires some knowledge of the metric incident on Steiner nodes.

1.1 Our Results In this paper, we provide a comprehensive understanding on the trade-off between approximation ratio and query complexity of the metric Steiner Tree problem.

Our first result establishes a separation between the behavior the Steiner Tree problem and the MST problem. Specifically, we show that for any $\varepsilon > 0$, any randomized algorithm to estimate Steiner Tree cost to within a $(5/3-\varepsilon)$-factor requires $\Omega(n^2)$ queries even if k is a constant. Together with an upper bound of $O(nk)$ queries for computing a (5/3)-approximate Steiner Tree (which follows from [28] and [27][2]), this result shows a phase transition in the query complexity at (5/3)-approximation. This is in contrast to the MST cost estimation problem where for any $\varepsilon > 0$, $\tilde{O}(n)$ queries suffice to estimate MST cost to within a factor of $(1+\varepsilon)$.

THEOREM 1.1. *For any constant $0 < \varepsilon < 2/3$, any randomized algorithm that with high probability estimates the metric Steiner Tree cost to within a factor of $(5/3-\varepsilon)$ performs $\Omega(n^2/4^{(1/\varepsilon)})$ queries in the worst case, even when k is a constant.*

Our proof of this result is based on constructing a pair of distributions on Steiner Tree instances whose costs differ by a $(5/3-\varepsilon)$ factor, and yet whose metrics differ in $O_\varepsilon(1)$ entries, leading to an $\Omega(n^2)$ lower bound for any fixed $\varepsilon > 0$.

We complement the above result by showing that even if we weaken the goal to simply computing a slightly better-than-2 approximate Steiner Tree, the query complexity remains $\Omega(nk)$.

THEOREM 1.2. *For any constant $0 < \varepsilon < 1/3$, any randomized algorithm that outputs a $(2-\varepsilon)$-approximate Steiner Tree performs at least $\Omega(nk)$ queries in the worst case.*

Our second set of results is concerned with understanding the query complexity of obtaining a strictly better-than-2 estimate of the Steiner Tree cost. The main technical contribution of this paper is a sublinear-query algorithm that obtains a strictly better-than-2 estimate of the cost, by performing $\tilde{O}(n^{12/7}+n^{6/7}\cdot k)$ queries (as $k \le n$, the query complexity is $\tilde{O}(n^{13/7}) = o(n^2)$).

THEOREM 1.3. *There is an efficient randomized algorithm that with high probability estimates the metric Steiner Tree cost to within a factor of $(2-\varepsilon_0)$ for some universal constant $\varepsilon_0 > 0$, by performing $\tilde{O}(n^{12/7}+n^{6/7}\cdot k)$ queries.*

At a high-level, the proof of the above theorem starts with a minimum spanning tree $\mathcal{T}$ of the graph induced by terminals. Even on simple metrics such as a metric where all weights are 1 or 2, the cost of such a tree can be up to a factor 2 away from the optimal Steiner Tree cost. But in this case, the optimal tree necessarily improves upon $\mathcal{T}$ by using Steiner nodes to efficiently connect together many terminals. The first challenge then becomes if such opportunities can be identified only by local exploration of the metric. Our main insight is that this task can be cast as a suitable *set cover* problem where the objective is to estimate the universe size minus the optimal set cover size. We then design a sublinear query algorithm for estimating the value of this set cover objective, and use it to determine whether or not the optimal Steiner tree cost is close to the cost of $\mathcal{T}$, or bounded away from it. The second challenge in obtaining this result is that explicit computation of the MST $\mathcal{T}$ in the graph induced by the terminals requires $O(k^2)$ queries which rules out a sublinear query complexity when $k = \Omega(n)$. To get around this, we design an outer algorithm that efficiently simulates access to $\mathcal{T}$ without ever explicitly computing it. The composed algorithm, achieves a strictly better-than-2 estimate of the Steiner Tree cost in $\tilde{O}(n^{12/7}+n^{6/7}\cdot k)$ queries.

[2] See Appendix A for an explanation.

The result above raises a natural question: can the task of obtaining a strictly better-than-2 estimate of the Steiner Tree cost be achieved with only $\tilde{O}(n)$ queries? Our next result rules out this possibility at least when k is sufficiently large. We show that any algorithm that estimates the Steiner Tree cost to a factor strictly better than 2, necessarily requires $\tilde{\Omega}(n + k^{6/5})$ queries.

THEOREM 1.4. *For any constant $0 < \varepsilon < 1/3$, any randomized algorithm that with high probability estimates the metric Steiner Tree cost to within a factor of $(2 - \varepsilon)$ performs at least $\tilde{\Omega}(n + k^{6/5})$ queries in the worst case.*

Our third and final set of results is concerned with understanding the query complexity of computing an α-approximate Steiner Tree for any $\alpha \geq 2$. We show that $\tilde{\Theta}(k^2/\alpha)$ queries are both sufficient and necessary for this task.

THEOREM 1.5. *Let $\alpha \geq 2$ be any constant. Then*

- *there exists an efficient randomized algorithm that with high probability computes an α-approximate Steiner Tree, by performing $\tilde{O}(k^2/\alpha)$ queries; and*
- *any randomized algorithm that outputs an α-approximate Steiner Tree performs at least $\Omega(k^2/\alpha)$ queries in the worst case.*

Our results on the tradeoff between query complexity and approximation quality are summarized in Figure 1. Together, they illustrate several interesting phase transitions in the query complexity of approximating metric Steiner Tree. The query complexity remains $\Theta(n^2)$ up to an approximation factor of $5/3$ even if k is a constant and the goal is to only estimate the Steiner Tree cost. Then at $(5/3)$-approximation, it drops to $\Theta(nk)$, and it is possible to also find a $(5/3)$-approximate Steiner Tree with $\Theta(nk)$ queries. Next if the goal is to find an α-approximate Steiner Tree, the query complexity remains $\Theta(nk)$ even as α approaches 2. At this point, another phase transition occurs: for any $\alpha \geq 2$, the query complexity of finding an α-approximate Steiner Tree becomes $\tilde{\Theta}(k^2/\alpha)$. For α-approximate estimation of the cost for any $\alpha < 2$, we show that $\tilde{\Omega}(n^{6/5})$ queries are necessary when $k = \Omega(n)$. On the other hand, we give a $(2 - \varepsilon_0)$-estimation algorithm that uses only $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$ queries, for some universal $\varepsilon_0 > 0$.

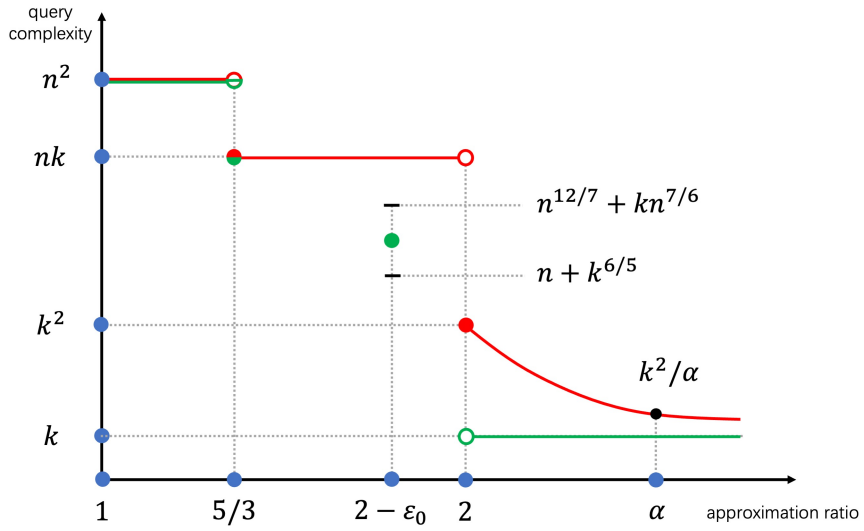


Figure 1: An illustration of the trade-off between query complexity and approximation ratio for the metric Steiner Tree problem. The red curve shows the complexity of computing a Steiner Tree, while the green curve shows the complexity of estimating the minimum metric Steiner Tree cost. The upper bound at $(5/3)$-approximation follows from [28] and [27]; the bottom green curve (showing $\Theta(k)$ query complexity for α-estimating the cost where $\alpha > 2$) is due to [24], and all other curves are results of this paper. All terms are inside a $\tilde{O}(\cdot)$ symbol.

Organization. We start with some preliminaries in Section 2. We then provide the proof of $(5/3-\varepsilon)$-approximation lower bound (Theorem 1.1) in Section 3. We present the $(2-\varepsilon_0)$-approximation algorithm for Theorem 1.3 in Section 4. The proofs of the lower bounds in Theorem 1.2, Theorem 1.4 and Theorem 1.5 are provided in Section 5, Section 7 and Section 6, respectively.

2 Preliminaries

Let G be a graph and let S be a subset of its vertices. We denote by $G[S]$ the subgraph of G induced by S. For two (not necessarily disjoint) subsets A, B of vertices of G, we denote by $E_G(A,B)$ the set of all edges with one endpoint in A and the other endpoint in B, and we denote by $E_G(A)$ the set of all edges with both endpoints in A. For a vertex $v \in V(G)$, we denote by $\deg_G(v)$ the degree of v in G. We sometimes omit the subscript G in the above notations if it is clear from the context.

Let $G=(V,E,w)$ be a weighted graph where $w : E(G) \to \mathbb{R}^+$ is a weight function on edges in G. For a subgraph $H \subseteq G$, we define $w(H) = \sum_{e\in E(H)} w(e)$. For a pair u, u' of vertices in G, we denote by $\mathrm{dist}_G(u,u')$ the shortest-path distance in G between u and u'. In this paper, we will often consider the case where G is a complete graph and w satisfies the triangle inequality. That is, for all $u, u', u'' \in V$, $w(u,u') \le w(u,u'') + w(u',u'')$. In this case, w can be also viewed as a metric on the set V of points. For a vertex v and a set $U \subseteq V$, we denote $w(v,U) = \min\{w(v,u) \mid u \in U\}$. For a pair $U, U' \subseteq V$ of sets, we denote $w(U,U') = \min\{w(u,u') \mid u \in U, u' \in U'\}$.

Let $\mathcal{T}$ be a tree rooted at a vertex $r \in V(\mathcal{T})$. Let u be a vertex in $\mathcal{T}$. The *height* of u in $\mathcal{T}$ is defined to be the minimum hop-distance between u and any leaf in the subtree of $\mathcal{T}$ rooted at u. For example, the height of a leaf is 0, and the height of a parent of a leaf is 1, etc.

For a weighted graph $G=(V,E,w)$ and a subset T of vertices in G, we denote by (G,T) the instance of the Steiner Tree problem where G is the graph and T is the set of vertices to be connected. We denote the optimal cost of a solution to this instance by $\mathsf{ST}(G,T)$. When G is a complete graph and w is a metric on V, an instance of the Steiner Tree problem is also denoted by (V,T,w), and the optimal cost of a solution is also denoted by $\mathsf{ST}(V,T,w)$. Vertices in T are called *terminals*, and vertices in $V \setminus T$ are called *Steiner vertices*.

Throughout the paper, we will refer to the algorithms that compute an α-approximate Steiner Tree of cost bounded by α-*approximations*, and will refer to the algorithms that estimate the metric Steiner Tree cost to within factor α by α-*estimations*.

We use the following standard version of Chernoff Bound (see. e.g., [5]).

LEMMA 2.1. (CHERNOFF BOUND) *Let $X_1,\ldots,X_n$ be independent randon variables taking values in $\{0,1\}$. Let $X=\sum_{1\le i\le n} X_i$, and let $\mu=\mathbb{E}[X]$. Then for any $t > 2e\mu$,*

$$\Pr\left[X > t\right] \le 2^{-t}.$$

Additionally, for any $0 \le \delta \le 1$,

$$\Pr\left[X < (1-\delta)\cdot\mu\right] \le e^{-\frac{\delta^2\cdot\mu}{2}}.$$

3 An $\Omega(n^2)$ Lower Bound for $(5/3-\varepsilon)$-Estimation

In this section we provide the proof of Theorem 1.1. Specifically, for any constant $0 < \varepsilon < 2/3$, we will construct a pair $I_{\mathsf{Y}} = (V,T,w_{\mathsf{Y}}), I_{\mathsf{N}} = (V,T,w_{\mathsf{N}})$ of instances of the metric Steiner Tree problem where $|T| = O(2^{(1/\varepsilon)})$, such that the metric Steiner Tree costs of instance I_{Y} and instance I_{N} differ by factor $(5/3-\varepsilon)$, while the metrics w_{Y} and w_{N} differ at only $O_\varepsilon(1)$ places. We will then construct two distributions $\mathcal{D}_{\mathsf{Y}}, \mathcal{D}_{\mathsf{N}}$ of instances by randomly naming the vertices in the instances $I_{\mathsf{Y}}, I_{\mathsf{N}}$ respectively; and show that any algorithm that with probability at least 0.51 distinguishes between instances $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$ (and in particular between the distributions on metrics w_{Y} and w_{N}) has to perform at least $\Omega(n^2/4^{(1/\varepsilon)})$ queries. We remark that it is easy to construct (by adding dummy terminals to instances I_{Y} and I_{N}), for every $k \ge \Omega(2^{1/\varepsilon})$, such a pair of instances $I_{\mathsf{Y}}^{(k)}$ and $I_{\mathsf{N}}^{(k)}$ with k terminals each. Our constructions are similar to the examples used in [26] to determine the worst-case k-Steiner ratios.

We start by giving a high-level overview of the construction of instances (V,T,w_{Y}) and (V,T,w_{N}). We would like to ensure that (i) metrics w_{Y} and w_{N} differ only in $O_\varepsilon(1)$ places; and (ii) $\mathsf{ST}(V,T,w_{\mathsf{Y}})$ and $\mathsf{ST}(V,T,w_{\mathsf{N}})$ differ by a factor of (5/3) roughly. In order to achieve property (i), for every pair (u,u') of vertices in V such that at least

one of u, u' is a terminal, $w_{\mathsf{Y}}(u,u') = w_{\mathsf{N}}(u,u')$ must hold, since otherwise simply querying all terminal-involved distances will distinguish between metrics w_{Y} and w_{N} by performing $O(nk)$ queries. However, once we are given that w_{Y} and w_{N} are identical on all terminal-involved pairs, the previous results in [27] and [28] imply that the values $\mathsf{ST}(V,T,w_{\mathsf{Y}})$ and $\mathsf{ST}(V,T,w_{\mathsf{N}})$ differ by a factor of at most $(5/3)$. Therefore, we have to design metrics w_{Y} and w_{N} such that the analysis from [27] and [28] is nearly tight. It turns out that the tree $\mathcal{T}_Y$ has to be quite balanced and symmetric.

We now describe the construction of instances (V,T,w_{Y}) and (V,T,w_{N}) in detail. Let $d = \lceil 1/\varepsilon \rceil$. We first define an auxiliary weighted tree ρ as follows. The tree ρ is a complete binary tree of depth d, so $|V(\rho)| = 2^{d+1}-1$. Let r be the root of ρ. For each node $v \in V(\rho)$, we say that v is *at level* i (or the level of v is i), iff the unique path in ρ that connects v to r contains i edges. Clearly, all leaves of ρ are at level d. For an edge $(u,u') \in E(\rho)$ where u' is the parent of u, we say that (u,u') is a *level-i edge* iff u is at level i. We now define the weights on edges in $E(\rho)$: all level-d edges have weight 1, and for each $1 \le i \le d-1$, all level-i edges have weight $2^{(d-1)-i}$. We denote by $L(\rho)$ the set of all leaves of ρ.

In order to avoid ambiguity, we refer to vertices of ρ as nodes and points in V as vertices. The vertex set V is partitioned into $(2^{d+1}-1)$ subsets: $V = \bigcup_{x \in V(\rho)} V_x$, where each subset is indexed by a node in ρ. For each leaf node x in $L(\rho)$, the set V_x contains a single vertex, that we denote by u_x. For each non-leaf node x in ρ, the set V_x contains either $\left\lfloor \frac{n-2^d}{2^d-1} \right\rfloor$ or $\left\lceil \frac{n-2^d}{2^d-1} \right\rceil$ vertices (so that the total number of vertices in V is n), with one of them designated as the *special vertex* in V_x, denoted by u_x, and all other vertices are called *regular vertices*. For consistency, for each leaf node $x \in L(\rho)$, we also call the only vertex u_x in V_x a special vertex. The terminal set is defined to be $T = \{u_x \mid x \in L(\rho)\}$, so $|T| = 2^d$. We denote by S the set of all special vertices, so $T \subseteq S$ and $|S| = 2^{d+1}-1$.

We now define metrics w_{Y} and w_{N} as follows. We first define w_{N}. For every pair v, v' of vertices in V, assume $v \in V_x$ and $v' \in V_{x'}$; denote by $\hat{\ell}$ the level of the lowest common ancestor of nodes x and x' in ρ; and assume without loss of generality that the level of x is at least the level of x'. Now let $\tilde{x}$ be any leaf of ρ that lies in the subtree of ρ rooted at x; then $w_{\mathsf{N}}(v,v') = \mathrm{dist}_\rho(\tilde{x},x) + \mathrm{dist}_\rho(\tilde{x},x')$ (note that this is well-defined since any such leaf $\tilde{x}$ will give the same value of $\mathrm{dist}_\rho(\tilde{x},x) + \mathrm{dist}_\rho(\tilde{x},x')$). We now define w_{Y}. For every pair $v, v' \in V$ such that at least one of v, v' does not lie in S, $w_{\mathsf{Y}}(v,v')$ is defined identically as $w_{\mathsf{N}}(v,v')$. For every pair v, v' of vertices in S with $v \in V_x$ and $v' \in V_{x'}$, the value $w_{\mathsf{Y}}(v,v')$ is defined slightly different as $w_{\mathsf{Y}}(v,v') = \mathrm{dist}_\rho(x,x')$.

We prove the following claim which says on w_{Y} and w_{N} defined above are indeed metrics (that is, they satisfy the triangle inequality). The proof is based on a straightforward case analysis, and is deferred to Appendix B.

Claim 3.1. *$w_{\mathsf{Y}}, w_{\mathsf{N}}$ are metrics on V.*

We next prove the following two claims showing that the minimum Steiner Tree cost of instances (V,T,w_{Y}) and (V,T,w_{N}) differ by a factor of roughly $(5/3)$.

Claim 3.2. *The minimum Steiner Tree cost of instance (V,T,w_{Y}) is at most $(d+1)\cdot 2^{d-1}$.*

Proof. Consider the following Steiner Tree $\mathcal{T}$ whose vertex set is S. Recall that $S = \{u_x \mid x \in V(\rho)\}$. The edge set of $\mathcal{T}$ contains, for each edge $(x,x') \in E(\rho)$, an edge $(u_x, u_{x'})$. From the definition of w_{Y}, it is easy to verify that the tree $\mathcal{T}$ is identical to the tree ρ (together with its edge weights). Therefore,

$$w_{\mathsf{Y}}(\mathcal{T}) = w(\rho) = 2^d + \sum_{1 \le i \le d-1} 2^{(d-1)-i} \cdot 2^i = (d+1)\cdot 2^{d-1}.$$

□

Claim 3.3. *The minimum Steiner Tree cost of instance (V,T,w_{N}) is at least $(5/3)\cdot d \cdot 2^{d-1}$.*

Proof. We start by proving the following observation on an optimal Steiner Tree of instance (V,T,w_{N}).

Observation 3.1. *There exists an optimal Steiner Tree $\mathcal{T}$ of instance (V,T,w_{N}), such that for each node $x \in V(\rho)$, $|V(\mathcal{T}) \cap V_x|$ is either 0 or 1.*

Proof. Let $\mathcal{T}^*$ be an optimal Steiner Tree of instance (V, T, w_{N}), and assume that there exists a node $x \in V(\rho)$, such that $\mathcal{T}^*$ contains at least two distinct vertices of V_x, that we denote by v, v'. Clearly, x is a non-leaf node in ρ and so both v and v' are Steiner vertices in $\mathcal{T}^*$. From the definition of w_{N}, for any vertex $u \in V, u \neq v, v'$, $w_{\mathsf{N}}(u, v) = w_{\mathsf{N}}(u, v')$. Let $\mathcal{T}$ be the tree obtained from $\mathcal{T}^*$ by replacing every v'-incident edge $(u, v') \in E(\mathcal{T}^*)$ with edge (u, v) and then removing any parallel edges. It is easy to verify that $\mathcal{T}$ is also a Steiner Tree of instance (V, T, w_{N}), and that $w(\mathcal{T}) \leq w(\mathcal{T}^*)$. Since $\mathcal{T}^*$ is an optimal Steiner Tree, $\mathcal{T}$ has to be an optimal Steiner Tree as well. We can keep modifying $\mathcal{T}$ in the same way until every group V_x contains at most one vertex in $\mathcal{T}$, while ensuring that the resulting tree $\mathcal{T}$ stays an optimal Steiner Tree of instance (V, T, w_{N}). This completes the proof of the observation. ∎

From Observation 3.1, and since all vertices in the same group behave identically with respect to w_{N}, we conclude that there is an optimal Steiner Tree $\mathcal{T}$ with $V(\mathcal{T}) \subseteq S$. Therefore, we from now on focus on showing that the optimal Steiner Tree of instance (S, T, w_{N}) is at least $(5/3) \cdot d \cdot 2^{d-1}$. Recall that $S = \{u_x \mid x \in V(\rho)\}$. We use the following observation, whose proof is deferred to Appendix C.

OBSERVATION 3.2. *Let $\mathcal{T}$ be an optimal Steiner Tree of the instance (S, T, w_{N}), then every edge of $\mathcal{T}$ is incident to a vertex in T.*

For each non-leaf node $x \in V(\rho)$, we denote by $R(x)$ the subtree of ρ rooted at x and denote by $R_1(x)$, $R_2(x)$ the subtrees of ρ rooted at two children of x, respectively. Recall that $S = \{u_{x'} \mid x' \in V(\rho)\}$. We define sets $S_1(x) = \{u_{x'} \mid x' \in R_1(x)\}$, $S_2(x) = \{u_{x'} \mid x' \in R_2(x)\}$ and $S_0(x) = \{u_{x'} \mid x' \notin R(x)\}$. We then define $T_i(x) = T \cap S_i(x)$ for $i \in \{0, 1, 2\}$, and $T(x) = T_1(x) \cup T_2(x)$. For convenience, for every node $x \in V(\rho)$ at level i of ρ, we also say that u_x is a *level-i vertex in S*. Similarly, if node x is the parent of node x' in ρ, we also say that vertex u_x is the parent of vertex $u_{x'}$ in S.

We use the following observation, whose proof is deferred to Appendix D.

OBSERVATION 3.3. *There is an optimal Steiner Tree $\mathcal{T}$ of the instance (S, T, w_{N}), such that for every Steiner vertex u_x in $\mathcal{T}$: (i) u_x has either one or two neighbors in $T_0(x)$, exactly one neighbor in $T_1(x)$, and exactly one neighbor in $T_2(x)$; (ii) if we denote by W_1 and W_2 the connected components in the graph $\mathcal{T}' \setminus \{u_x\}$ that contains the $T_1(x)$-neighbor of u_x and the $T_2(x)$-neighbor of u_x, respectively, then $T_1(x) \subseteq V(W_1) \subseteq S_1(x)$, and $T_2(x) \subseteq V(W_2) \subseteq S_2(x)$; and (iii) for every vertex u_x at level at most $d-2$ in S, exactly one vertex from the set containing u_x and its two children belongs to $V(\mathcal{T})$.*

We are now ready to complete the proof of Claim 3.3. Let r be the root of tree ρ. Note that metric w_{N} and tree ρ are both determined by a single nonnegative integer d. In order to avoid ambiguity, we denote by $(S, T, w_{\mathsf{N}})_d$ the instance determined by d. For each $d \geq 0$, we define

- $A(d)$ as the minimum cost of a Steiner tree of instance $(S, T, w_{\mathsf{N}})_d$ that does not contain u_r; and
- $B(d)$ as the minimum cost of a Steiner tree of instance $(S, T, w_{\mathsf{N}})_d$ that contains u_r.

It is easy to verify that $A(1) = 2$ and $B(1) = 2$. We now show that, for each $d \geq 1$,

$$A(d+1) = A(d) + B(d) + 2^{d-1} + 2^d; \quad \text{and} \quad B(d+1) = 2 \cdot A(d) + 2^{d+1}. \tag{3.1}$$

On the one hand, let $\mathcal{T}$ be an optimal Steiner tree of instance $(S, T, w_{\mathsf{N}})_{d+1}$ that does not contain u_r. Let u_1, u_2 be the children of u_r. From Observation 3.3, exactly one of u_1, u_2 is contained in $\mathcal{T}$. Assume w.l.o.g. that $u_1 \in V(\mathcal{T})$. From Observation 3.3, $\mathcal{T}$ is the union of

- a Steiner tree of instance $(S_1(u_r), T_1(u_r), w_{\mathsf{N}})_{d+1}$ that contains u_1 (denote by $\mathcal{T}_1$);
- a Steiner tree of instance $(S_2(u_r), T_2(u_r), w_{\mathsf{N}})_{d+1}$ that does not contain u_2 (denote by $\mathcal{T}_2$); and
- an edge connecting $\mathcal{T}_1$ to $\mathcal{T}_2$.

Note that instances $(S_1(u_r), T_1(u_r), w_{\mathsf{N}})_{d+1}$ and $(S_2(u_r), T_2(u_r), w_{\mathsf{N}})_{d+1}$ are identical to the instance $(S, T, w_{\mathsf{N}})_d$, so $w(\mathcal{T}_1) = B(d)$ and $w(\mathcal{T}_2) = A(d)$. Note that the minimum weight of an edge connecting $\mathcal{T}_1$ to $\mathcal{T}_2$ is the edge connecting u_1 to any leaf in $T_2(u_r)$, which has cost $2^{d-1} + 2^d$. Therefore, $A(d+1) = A(d) + B(d) + 2^{d-1} + 2^d$.

On the one hand, let $\mathcal{T}$ be an optimal Steiner tree of instance $(S, T, w_{\mathsf{N}})_{d+1}$ that contains u_r. From Observation 3.3, both u_1, u_2 are contained in $\mathcal{T}$. Indeed, $\mathcal{T}$ is the union of

- a Steiner tree of instance $(S_1(u_r), T_1(u_r), w_{\mathsf{N}})_{d+1}$ that does not contain u_1 (denote by $\mathcal{T}_1$);
- a Steiner tree of instance $(S_2(u_r), T_2(u_r), w_{\mathsf{N}})_{d+1}$ that does not contain u_2 (denote by $\mathcal{T}_2$); and
- an edge connecting $\mathcal{T}_1$ to u_r and an edge connecting $\mathcal{T}_2$ to u_r.

Via similar arguments, we can show that $w(\mathcal{T}_1) = w(\mathcal{T}_2) = A(d)$. Note that the minimum weight of an edge connecting $\mathcal{T}_1$ to u_r is the edge connecting any leaf in $T_1(u_r)$ to u_r, which has cost 2^d. Therefore, $B(d+1) = 2 \cdot A(d) + 2^{d+1}$.

We now use the inequality 3.1 to complete the proof of Claim 3.3. From 3.1, we get that, for each $d \geq 1$, $A(d+1) = A(d) + 2 \cdot A(d-1) + 5 \cdot 2^{d-1}$. Using standard techniques and the initial values $A(0) = 0$ and $A(1) = 2$, we get that

$$A(d) = \left(\frac{5}{6}\right) \cdot d \cdot 2^d + \left(\frac{-1}{9}\right) \cdot (-1)^d + \left(\frac{1}{9}\right) \cdot 2^d.$$

Therefore, from 3.1, we get that

$$\begin{aligned} B(d) = 2 \cdot A(d-1) + 2^d &= 2 \cdot \left(\left(\frac{5}{6}\right) \cdot (d-1) \cdot 2^{d-1} + \left(\frac{-1}{9}\right) \cdot (-1)^{d-1} + \left(\frac{1}{9}\right) \cdot 2^{d-1}\right) + 2^d \\ &= \left(\frac{5}{6}\right) \cdot d \cdot 2^d + \left(\frac{2}{9}\right) \cdot (-1)^d + \left(\frac{5}{18}\right) \cdot 2^d. \end{aligned}$$

Therefore, $A(d), B(d) \geq (5/6) \cdot d \cdot 2^d$. This completes the proof of Claim 3.3. ∎

From Claim 3.2 and Claim 3.3, we get that

$$\frac{\mathsf{ST}(I_{\mathsf{N}})}{\mathsf{ST}(I_{\mathsf{Y}})} \geq \frac{(5/3) \cdot d \cdot 2^{d-1}}{(d+1) \cdot 2^{d-1}} \geq \frac{5}{3} - \frac{1}{d} \geq \frac{5}{3} - \varepsilon.$$

We now complete the proof of Theorem 1.1 using the metrics $w_{\mathsf{Y}}, w_{\mathsf{N}}$ defined above.

We construct a pair $\mathcal{D}_{\mathsf{Y}}, \mathcal{D}_{\mathsf{N}}$ of distributions on metric Steiner Tree instances (V', T', w') as follows. Set V' is fixed and contains n vertices. Let $\mathcal{F}$ be the set of all one-to-one mappings from V' to V. For each mapping $f \in \mathcal{F}$, we define a pair of instances I^f_{Y} and I^f_{N} as follows:

- $I^f_{\mathsf{Y}} = (V', f^{-1}(S), w^f_{\mathsf{Y}})$, where w^f_{Y} is defined as: $\forall v, v' \in V'$, $w^f_{\mathsf{Y}}(v, v') = w_{\mathsf{Y}}(f(v), f(v'))$;
- $I^f_{\mathsf{N}} = (V', f^{-1}(S), w^f_{\mathsf{N}})$, where w^f_{N} is defined as: $\forall v, v' \in V'$, $w^f_{\mathsf{N}}(v, v') = w_{\mathsf{N}}(f(v), f(v'))$;

where $f^{-1}(S) = \{v \in V' \mid f(v) \in S\}$. We then let $\mathcal{D}_{\mathsf{Y}}$ be the uniform distribution on all instances in $\left\{I^f_{\mathsf{Y}} \mid f \in \mathcal{F}\right\}$, and let $\mathcal{D}_{\mathsf{N}}$ be the uniform distribution on all instances in $\left\{I^f_{\mathsf{N}} \mid f \in \mathcal{F}\right\}$. Let $\mathcal{D}$ be the distribution that sample an instance from $\mathcal{D}_{\mathsf{Y}}$ with probability $1/2$, and sample an instance from $\mathcal{D}_{\mathsf{N}}$ with probability $1/2$.

It is easy to verify that for each mapping $f \in \mathcal{F}$, $\mathsf{ST}(I^f_{\mathsf{Y}}) = \mathsf{ST}(I_{\mathsf{Y}})$ and $\mathsf{ST}(I^f_{\mathsf{Y}}) = \mathsf{ST}(I_{\mathsf{Y}})$ hold, so any algorithm that with probability 0.51 estimates the cost to within factor $(5/3 - \varepsilon)$ can correctly report a random instance from $\mathcal{D}$ comes from $\mathcal{D}_{\mathsf{Y}}$ or $\mathcal{D}_{\mathsf{N}}$ with probability 0.51.

Recall that the metrics w_{Y} and w_{N} are identical on all pairs of vertices that are not both terminals. For each instance I^f_{Y}, we say a pair (v'_1, v'_2) of vertices in V' is *crucial* iff $v'_1, v'_2 \in f^{-1}(S)$, and we say that the pair (v'_1, v'_2) is *discovered* iff the pair (v'_1, v'_2) is queried by the algorithm. From Yao's minimax principle [7] and the above discussion, in order to distinguish between $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$ with probability 0.51, it is necessary that the algorithm discovers a crucial pair on at least 0.01-fraction of the instances in I_{Y}. Therefore, the proof of Theorem 1.1 is concluded by the following lemma.

LEMMA 3.1. *Any deterministic algorithm that discovers a crucial pair on at least* 0.01*-fraction of the instances in* I_{Y} *performs at least* $\Omega(n^2/2^{2d})$ *queries in expectation.*

Proof. Since f is a random one-to-one mapping from V' to V, the set $f^{-1}(S)$ is a random size-$|S|$ subset of V'. Therefore, the probability that a single query discoveres a crucial pair is $\binom{|S|}{2}/\binom{n}{2}$, and it follows that the expected number of queries that is required to discovers a crucial pair with probability at least $\Omega(1)$ is $\Omega\left(\binom{n}{2}/\binom{|S|}{2}\right) = \Omega(n^2/2^{2d})$. ∎

4 Algorithm for a $(2-\varepsilon_0)$-Estimation of Steiner Tree Cost

In this section we provide the proof of Theorem 1.3. Specifically, we will construct an algorithm, that, takes as input a set V of n points, a set $T \subseteq V$ of k terminals, and access to a metric w on V, estimates the metric Steiner Tree cost to within a factor of $(2-\varepsilon_0)$, for some universal constant ε_0 that does not depend on n and k, by performing $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$ queries. This section is organized as follows. First, in Section 4.1, we give an algorithm that, assumes that the induced metric of w on T is known to us upfront, estimates the metric Steiner Tree cost to within a factor of $(2-\varepsilon_0)$ with $\tilde{O}(n^{3/2} + n^{3/4} \cdot k)$ queries. The detail of some critical subroutine of the algorithm is provided in Section 4.2. Then in Section 4.3, we show how to remove the assumption that the induced metric of w on T is given, while still attaining a slighly worse (while still sublinear) query complexity $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$.

4.1 An Algorithm with the Terminal-Induced Metric Given Upfront

In this subsection, we assume that the metric on terminals induced by w is given to us upfront, and give an algorithm that estimates the metric Steiner Tree cost to within a factor of $(2-\varepsilon_0)$. We first give a high-level overview, and then describe the algorithm in detail and provide its analysis.

Overview of the algorithm We start by constructing a minimum spanning tree over the terminals, say $\mathcal{T}^*$. Let MST denote the weight of $\mathcal{T}^*$, so $\mathsf{ST}(V,T,w) \geq \mathsf{MST}/2$. The rest of the algorithm focuses on gathering "local evidence" to ascertain that $\mathsf{ST}(V,T,w)$ is bounded away from MST. If the algorithm fails to find the evidence to support this assertion, then we will be able to claim that $\mathsf{ST}(V,T,w)$ is bounded away from $\mathsf{MST}/2$.

We now describe what constitutes this local evidence. At a high-level, it is some property of the metric w that allows us to locally "restructure" the minimum terminal spanning tree maintaining connectivity among the terminals while reducing its total cost. To get some intuition for this process, let us consider a metric where all distances are 1 or 2. Suppose that the distances between all terminals are 2, the distances between all Steiner vertices are 2, and the distances between a terminal and a Steiner vertex is either 1 or 2. Clearly, $\mathsf{MST} = 2k-2$. Assume that a Steiner vertex v is at distance 1 to three terminals u_1, u_2, u_3. Now if we remove two edges from $\mathcal{T}^*$ such that terminals u_1, u_2, u_3 lie in different connected subtrees, and then add the edges $(v,u_1), (v,u_2)$, and (v,u_3), then we get another Steiner Tree that now contains v. Note that, in this process we have deleted two edges of cost 2 each and added three edges of cost 1 each, so essentially the total cost decreases by 1. We view this "Steiner vertex v connects to terminals u_1, u_2, u_3 via length-1 edges" structure as a "local evidence that separates $\mathsf{ST}(V,T,w)$ from MST".

It is not hard to observe that, this type of evidence is both local and aggregatable, e.g., if a Steiner vertex v is at distance 1 to terminals u_1-u_3 and another Steiner vertex v' is at distance 1 to terminals u_4-u_7, then we can "save a total of $(4-3)+(6-4)=3$ units of cost" from MST. The process of identifying the best way to aggregate these local cost-saving improvements is reminiscent of solving an instance of *Set Cover*. Specifically, if we define, for each Steiner vertex v, a set W_v containing all terminals $u \in T$ with $w(u,v) = 1$, then a good aggregation of the local evidence is a collection of a small number of sets W_v that cover many terminals. In particular, if we denote $\mathcal{W} = \{W_v \mid v \notin T\}$, then using similar "local MST restructure"-type arguments, we can show that $\mathsf{ST}(V,T,w) \leq \mathsf{MST} - \Omega(k - \mathsf{SC}(T,\mathcal{W}))$, where $\mathsf{SC}(T,\mathcal{W})$ is the minimum solution size of the Set Cover instance $(T,\mathcal{W})$. Therefore, our goal now is to estimate the value of $k - \mathsf{SC}(T,\mathcal{W})$ to within an additive εk factor (and some small multiplicative factor). We provide an $\tilde{O}(n^{3/2} + n^{3/4} \cdot k)$-query algorithm for this task in Section 4.2, and then show how to implement this algorithm when the terminal-induced metric is not given upfront, with a slightly worse query complexity $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$.

We next describe how the ideas outlined above for the special case of $(1,2)$-metric, can be extended to the general case. Let us consider the construction of the minimum spanning tree $\mathcal{T}^*$ on T using Kruskal's algorithm. Assume that the weight of every terminal-terminal edge is $(1+\varepsilon)^i$ for some non-negative integer i. Then in the first round we add all weight-1 edges and obtain some connected components (called *clusters*), and in the second round we add all weight-$(1+\varepsilon)$ edges, and some clusters in the first round are merged into bigger clusters, etc. The main observation is that, in every round, we can use the Steiner vertices to locally restructure this cluster-merging step just as the special case. In particular, if there exists a Steiner vertex v and three first-round clusters U_1, U_2, U_3, such that U_1, U_2, U_3 are merged in the second round, and $w(v,U_1), w(v,U_2), w(v,U_3)$ are close to $(1+\varepsilon)/2$, then we can replace two weight-$(1+\varepsilon)$ edges with three weight-roughly-$(1+\varepsilon)/2$ edges, thereby saving the total cost by roughly $(1+\varepsilon)/2$ without destroying the connectivity between the terminals in U_1, U_2, U_3.

Therefore, the main framework of our algorithm is to compute the hierarchical structure of the terminal minimum spanning tree $\mathcal{T}^*$ and use this set-cover-type algorithm at every "level" of $\mathcal{T}^*$ to search for local evidence that there is a Steiner tree of cost significantly better than $\mathcal{T}^*$.

There is one subtlety in the above algorithmic framework, which is that the cardinality-2 sets do not provide cost-saving. In particular, if we replace one terminal-terminal edge with two terminal-Steiner edge (of weight about half of that of a terminal-terminal edge), then the total cost will not decrease. More concretely, if we consider the metric w_{Υ} defined in Section 3, which is the shortest-path metric induced by a complete binary tree with edge cost geometrically decreasing along with the levels, then when we construct Set Cover instances on different levels, we will only get cardinality-2 sets and ended up finding no local evidence at all, but in fact $\mathsf{ST}(V, T, w_{\Upsilon}) = \mathsf{MST}/2$ holds. To overcome this issue, we introduce another evidence-searching subroutine that goes beyond a single level of the hierarchical structure of $\mathcal{T}^*$ while involving only $O(1)$ vertices. This is based on the observation that, in this very special case, although local evidence cannot be found at a single level, it can be found by looking at two consecutive levels and only focusing on sets of $O(1)$ vertices. It turns out that incorporating this subroutine into the above framework gives us the desired algorithm.

We now describe the algorithm in detail. Recall that we are given an instance (V, T, w) of the metric Steiner Tree problem and are allowed to perform queries to metric w. Also recall that the induced metric of w on T is also given to us upfront. We use the following parameters: $\varepsilon_0 = 2^{-40}; \varepsilon = 2^{-20}$.

Step 1. Computing an MST on T and its hierarchical structure We pre-process the instance (V, T, w) as follows. Let $D = \max\{w(u, u') \mid u, u' \in T\}$. While there exists a pair $u, u' \in T$ with $w(u, u') \leq D/k^2$, we delete an arbitrary one of them from T and V. We repeat this until all pairwise distances between vertices of T are at least D/k^2. Let T' be the resulting terminal set we get, and define $V' = (V \setminus T) \cup T'$. We then scale the metric w by defining another metric w' on V' such that for every pair $v, v' \in V$, $w'(v, v') = w(v, v')/\min\{w(u, u') \mid u, u' \in T'\}$, so now the distance (under w') between every pair of terminals in T' is at least 1 and at most k^2. It is easy to verify that: (i) the metric Steiner Tree cost $\mathsf{ST}(V', T', w')$ of instance (V', T', w') is within factor $(1 + O(1/k))$ of $\mathsf{ST}(V, T, w)/\min\{w(u, u') \mid u, u' \in T'\}$; and (ii) every distance query to w' can be simulated by a distance query to w. Therefore, from now on we work with instance (V', T', w'), and we will construct an algorithm that with high probability estimates the value of $\mathsf{ST}(V', T', w')$ to within a factor of $(2 - 2\varepsilon_0)$. Eventually, when the algorithm returns an estimate X of $\mathsf{ST}(V', T', w')$, we return $X \cdot \min\{w(u, u') \mid u, u' \in T'\}$ as the output estimate of $\mathsf{ST}(V, T, w)$. It is easy to verify that the final output of the algorithm is with high probability a $(2 - \varepsilon_0)$-approximation of $\mathsf{ST}(V, T, w)$. For convenience, in the remainder of this section, we rename the vertex set V', the terminal set T' and the metric w' by V, T, w, respectively.

Let $L = \lceil \log_{1+\varepsilon} k^2 \rceil$. For every pair $u, u' \in T$, we say that the edge (u, u') is *at level* i (or (u, u') is an *level-*i edge), iff $(1 + \varepsilon)^{i-1} \leq w(u, u') < (1 + \varepsilon)^i$. Clearly, every edge connecting a pair of terminals is at level at most L. For each $1 \leq i \leq L$, we define H_i to be the graph on T that contain all edges up to level i, and we define H_0 to be the empty graph on T. For each index $1 \leq i \leq L$, we define $\mathcal{S}_i$ as the collection of vertex sets of connected components of graph H_{i-1}. That is, each set in $\mathcal{S}_i$ contains all vertices of some connected component of H_{i-1}. Clearly, each $\mathcal{S}_i$ is a partition of T.

Let $\mathcal{S} = \bigcup_{1 \leq i \leq L} \mathcal{S}_i$. It is easy to verify that $\mathcal{S}$ is a laminar family. That is, every pair S, S' of sets in $\mathcal{S}$, either $S \subseteq S'$, or $S' \subseteq S$, or $S \cap S = \emptyset$. We associate with $\mathcal{S}$ a partition tree $\mathcal{T}$ as follows. The vertex set of $\mathcal{T}$ contains, for each set $S \in \mathcal{S}$, a node x_S representing the set S. The edge set of $\mathcal{T}$ contains, for each pair $S, S' \in \mathcal{S}$ such that $S \subseteq S'$ and S, S' lie on consecutive levels, an edge $(x_S, x_{S'})$. In this case, we say that $x_{S'}$ is the parent node of x_S (and x_S is a child node of $x_{S'}$); similarly, we say that S' is a parent set of S (and x_S is a child set of S'). Note that $\mathcal{S}_L$ contains a single set, and its corresponding node in $\mathcal{T}$ is designated as the root of $\mathcal{T}$. It is easy to verify that $\mathcal{T}$ is a tree.

Lastly, we compute a minimum spanning tree $\mathcal{T}^*$ on T, and denote $\mathsf{MST} = w(\mathcal{T}^*)$. As the induced metric of w on T is given to us upfront, in this step we did not perform any additional queries.

Step 2. Finding local evidence using a set-cover-type subroutine We start by introducing a set-cover-type subroutine that we will use in this step.

Algorithm AlgSetCover. Let $(U, \mathcal{W})$ be an instance of the Set Cover problem, where U is a collection of elements and $\mathcal{W}$ is a collection of subsets of U. For convenience, throughout the paper, when we consider an instance $(U, \mathcal{W})$ of Set Cover, we always assume that $\mathcal{W}$ contains, for each element $u \in U$, a singleton set $\{u\}$,

so $|\mathcal{W}| \geq |U|$, and if we denote by $\mathsf{SC}(U,\mathcal{W})$ the size of the smallest set cover for the instance $(U,\mathcal{W})$, then $\mathsf{SC}(U,\mathcal{W}) \leq |U|$. The elements of U and the number of sets in $\mathcal{W}$ are known to us, but we do not know which elements each set of $\mathcal{W}$ contains. We are allowed to perform queries to a *membership oracle* of instance $(U,\mathcal{W})$, which is an oracle that, takes as input an element $u \in U$ and a set $W \in \mathcal{W}$, returns whether or not u belongs to W. A query to the membership oracle of instance $(U,\mathcal{W})$ is also called a *membership query* to instance $(U,\mathcal{W})$.

For a collection $\mathcal{W}$ of subsets of U, we denote by $\mathcal{W}_{\neq 2}$ the collection that contains all sets in $\mathcal{W}$ with size not equal to 2. For positive real numbers X, Y, a, b with $a > 1$, we say that X is an (a,b)-*estimation of* Y iff $Y \leq X \leq aY + b$.

We use the following theorem, whose proof is deferred to Section 4.2.

THEOREM 4.1. *There is a polynomial-time randomized algorithm called* AlgSetCover*, that, given any instance $(U,\mathcal{W})$ of Set Cover and any constant $0 < \varepsilon < 1$, with high probability, returns a $(4,\varepsilon|U|)$-estimation of $\big(|U| - \mathsf{SC}(U,\mathcal{W}_{\neq 2})\big)$, by performing $O((|\mathcal{W}|^{3/2} + |\mathcal{W}|^{3/4}|U|)(\log|\mathcal{W}|)^2/\varepsilon^3)$ membership queries to the instance $(U,\mathcal{W})$.*

We remark that improving the query complexity of the result above will immediately improve the query complexity for the $(2-\varepsilon_0)$-estimation algorithm. Also, note that the number of queries needed by a naive algorithm to solve the estimation problem above would be $O(|U||\mathcal{W}|)$, the size of the input description. So the theorem above provides an estimation algorithm that is sublinear in the size of the input description whenever $|\mathcal{W}|^{3/2} = o(|U||\mathcal{W}|)$.

For each index $1 \leq i \leq L$ and each pair $u, u' \in T$, we let $w_i(u,u') = w(u,u')$ if (u,u') is a level-i edge; otherwise we let $w_i(u,u') = 0$. Consider the minimum spanning tree $\mathcal{T}^*$ computed in Step 1. For each $1 \leq i \leq L$, we say that level i is *light* iff $w_i(\mathcal{T}^*) < \mathsf{MST}/(L\log n)$ (that is, the total weight of all level-i edges in $\mathcal{T}^*$ is less than $\mathsf{MST}/(L\log n)$); otherwise we say that level i is *heavy*. For a set E of edges in $E(\mathcal{T}^*)$, we define $w_i(E) = \sum_{(u,u')\in E} w_i(u,u')$ and call $w_i(E)$ the *level-i weight* of set E. The following observation is immediate.

OBSERVATION 4.1. $\sum_{i:\ \text{level } i \text{ is light}} w_i(\mathcal{T}^*) \leq \mathsf{MST}/\log n$.

Before we describe the set-cover-type subroutine in detail, we give some intuition. We intend to find local evidence on different levels separately. For each $0 \leq i \leq L$ and for each set $S \in \mathcal{S}_i$, we think of vertices in S as already connected via edges up to level i. So we will search for better ways to connect different sets in $\mathcal{S}_i$ via Steiner vertices. However, for Steiner vertex v and set $S \in \mathcal{S}$, naively it takes $O(|S|)$ queries to compute $w(v,S)$, which we cannot afford as $|S|$ can be as large as k. Therefore, for each $S \in \mathcal{S}_i$, we will first compute a subset $\tilde{S} \subseteq S$ as its "representative", such that $|\tilde{S}|$ is small, and the values $w(v,S)$ and $w(v,\tilde{S})$ are close for all Steiner vertices v.

We now describe the set-cover-type subroutine in detail. For each level i that is heavy, we construct an instance $I_i = (U_i, \mathcal{W}_i)$ of Set Cover as follows. Recall that $\mathcal{S}_i$ is the collection of all level-i sets. We first compute, for each index $0 \leq i \leq L$ and for each set $S \in \mathcal{S}_i$, a maximal subset $\tilde{S}$ of S, such that the distance between every pair of terminals in $\tilde{S}$ is at least $\varepsilon \cdot (1+\varepsilon)^i$. We define $\tilde{\mathcal{S}}_i$ to be the collection that contains all such sets $\tilde{S}$ with $|\tilde{S}| \leq (L\log^2 n)/\varepsilon$. The ground set U_i in instance I_i is defined as $U_i = \left\{x_S \mid \tilde{S} \in \tilde{\mathcal{S}}_i\right\}$. The collection $\mathcal{W}_i$ contains, for each vertex $v \in V \setminus T$, a set $W_i(v)$ that is defined as $W_i(v) = \left\{x_S \mid w(v,\tilde{S}) \leq (3/5)\cdot(1+\varepsilon)^i\right\}$. We use the following simple observations.

OBSERVATION 4.2. *$|U_i| \leq k$, $|\mathcal{W}_i| \leq n-k$, and every membership query in the instance $(U_i,\mathcal{W}_i)$ can be simulated by at most $(L\log^2 n)/\varepsilon$ distance queries to the metric w.*

OBSERVATION 4.3. *For each level i that is heavy, $|U_i| \geq (1 - O(1/\log n)) \cdot |\mathcal{S}_i|$.*

Proof. Recall that $U_i = \left\{x_S \mid \tilde{S} \in \tilde{\mathcal{S}}_i\right\}$. It suffices to show that, if level i is heavy, then there are at most $O(1/\log n)$ fraction of sets S in $\mathcal{S}_i$ with $|\tilde{S}| > (L\log^2 n)/\varepsilon$. Define $X = \bigcup_{S\in\mathcal{S}_i} \tilde{S}$. Note that every pair of vertices in X are at distance at least $\varepsilon\cdot(1+\varepsilon)^i$ in w. Therefore, $|X| \leq 1 + \mathsf{MST}/(\varepsilon\cdot(1+\varepsilon)^i) \leq 2\cdot\mathsf{MST}/(\varepsilon\cdot(1+\varepsilon)^i)$. On the other hand, since level i is heavy, $w_i(\mathcal{T}^*) \geq \mathsf{MST}/(L\log n)$, and so $|\mathcal{S}_i| \geq \mathsf{MST}/(L\cdot\log n\cdot(1+\varepsilon)^{i-1})$. Altogether,

$$\frac{L\log^2 n}{\varepsilon} \cdot \left|\left\{S \in \mathcal{S}_i \mid |\tilde{S}| > \frac{L\log^2 n}{\varepsilon}\right\}\right| \leq |X| \leq \frac{2\cdot\mathsf{MST}}{\varepsilon\cdot(1+\varepsilon)^i} \leq |\mathcal{S}_i| \cdot \frac{2L\cdot\log n}{\varepsilon\cdot(1+\varepsilon)},$$

and it follows that there are at most $O(1/\log n)$ fraction of sets S in $\mathcal{S}_i$ with $|\tilde{S}| > (L\log^2 n)/\varepsilon$. ∎

Then for each $0 \le i \le L$, we apply the algorithm $\mathsf{AlgSetCover}$ to instance $I_i = (U_i, \mathcal{W}_i)$ constructed above, and obtain an estimate X_i of $\big(|U_i| - \mathsf{SC}(U_i, (\mathcal{W}_i)_{\neq 2})\big)$. If $\sum_{0\le i\le L}(1+\varepsilon)^i \cdot X_i > 2^{30}\cdot\varepsilon_0\cdot\mathsf{MST}$, then we return $(1-\varepsilon_0)\cdot\mathsf{MST}$ as an estimate of $\mathsf{ST}(V,T,w)$. Otherwise, we go to the next step. Since the algorithm in Theorem 4.1 performs $(|U_i||\mathcal{W}_i|^{3/4}+|\mathcal{W}_i|^{3/2})\cdot(\log(|U_i|+|\mathcal{W}_i|))^{O(1)} \le \tilde{O}(n^{3/2}+n^{3/4}k)$ queries, using Observation 4.2, we get that the algorithm performs $\tilde{O}(n^{3/2}+n^{3/4}k)\cdot O(L\log^2 n) = \tilde{O}(n^{3/2}+n^{3/4}k)$ queries on the metric w in this step.

Analysis of Step 2 when the algorithm returns $(1-\varepsilon_0)\cdot\mathsf{MST}$ as the estimate of $\mathsf{ST}(V,T,w)$ Before we proceed to describe the next steps of the algorithm, we first show in this subsection that, if we collected enough local evidence in this step, then indeed $\mathsf{ST}(V,T,w)$ is bounded away from MST. Specifically, we will show that, if $\sum_{0\le i\le L}(1+\varepsilon)^i\cdot X_i > 2^{30}\cdot\varepsilon_0\cdot\mathsf{MST}$, then $\mathsf{ST}(V,T,w) \le (1-\varepsilon_0)\cdot\mathsf{MST}$. Since $\mathsf{ST}(V,T,w)\ge\mathsf{MST}/2$, our estimate in this case is indeed a $(2-2\varepsilon_0)$-approximation of $\mathsf{ST}(V,T,w)$. For each $0\le i\le L$, we define $Y_i = |U_i| - \mathsf{SC}(U_i,(\mathcal{W}_i)_{\neq 2})$. Define $L' = \lceil \log_{1+\varepsilon} 2\rceil$, so L' is a constant between 2^{19} and 2^{20} from the definition of ε. We start by proving the following claim.

Claim 4.1. *For each $0\le i\le L-1$, there exists a set E_i of edges, each connecting a terminal in T to a Steiner vertex in $V\setminus T$, such that (i) the vertex sets of the connected components in graph $H_{i-1}\cup E_i$ are exactly the sets in $\mathcal{S}_{i+L'-1}$; and (ii) $w(E_i)\le\big(\sum_{i\le s<i+L'} w_s(\mathcal{T}^*)\big) - (1/20)\cdot(1+\varepsilon)^i\cdot|Y_i|$.*

Proof. Fix an index $0\le i\le L-1$ and consider the instance $I_i = (U_i,(\mathcal{W}_i)_{\neq 2})$. Let $\mathcal{W}_i^*$ be an optimal set cover of instance $(U_i,(\mathcal{W}_i)_{\neq 2})$. We compute a sub-collection $\widetilde{\mathcal{W}}_i$ of $\mathcal{W}_i^*$ as follows. We process the sets of $\mathcal{W}_i^*$ in an arbitrary order. Upon processing each set in $\mathcal{W}_i^*$, we add it into to $\widetilde{\mathcal{W}}_i$ iff the set contains at least two elements that are not contained in all previously processed sets in $\mathcal{W}_i^*$. Denote the resulting set by $\widetilde{\mathcal{W}}_i = \{W_i(v_1),\ldots,W_i(v_r)\}$, where the sets are indexed according to the order in which they are added to $\widetilde{\mathcal{W}}_i$. For each $1\le j\le r-1$, define $U_i(v_j) = W_i(v_j)\setminus\big(\bigcup_{1\le t\le j-1} W_i(v_t)\big)$. From the above discussion, sets $U_i(v_1),\ldots,U_i(v_r)$ are mutually disjoint and each containing at least two elements.

On the one hand, we show that $\sum_{1\le j\le r}|U_i(v_j)|\ge 2\cdot|Y_i|$. In fact, in the process of iteratively processing the sets of $\mathcal{W}_i^*$ to obtain a subcollection $\widetilde{\mathcal{W}}_i$, every set that is not added into $\widetilde{\mathcal{W}}_i$ contains exactly one element that does not lie in previously processed sets. Therefore, at least Y_i sets are eventually added to $\widetilde{\mathcal{W}}_i$. Note that $|U_i(v_j)|\ge 2$ for each $1\le j\le r$, we get that $\sum_{1\le j\le r}|U_i(v_j)|\ge 2\cdot|Y_i|$.

On the other hand, we construct the set E_i of edges via the following iterative process. Throughout, we maintain a set $\hat{E}$ of edges, that is initialized to be the set of all edges in $\mathcal{T}^*$ from level i to level $i+L'$ (so the initial total weight of $\hat{E}$ is $\sum_{i\le s<i+L'} w_s(\mathcal{T}^*)$). We will ensure that set $\hat{E}$ always satisfies the property (i) in the claim. That is, the vertex sets of the connected components in graph $H_{i-1}\cup\hat{E}$ are exactly the sets in $\mathcal{S}_{i+L'-1}$. We iteratively process Steiner vertices $v_1,\ldots,v_r$ while modifying set $\hat{E}_i$ as follows. Consider now the iteration of processing v_j for some $1\le j\le r$. Denote $U_i(v_j) = \{S_1,\ldots,S_p\}$, where $S_1,\ldots,S_p\in\mathcal{S}_{i-1}$. Clearly, sets $S_1,\ldots,S_p$ are subsets of the same set in $\mathcal{S}_{i+L'-1}$, as each pair of them is at distance at most $(6/5)\cdot(1+\varepsilon)^i < (1+\varepsilon)^{i+L'-1}$ in w. Moreover, from the definition of $W_i(v_j)$, for each $1\le q\le p$, there exists a terminal $u_q\in S_q$ such that $w(v_j,u_q)\le(3/5)\cdot(1+\varepsilon)^i$. We distinguish between the following two cases.

Case 1. $|U_i(v_j)|\ge 3$. We simply add edges $(v_j,u_1),\ldots,(v_j,u_p)$ into the set $\hat{E}$. Since initially set $\hat{E}$ contains all edges of $E(\mathcal{T}^*)$ at level i, and since elements in $U_i(v_j)$ do not belong to any other set in $\{U_i(v_1),\ldots,U_i(v_r)\}$, it is easy to see that, we can delete $(p-1)$ edges from the current set $\hat{E}$ that are level-i edges of $E(\mathcal{T}^*)$, such that the resulting set $\hat{E}$ still satisfies property (i) in the claim.

Case 2. $|U_i(v_j)| = 2$. Since $|W_i(v_j)|\ge 3$, there must exist another set $S_0\in\mathcal{S}_i$ such that element x_{S_0} is contained in $W_i(v_j)$ and some previous set $W_i(v_{j'})$ (for some $j'<j$), and so there exists a terminal $u_0\in S_0$ with $w(v_j,u_0)\le(3/5)\cdot(1+\varepsilon)^i$. We simply add edges $(v_j,u_0),(v_j,u_1),(v_j,u_2)$ into the set $\hat{E}$. For similar reasons, it is easy to see that we can delete 2 edges from the current set $\hat{E}$ that are level-i edges of $E(\mathcal{T}^*)$, such that the resulting set $\hat{E}$ still satisfies property (i) in the claim.

In either case, we add t_j edges into set $\hat{E}$ and delete (t_j-1) from set $\hat{E}$, for some $t_j\ge 3$ (and in fact $t_j\ge|U_i(v_j)|$). Since the edges that are added to set $\hat{E}$ have weight at most $(3/5)\cdot(1+\varepsilon)^i$, and the edges that are deleted from set $\hat{E}$ have weight at least $(1+\varepsilon)^{i-1}$. In the iteration of processing v_j, the total weight of $\hat{E}$

decreases by at least $(t_j - 1) \cdot (1+\varepsilon)^{i-1} - t_j \cdot \left((3/5) \cdot (1+\varepsilon)^i\right) \geq (1/40) \cdot t_j \cdot (1+\varepsilon)^i$, as $t_j \geq 3$. Therefore, the accumulative decrease of the total weight of $\hat{E}$ has weight is at least

$$\sum_{1 \leq j \leq r} \frac{t_j \cdot (1+\varepsilon)^i}{40} \geq \sum_{1 \leq j \leq r} \frac{|U_i(v_j)| \cdot (1+\varepsilon)^i}{40} \geq \frac{|Y_i| \cdot (1+\varepsilon)^i}{20}.$$

Denote the resulting set $\hat{E}$ by E_i, and the claim now follows. ∎

We now use Claim 4.1 to complete the analysis of Step 2. For each index $0 \leq j \leq L' - 1$, we define $\alpha_j = \sum_{0 \leq i \leq L:\ i \equiv j \pmod{L'}} Y_i \cdot (1+\varepsilon)^i$. Clearly, $\sum_{0 \leq j < L'} \alpha_j = \sum_{0 \leq i \leq L} (1+\varepsilon)^i \cdot Y_i$, and so there exists some $0 \leq j^* \leq L' - 1$ such that $\alpha_{j^*} \geq (1/L') \cdot \sum_{0 \leq i \leq L} (1+\varepsilon)^i \cdot Y_i$. We now define E' as the set that contains (i) all edges of $\mathcal{T}^*$ that are at level $0, 1, \ldots, j^* - 1$; and (ii) all edges of $E_{j^*}, E_{j^*+L'}, E_{j^*+2L'}, \ldots$, that are given by Claim 4.1. From Claim 4.1, it is easy to verify that the graph induced by edges of E' is a Steiner tree of instance (V, T, w), and moreover,

$$\begin{aligned}
w(E') &\leq w(\mathcal{T}^*) - \frac{1}{20} \cdot \alpha_{j^*} \\
&\leq w(\mathcal{T}^*) - \frac{1}{20L'} \cdot \sum_{0 \leq i \leq L} (1+\varepsilon)^i \cdot Y_i \\
&\leq w(\mathcal{T}^*) - \frac{1}{20L'} \cdot \sum_{0 \leq i \leq L} (1+\varepsilon)^i \cdot \frac{X_i - \varepsilon |U_i|}{4} \\
&= w(\mathcal{T}^*) - \frac{1}{20L'} \cdot \left(\sum_{0 \leq i \leq L} \frac{(1+\varepsilon)^i \cdot X_i}{4} - \sum_{0 \leq i \leq L} \frac{(1+\varepsilon)^i \cdot \varepsilon |U_i|}{4} \right) \\
&\leq w(\mathcal{T}^*) - \frac{1}{20L'} \cdot \left(\frac{2^{30} \cdot \varepsilon_0 \cdot \mathsf{MST}}{4} - \varepsilon \cdot \mathsf{MST} \right) \leq (1 - \varepsilon_0) \cdot w(\mathcal{T}^*),
\end{aligned}$$

according to the definition of $\varepsilon, \varepsilon_0$ and the fact that $2^{19} \leq L' \leq 2^{20}$. This shows that our estimate in this case is indeed a $(2 - 2\varepsilon_0)$-approximation of $\mathsf{ST}(V, T, w)$.

Step 3. Finding local evidence using a 4-vertex subroutine In the third and last step, we focus on finding one specific type of local evidence, by querying distances related to groups of 4 vertices.

Recall that we have computed a laminar family $\mathcal{S}$ of subsets of terminals in T and its partition tree $\mathcal{T}$. We say that a node x_S in $\mathcal{T}$ is *good* iff x_S has exactly two children in $\mathcal{T}$, and each child node of x_S also has exactly two children in $\mathcal{T}$. In this case, we also say that the corresponding set S in $\mathcal{S}$ is good. Consider a good set $S \in \mathcal{S}$. Let S_1, S_2 be its child sets, let S_{11}, S_{12} be the child sets of S_1, and let S_{21}, S_{22} be the child sets of S_2. We define the *advantage* of set S, denoted by $\mathsf{adv}(S)$, as follows. We define $w^*(S) = w(S_{11}, S_{12}) + w(S_{21}, S_{22}) + w(S_1, S_2)$, that is, the total edge weight in $\mathcal{T}^*$ that is used to connect the four sets $S_{11}, S_{12}, S_{21}, S_{22}$ into a single set S. We say that a set Y *represents* S, iff Y contains exactly four terminals $u_{11}, u_{12}, u_{21}, u_{22} \in T$, such that $u_{11} \in S_{11}$, $u_{12} \in S_{12}$, $u_{21} \in S_{21}$, and $u_{22} \in S_{22}$. For a set Y that represents S, we define $\mathsf{adv}(S, Y) = w^*(S) - \min_{v \in V \setminus T} \{\mathsf{ST}(Y \cup \{v\}, Y, w)\}$, so $\mathsf{adv}(S, Y)$ is the maximum cost reduction (local evidence) that can be achieved with the help of any single Steiner vertex. We define $\mathsf{adv}(S) = \max_Y \{\mathsf{adv}(S, Y)\}$. Intuitively, in this step we are searching for the benefit of utilizing one Steiner vertex to restructure a specific type of 2-level local structure in $\mathcal{T}^*$.

We now describe the algorithm in this step. We denote by $\mathcal{S}_g$ the collection of all good sets in $\mathcal{S}$. For each $0 \leq i \leq L$, let $\mathcal{S}_g^i$ be the set of all level-i good sets. For each good set $S \in \mathcal{S}_g^i$, similar to Step 2, we compute a maximal subset $\tilde{S}$ of S, such that the distance between every pair of terminals in $\tilde{S}$ is at least $\varepsilon \cdot (1+\varepsilon)^i$. We then define

$$A_i = \sum_{S \in \mathcal{S}_g^i:\ |\tilde{S}| \leq (L \log^2 n/\varepsilon)} \mathsf{adv}(S) \cdot \mathbf{1}\left[\mathsf{adv}(S) \geq \varepsilon^{3/4} (1+\varepsilon)^i\right].$$

However, we are unable to compute A_i using few queries, as the number of sets S with $|\tilde{S}| \leq (L \log^2 n/\varepsilon)$ can be large, and for each such set, computing $\mathsf{adv}(S)$ takes $\Omega(n)$ queries since we need to try all Steiner vertuces. To get

around this obstacle, we will compute, for each i, an estimate B_i of A_i as follows. Let $\hat{\mathcal{S}}_g^i$ be the collection of all level-i good sets S with $|\tilde{S}| \leq (L\log^2 n)/\varepsilon$. We sample $(\log n)/\varepsilon^{10}$ sets in $\hat{\mathcal{S}}_g^i$. For each sampled set $S \in \hat{\mathcal{S}}_g^i$, we first query all distances between any terminal in $\tilde{S}$ and any vertex in $V \setminus T$. We then try all four-terminal sets Y, such that $Y \subseteq \tilde{S}$ and Y represents S (note that there are at most $O((L\log^2 n/\varepsilon)^4)$ such sets), and compute $\mathsf{adv}(S,Y)$ using the acquired distance information. We then let $\mathsf{adv}(S)$ be the maximum over all values $\mathsf{adv}(S,Y)$ that we computed. We then let B_i be the sum of $\mathsf{adv}(S)$ for all sampled sets S such that $\mathsf{adv}(S) > (\varepsilon^{3/4}/2)\cdot(1+\varepsilon)^i$, namely

$$B_i = \sum_{S \text{ sampled}} \left((\varepsilon^{3/4}/2)\cdot(1+\varepsilon)^i\right)\cdot \mathbf{1}\left[\mathsf{adv}(S) \geq (\varepsilon^{3/4}/2)\cdot(1+\varepsilon)^i\right].$$

Finally, we compute $\sum_{0\leq i\leq L} B_i \cdot |\hat{\mathcal{S}}_g^i|/(\log n/\varepsilon^{10})$. If it is greater than $5\varepsilon^{3/4}\cdot\mathsf{MST}$, then we return $(1-\varepsilon_0)\cdot\mathsf{MST}$ as an estimate of $\mathsf{ST}(V,T,w)$. Otherwise, we return MST as an estimate of $\mathsf{ST}(V,T,w)$. This completes the description of Step 3, and also completes the description of the whole algorithm. In Step 3, we performed in total $\left((\log n)/\varepsilon^{10}\right)\cdot O((L\log^2 n/\varepsilon)^4)\cdot O(n) = \tilde{O}(n)$ queries. Overall, the algorithm performs $\tilde{O}(n^{3/2}+n^{3/4}k)+\tilde{O}(n) = \tilde{O}(n^{3/2}+n^{3/4}k)$ queries.

Analysis of Step 3 when the algorithm returns $(1-\varepsilon_0)\cdot\mathsf{MST}$ as the estimate of $\mathsf{ST}(V,T,w)$ We now show that, if we collected enough local evidence in this step, then indeed $\mathsf{ST}(V,T,w)$ is bounded away from MST. Specifically, we will show that, with high probability, if $\sum_{0\leq i\leq L} B_i\cdot|\hat{\mathcal{S}}_g^i|/(\log n/\varepsilon^{10}) > 5\varepsilon^{3/4}\cdot\mathsf{MST}$, then $\mathsf{ST}(V,T,w) \leq (1-\varepsilon_0)\cdot\mathsf{MST}$. Since $\mathsf{ST}(V,T,w)\geq\mathsf{MST}/2$, this implies that our estimate in this case is indeed a $(2-2\varepsilon_0)$-approximation of $\mathsf{ST}(V,T,w)$.

We first show that, if $\sum_{0\leq i\leq L} A_i > (2\varepsilon_0)\cdot\mathsf{MST}$ holds, then $\mathsf{ST}(V,T,w)\leq(1-\varepsilon_0)\cdot\mathsf{MST}$. Note that we have $\sum_{0\leq i\leq L}A_i = \sum_{i\text{ even}}A_i + \sum_{i\text{ odd}}A_i$. We assume that $\sum_{i\text{ even}}A_i > \varepsilon_0\cdot\mathsf{MST}$ (the case where $\sum_{i\text{ odd}}A_i>\varepsilon_0\cdot\mathsf{MST}$ is symmetric). Consider now the minimum spanning tree $\mathcal{T}^*$ computed in Step 1. We will iteratively modify $\mathcal{T}^*$ by processing good sets in $\mathcal{S}_g$ and eventually obtain a Steiner Tree of instance (V,T,w), such that the total cost decreases by at least $\sum_{i\text{ even}}A_i$.

We now formally describe the iterative modification process. Throughout the process, we will maintain a Steiner Tree $\mathcal{T}$ of instance (V,T,w), that is initialized to be $\mathcal{T}^*$. Let $\mathcal{S}_g^e$ be the set of all good sets S at an even-index level, such that $|\tilde{S}|\leq(L\log^2 n)/\varepsilon$. In each iteration, we pick a set $S\in\mathcal{S}_g^e$ that is at the lowest level, and after processing S we discard it from $\mathcal{S}_g^e$. We now describe the iteration of processing set S. Let sets $S_1,S_2,S_{11},S_{12},S_{21},S_{22}$ be defined as before. Before this iteration, each of these six sets induce a connected subgraph of the current tree $\mathcal{T}$, and they are connected in $\mathcal{T}$ by an edge e_1 connecting S_{11} to S_{12}, an edge e_2 connecting S_{21} to S_{22}, and an edge e connecting S_1 to S_2. Clearly, the total cost of these three edges is at most $w^*(S)$, by the definition of $w^*(S)$. Consider now the set Y that represents S such that $\mathsf{adv}(S)=\mathsf{adv}(S,Y)$. Let E'_Y be the set of edges in the Steiner Tree that achieves $\mathsf{adv}(S,Y)$. In this iteration we simply replace edges e,e_1,e_2 with edges in E'_Y. It is easy to see that the resulting tree $\mathcal{T}$ is still a Steiner Tree of instance (V,T,w), and the decrease of total weight is $w^*(S)-w(E'_Y)=\mathsf{adv}(S,Y)=\mathsf{adv}(S)$. Therefore, after processing all sets in $\mathcal{S}_g^e$ in this way, we obtain a Steiner Tree of instance (V,T,w) with total cost at most $w(\mathcal{T}^*)-\sum_{S\in\mathcal{S}_g^e}\mathsf{adv}(S) = w(\mathcal{T}^*)-\sum_{i\text{ even}}A_i\leq(1-\varepsilon_0)\cdot\mathsf{MST}$. This shows that $\mathsf{ST}(V,T,w)\leq(1-\varepsilon_0)\cdot\mathsf{MST}$.

We now show that, if $\sum_{0\leq i\leq L}B_i\cdot|\hat{\mathcal{S}}_g^i|/(\log n/\varepsilon^{10}) > 5\varepsilon^{3/4}\cdot\mathsf{MST}$, then $\sum_{0\leq i\leq L}A_i > (2\varepsilon_0)\cdot\mathsf{MST}$ holds, completing the analysis of Step 3 when the output is $(1-\varepsilon_0)\cdot\mathsf{MST}$. We start with the following simple observation.

OBSERVATION 4.4. *For each good set $S\in\mathcal{S}_g^i$ and each set Y that represents S, there exists a set $\tilde{Y}\subseteq\tilde{S}$ that represents S, such that $\mathsf{adv}(S,Y)\leq\mathsf{adv}(S,\tilde{Y})+8\varepsilon\cdot(1+\varepsilon)^i$.*

Proof. Let $Y=\{u_{11},u_{12},u_{21},u_{22}\}$. Recall that $\tilde{S}$ is a maximal subset of S, such that the distance between every pair of terminals in $\tilde{S}$ is at least $\varepsilon\cdot(1+\varepsilon)^i$. So there exist vertices $\tilde{u}_{11}\in S_{11}$, such that $w(u_{11},\tilde{u}_{11})\leq\varepsilon\cdot(1+\varepsilon)^i$. Similarly, there exist vertices $\tilde{u}_{12}\in S_{12}$, $\tilde{u}_{21}\in S_{21}$, $\tilde{u}_{22}\in S_{22}$ that are close to u_{12},u_{21},u_{22}, respectively. We simply let $\tilde{Y}=\{\tilde{u}_{11},\tilde{u}_{12},\tilde{u}_{21},\tilde{u}_{22}\}$. Assume that the set Y achieves the cost $\mathsf{adv}(S,Y)$ via Steiner vertex v and tree $\mathcal{T}$. It is easy to observe that by replacing vertex u_{ij} with $\tilde{u}_{ij}$, we obtain another Steiner tree $\tilde{\mathcal{T}}$ that achieves advantage at least $\mathsf{adv}(S,Y)-8\varepsilon\cdot(1+\varepsilon)^i$. Observation 4.4 now follows. ∎

Consider now the collection $\hat{\mathcal{S}}_g^i$. Let $\mathcal{S}'\subseteq\hat{\mathcal{S}}_g^i$ contain all sets $S\in\hat{\mathcal{S}}_g^i$ with $\mathsf{adv}(S)\geq\varepsilon^{3/4}(1+\varepsilon)^i$, and let $\mathcal{S}''$ contain other sets. Since we have sampled $\log n/\varepsilon^{10}$ sets in $\hat{\mathcal{S}}_g^i$, from Chernoff Bound,

- if $|\mathcal{S}'| \geq \varepsilon \cdot |\hat{\mathcal{S}}_g^i|$, then with probability $(1 - n^{-10})$ the number of sampled sets in $\mathcal{S}'$ is within factor $(1+\varepsilon)$ from $(\log n/\varepsilon^{10})|\mathcal{S}'|/|\hat{\mathcal{S}}_g^i|$, so

$$\frac{|\hat{\mathcal{S}}_g^i|}{(\log n/\varepsilon^{10})} \cdot \sum_{S \text{ sampled}, S \in \mathcal{S}'} (\varepsilon^{3/4}/2) \cdot (1+\varepsilon)^i \leq (1+\varepsilon) \cdot \sum_{S \in \mathcal{S}'} \mathsf{adv}(S);$$

- if $|\mathcal{S}'| < \varepsilon \cdot |\hat{\mathcal{S}}_g^i|$, then then with probability $(1 - n^{-10})$, the number of sampled sets in $\mathcal{S}'$ is at most $10 \log n/\varepsilon^9$,

$$\frac{|\hat{\mathcal{S}}_g^i|}{(\log n/\varepsilon^{10})} \cdot \sum_{S \text{ sampled}, S \in \mathcal{S}'} (\varepsilon^{3/4}/2) \cdot (1+\varepsilon)^i \leq (10\varepsilon) \cdot (\varepsilon^{3/4}/2) \cdot (1+\varepsilon)^i \cdot |\hat{\mathcal{S}}_g^i| \leq \varepsilon \cdot w_i(\mathcal{T}^*).$$

On the other hand, it is clear that

$$\frac{|\hat{\mathcal{S}}_g^i|}{(\log n/\varepsilon^{10})} \cdot \sum_{S \text{ sampled}, \ S \in \mathcal{S}''} (\varepsilon^{3/4}/2) \cdot (1+\varepsilon)^i \leq \varepsilon^{3/4} \cdot w_i(\mathcal{T}^*).$$

Altogether, we get that, with probability $1 - O(n^{-10})$, $A_i \geq (B_i - \varepsilon^{3/4} \cdot w_i(\mathcal{T}^*))/2$. Taking the union bound over all $0 \leq i \leq L$, we get that, with probability $1 - O(n^{-9})$, $\sum_{0 \leq i \leq L} A_i \geq \sum_{0 \leq i \leq L} (B_i - \varepsilon^{3/4} \cdot w_i(\mathcal{T}^*))/2$. Therefore, if $\sum_{0 \leq i \leq L} B_i \geq 5\varepsilon^{3/4} \cdot \mathsf{MST}$, then $\sum_{0 \leq i \leq L} A_i \geq 2\varepsilon^{3/4} \cdot \mathsf{MST} \geq 2\varepsilon_0 \cdot \mathsf{MST}$. This completes the analysis of Step 3 when the output is $(1 - \varepsilon_0) \cdot \mathsf{MST}$.

Note that, if the algorithm did not return $(1 - \varepsilon_0) \cdot \mathsf{MST}$, then according to the algorithm, $\sum_{0 \leq i \leq L} B_i < 5\varepsilon^{3/4} \cdot \mathsf{MST}$. From the definition of A_i and B_i and similar arguments in the above analysis, we get that with high probability, $\sum_{0 \leq i \leq L} A_i < 5\varepsilon^{1/2} \cdot \mathsf{MST}$. Then from similar arguments in Observation 4.3, we can then show that $\sum_{S \in \mathcal{S}_g} w^*(S) \leq O(\mathsf{MST}/\log n)$, and $\sum_{S \in \mathcal{S}_g} \mathsf{adv}(S) \leq 6\varepsilon^{1/2} \cdot \mathsf{MST}$.

Analysis when the algorithm returns MST as the estimate of $\mathsf{ST}(V, T, w)$ Lastly, we show that, if the algorithm did not collect enough local evidence in Step 2 and Step 3, then $\mathsf{ST}(V, T, w)$ is indeed bounded away from $\mathsf{MST}/2$. Specifically, we show that, if the algorithm returns MST as the estimate, then $\mathsf{ST}(V, T, w) \geq \mathsf{MST}/(2 - 2\varepsilon_0)$, and so in this case the estimate is indeed a $(2 - 2\varepsilon_0)$-approximation of $\mathsf{ST}(V, T, w)$.

Before diving into the details, we give some intuition. Consider the optimal Steiner Tree $\mathcal{T}$, and we will iteratively remove Steiner vertices from it such that eventually it becomes a spanning tree over terminals. In each iteration, we will try to replace some set E of edges in the current tree with another set E' of terminal-terminal edges, such that $w(E') \leq (2 - \Omega(\varepsilon_0)) \cdot w(E)$ holds and the resulting graph is still a Steiner Tree. Intuitively, if we cannot find sufficient local evidence in Step 2, then most Steiner vertices in $\mathcal{T}$ can be eliminated such that the resulting tree $\mathcal{T}$ satisfies that $w(\mathcal{T}) \cdot (2 - \Omega(\varepsilon_0)) \leq \mathsf{MST}$. However, it is also possible that $\mathcal{T}$ behaves in a similar way as w_{Y} defined in Section 3 and we cannot find Steiner vertices to eliminate in $\mathcal{T}$. In this case, we will replace some set E of edges in the current tree with a set E' of terminal-terminal edges, such that $w(E') \leq 2 \cdot w(E)$ holds and simultaneously construct a set of four vertices that represents some set in $\mathcal{S}$ as defined in Step 3, and achieves cost reduction comparable to $w(E')$. Eventually, we will collect sufficient four-vertex sets, indicating that the local evidence that should have been found by the 4-vertex subroutine is large, contradicting the outcome of Step 3.

We now provide the complete proof. Let $\mathcal{T}_{\mathsf{OPT}}$ be an optimal solution of instance (V, T, w). We will iteratively modify tree $\mathcal{T}_{\mathsf{OPT}}$, such that eventually we obtain a spanning tree on T whose total weight is at most $(2 - 2\varepsilon_0)$ times the weight of $\mathcal{T}_{\mathsf{OPT}}$. Since such a tree has total weight at least MST, we get that $w(\mathcal{T}_{\mathsf{OPT}}) \geq \mathsf{MST}/(2 - 2\varepsilon_0)$.

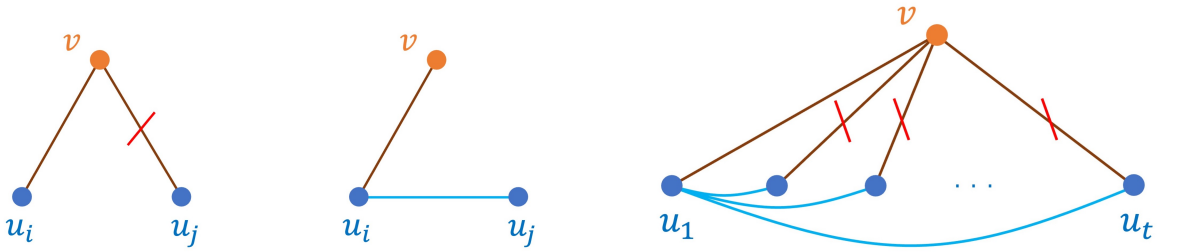
We now describe the tree-modification process. Throughout, we maintain a Steiner Tree $\mathcal{T}$ of instance (V, T, w), that is initialized to be $\mathcal{T}_{\mathsf{OPT}}$. Note that we can assume without loss of generality that $\mathcal{T}_{\mathsf{OPT}}$ does not contain degree-2 Steiner vertices (since such vertices can be suppressed without increasing the total weight of the tree), and whenever degree-2 Steiner vertices emerge in tree $\mathcal{T}$, we immediately suppress them. We will also maintain two collections $\mathcal{X}, \mathcal{Y}$ of sets of terminals in T, such that both $\mathcal{X}$ and $\mathcal{Y}$ initially contain no sets, and (i) every set added into $\mathcal{X}$ has size at least 3, and will be denoted by $X_i(v)$ for some integer $0 \leq i \leq L - 1$ and some

Steiner vertex v; and (ii) every set added into $\mathcal{Y}$ has size exactly 4. In each iteration, we distinguish between the following cases.

Case 1. $\mathcal{T}$ contains a Steiner vertex that is adjacent to at least three terminals. Let v be such a vertex. Let $u_1, \ldots, u_t$ be the terminals that are adjacent to v, such that distances $w(v,u_1) \leq w(v,u_2) \leq \cdots \leq w(v,u_t)$. Intuitively, if the distances differ significantly, then we can replace the heaviest edge with a terminal-terminal edge; if the distances are almost the same and close to half of terminal-terminal distances, then they should contribute a set to the Set Cover instance on this level (defined in Step 2); if the distances are almost the same and significantly greater than half of terminal-terminal distances, then they can all be replaced by terminal-terminal edges, with the total weight increasing by a factor at most $(2-\Omega(\varepsilon_0))$. Specifically, we distinguish between the following two cases.

Case 1.1. There exists a pair i,j of indices such that $w(u_i,u_j) \leq (2-4\varepsilon_0)\cdot w(v,u_j)$. In this case, we simply replace the edge (v,u_j) in $\mathcal{T}$ with edge (u_i,u_j). See Figure 2(a) for an illustration.

Case 1.2. For every pair i,j of indices, $w(u_i,u_j) > (2-4\varepsilon_0)\cdot w(v,u_j)$. Denote $\ell = w(u,v_1)$. Observe that, in this case, $w(v,u_t) \leq (1+5\varepsilon_0)\cdot\ell$ must hold, since otherwise $w(u_1,u_t) \leq w(v,u_1)+w(v,u_t) \leq (1+\frac{1}{1+5\varepsilon_0})\cdot w(v,u_t) \leq (2-4\varepsilon_0)\cdot w(v,u_t)$, a contradiction to the assumption in this case. Also observe that, for every pair i,j, $w(u_i,u_j) > (2-4\varepsilon_0)\cdot\ell$. We denote $i^* = \lfloor \log_{1+\varepsilon}\ell \rfloor$, and then define the set $X_{i^*}(v) = \{u_1,\ldots,u_t\}$ and add it into $\mathcal{X}$ (note that $t \geq 3$). We then replace, for each $2 \leq i \leq t$, edge (v,u_i) with edge (u_1,u_i). See Figure 2(b) for an illustration.



(a) Case 1.1: before (left) and after (right).

(b) Case 1.2: deleted edges (red) and new edges (blue).

Figure 2: An illustration of edge replacement in Case 1.

Assume now that Case 1 does not happen, so every Steiner vertex is adjacent to at most two terminals. We root tree $\mathcal{T}$ at an arbitrary Steiner vertex. Since $\mathcal{T}$ does not contain degree-2 Steiner vertices, every height-1 Steiner vertex is incident to exactly two terminals (since otherwise it is either a leaf or a degree-2 Steiner vertex, a contradiction).

Consider now any Steiner vertex v of height 2 in $\mathcal{T}$. Let $u'_1,\ldots,u'_p$ be the terminals that v is adjacent to, let $v_1,\ldots,v_t$ be the height-1 Steiner vertices adjacent to v, and for each $1 \leq j \leq t$, let u^j_1,u^j_2 be the two terminals adjacent to v_j, such that $w(v_j,u^j_1) \leq w(v_j,u^j_2)$. See Figure 3 for an illustration. Since v is not a degree-2 Steiner vertex in $\mathcal{T}$, either $t \geq 2$, or $t = 1$ and $p \geq 1$.

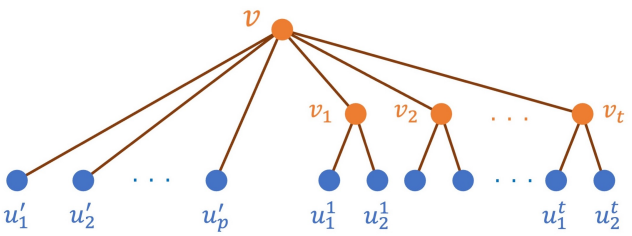


Figure 3: A schematic view of vertices and edges in Case 2.

Case 2. The tree-distances in $\mathcal{T}$ between v and terminals $u'_1, \ldots, u'_p, u^1_1, u^1_2, \ldots, u^t_1, u^t_2$ are not all within factor $(1 + O(\varepsilon_0))$. Intuitively, in this case the subtree of $\mathcal{T}$ rooted at v is not balanced enough, and so we can always find some terminal-Steiner edge to replace with a terminal-terminal edge. In particular, we distinguish between the following six cases.

Case 2.1. There exists a pair $1 \le i, j \le p$ such that $w(u'_i, u'_j) \le (2 - 4\varepsilon_0) \cdot w(v, u'_j)$. Similar to Case 1.1, we replace edge (v, u'_j) in $\mathcal{T}$ with edge (u'_i, u'_j) (see Figure 4(a)).

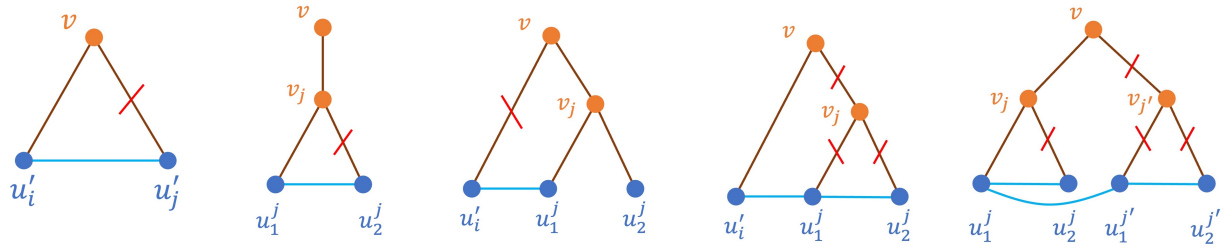
Case 2.2. There exists an index $1 \le j \le t$ such that $w(u^j_1, u^j_2) \le (2 - 4\varepsilon_0) \cdot w(v, u^j_2)$. We replace edge (v, u^j_2) in $\mathcal{T}$ with edge (u^j_1, u^j_2) (see Figure 4(b)).

Case 2.3. There exist $1 \le i \le p$, $1 \le j \le t$ and $z \in \{1, 2\}$, such that $w(u'_i, u^j_z) \le (2 - 4\varepsilon_0) \cdot w(v, u'_i)$. We replace edge (v, u'_i) in $\mathcal{T}$ with edge (u'_i, u^j_z) (see Figure 4(c)).

Case 2.4. There exist indices $1 \le i \le p$, $1 \le j \le t$ and $z \in \{1, 2\}$, such that $w(u'_i, u^j_z) \le (2 - 8\varepsilon_0) \cdot \big(w(v, v_j) + w(v_j, u^j_z)\big)$. We replace edges $(v, v_j), (v_j, u^j_1), (v_j, u^j_2)$ in $\mathcal{T}$ with edges (u^j_1, u^j_2) and (u'_i, u^j_z) (see Figure 4(d)).

Case 2.5. There exist $1 \le j, j' \le t$ and $z, z' \in \{1, 2\}$, such that $w(u^j_z, u^{j'}_{z'}) \le (2 - 8\varepsilon_0) \cdot \big(w(v, v_j) + w(v_j, u^j_z)\big)$. We replace edges $(v, v_j), (v_j, u^j_1), (v_j, u^j_2)$ in $\mathcal{T}$ with edges (u^j_1, u^j_2) and (u'_i, u^j_z), edge (v_j, u^j_2) with edge (u^j_1, u^j_2) (see Figure 4(e)).

Case 2.6. There exist $1 \le j \le t$ and $z \in \{1, 2\}$, such that $w(v, u^j_z) \le (1 - 4\varepsilon_0) \cdot \big(w(v, v_j) + w(v_j, u^j_z)\big)$. We replace edges $(v, v_j), (v_j, u^j_1), (v_j, u^j_2)$ in $\mathcal{T}$ with edges (u^j_1, u^j_2) and (v, u^j_z).



(a) Case 2.1. (b) Case 2.2. (c) Case 2.3. (d) Case 2.4. (e) Case 2.5.

Figure 4: An illustration of edge replacement in Case 2.

Assume that Case 1 and 2 do not happen. We denote $U = \{u'_1, \ldots, u'_p, u^1_1, u^1_2, \ldots, u^t_1, u^t_2\}$. From the discussion in Case 2, it is easy to observe that the tree-distances in $\mathcal{T}$ between v and terminals U are within factor $(1 + 20\varepsilon_0)$ from each other. Denote $\ell = \min\{w(v, u) \mid u \in U\}$. So for every terminal $u \in U$, $\ell \le w(v, u) \le (1 + 20\varepsilon_0) \cdot \ell$. We denote $i^* = \lfloor \log_{1+\varepsilon} \ell \rfloor$, and let $u^* = \arg\min_{u \in U} \{w(v, u) \mid u \in U\}$.

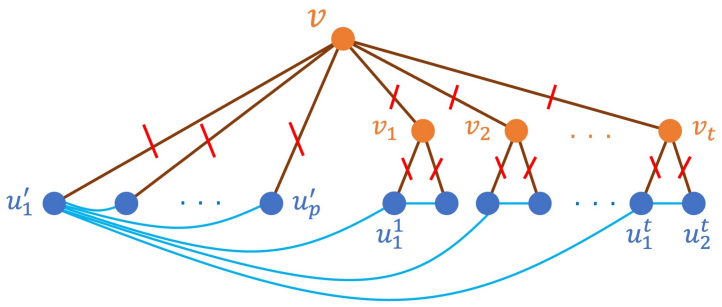


Figure 5: An illustration of edge replacement in Case 3.1 (assume that $u^* = u'_1$).

Case 3. We say Case 3 happens if one of the following subcases happen.

Case 3.1. $p + t \geq 3$. We define the set $X_{i^*}(v) = \{u'_1, \ldots, u'_p, u^1_1, \ldots, u^1_t\}$ and add it into $\mathcal{X}$. Then for each $1 \leq j \leq t$, we replace edge (v_j, u^j_2) with (u^j_1, u^j_2), and for each vertex $u \in U \setminus \{u^*, u^1_2, \ldots, u^t_2\}$, we replace edges in the v-u path in $\mathcal{T}$ with edge (u^*, u) (see Figure 5).

Case 3.2. $p = 1$ and $t = 1$. This case can be actually viewed as the special case of the next case by letting $v_2 = u^2_1 = u^2_2 = u'_1$.

Case 3.3. $p = 0$ and $t = 2$. We assume without loss of generality that $u^* = u^1_1$.

Case 3.3.1. We say that Case 3.3.1 happens if one of the following two cases happen:

- if $w(v, v_1) \leq (50\varepsilon_0) \cdot \ell$, then we define set $X_{i^*}(v) = \{u^1_1, u^1_2, u^2_1\}$ and add it into $\mathcal{X}$;
- if $w(v, v_2) \leq (50\varepsilon_0) \cdot \ell$, then we define set $X_{i^*}(v) = \{u^1_1, u^2_1, u^2_2\}$ and add it into $\mathcal{X}$.

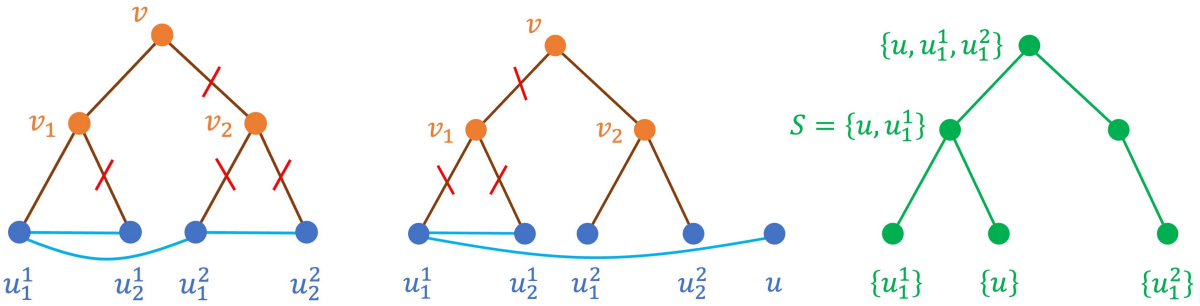
In addition, in the above cases, we replace edge (v_2, u^2_2) with (u^2_1, u^2_2), edges $(v_2, v), (v_2, u^2_1)$ with edge (u^2_1, u^1_1), and edge (v_1, u^1_2) with edge (u^1_1, u^1_2) (see Figure 6(a)).

Case 3.3.2. Consider now the laminar family $\mathcal{S}$ computed in Step 1. We say that a set U' of terminals in T is *interfered* iff there is a set S in $\mathcal{S}$, such that both $S \setminus U', U' \setminus S \neq \emptyset$, and in this case we say that any vertex $u \in S \setminus U'$ is a *witness*. If the pair u^1_1, u^1_2 of terminals are interfered, then let u be a witness, and assume without loss of generality that there exists a set $S \in \mathcal{S}$ that contains u^1_1 and u but not u^1_2 (the case where the pair u^2_1, u^2_2 of terminals are interfered is symmetric). If the set $\{u^1_1, u^1_2, u^2_1, u^2_2\}$ is interfered, then let u be a witness, and assume in particular that there exists a set $S \in \mathcal{S}$ that contains u^1_1, u^1_2 and u but not u^2_1, u^2_2. In both cases, we delete vertex v_1 and all its incident edges, and add edges (u^1_1, u^1_2) and (u^1_1, u) (see Figure 6(b)).

If Case 3.3.2 does not happen, then the collection $\mathcal{S}$ computed in Step 1 must contain sets $\{u^1_1, u^1_2\}$, $\{u^2_1, u^2_2\}$ and some set that contains all elements $u^1_1, u^1_2, u^2_1, u^2_2$. Since $\mathcal{S}$ is a laminar family, there exists a minimum set in $\mathcal{S}$ that contains all elements $u^1_1, u^1_2, u^2_1, u^2_2$, that we denote by S.

Case 3.3.3. $S = \{u^1_1, u^1_2, u^2_1, u^2_2\}$, and either $w(v, v_1) \leq (1 - 4\varepsilon^{1/4})\ell$ or $w(v, v_2) \leq (1 - 4\varepsilon^{1/4})\ell$ holds. Assume without loss of generality that $w(v, v_1) \leq (1 - 4\varepsilon^{1/4})\ell$. Then we add the set S into $\mathcal{Y}$, and then we replace edge (v_2, u^2_2) with (u^2_1, u^2_2), edges $(v_2, v), (v_2, u^2_1)$ with edge (u^2_1, u^1_1), and edge (v_1, u^1_2) with edge (u^1_1, u^1_2). The illustration figure in this case is identical to that of Case 3.3.1 (see Figure 6(a)).

Case 3.3.4. $w(v, v_1), w(v, v_2) > (1 - 4\varepsilon^{1/4}) \cdot \ell$. In this case we simply replace edge (v, u^1_2) with (u^1_1, u^1_2) and edge (v, u^2_2) with (u^2_1, u^2_2). We call the operation particularly in this case a *bad replacement*.



(a) Case 3.3.1.

(b) Case 3.3.2: old and new edges (left) and part of the tree $\mathcal{T}$ (right).

Figure 6: An illustration of edge replacement in Case 3.3.1 and Case 3.3.2.

The only possibility that Cases 2,3 do not happen is when:

- v has two children v_1, v_2; v_1 has two children u^1_1, u^1_2; and v_2 has two children u^2_1, u^2_2;
- $\ell \leq w(v, u^1_1), w(v, u^1_2), w(v, u^2_1), w(v, u^2_2) \leq (1 + 20\varepsilon_0)\ell$; and
- the set $\{u^1_1, u^1_2, u^2_1, u^2_2\}$ of vertices are not interfered in $\mathcal{S}$, but $S \neq \{u^1_1, u^1_2, u^2_1, u^2_2\}$.

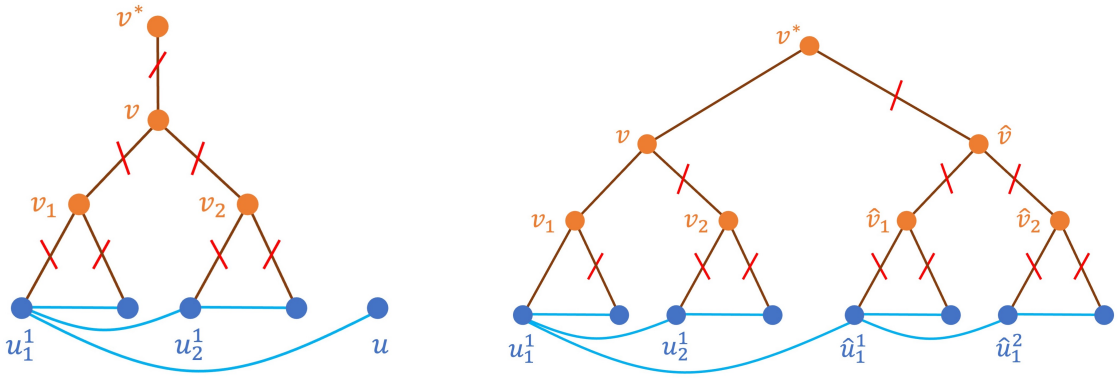
In this case, we say that v is an *unlucky* vertex. If there exists a height-2 non-unlucky vertex, then we process it using the operations described in Cases 2 and 3. We now consider the fourth and the last case, where all height-2 vertices in the current tree $\mathcal{T}$ are unlucky.

Case 4. All height-2 vertices in $\mathcal{T}$ are unlucky. If tree $\mathcal{T}$ does not contain any height-3 vertices, then $\mathcal{T}$ contains a unique height-2 vertex, but then this unique height-2 vertex is a degree-2 Steiner vertex in $\mathcal{T}$, a contradiction. Therefore, tree $\mathcal{T}$ contains height-3 vertices, and every height-3 vertex has at least two children. Consider now any height-3 vertex v^* and let $v, \hat{v}$ be two of its height-2 children. Since v is unlucky, we let the vertices $v_1, v_2, u_1^1, u_2^1, u_1^2, u_2^2$ be defined as before, and we define the vertices $\hat{v}_1, \hat{v}_2, \hat{u}_1^1, \hat{u}_2^1, \hat{u}_1^2, \hat{u}_2^2$ similarly for $\hat{v}$. We also define set S for v as before.

Recall that $S \neq \{u_1^1, u_2^1, u_1^2, u_2^2\}$. From the construction of laminar family $\mathcal{S}$, there exists a terminal u in $S \setminus \{u_1^1, u_2^1, u_1^2, u_2^2\}$, such that $\min\{w(u, u_1^1), w(u, u_2^1), w(u, u_1^2), w(u, u_2^2)\} \leq 2\ell \cdot (1+\varepsilon)$. Assume without loss of generality that $w(u, u_1^1) \leq 2\ell \cdot (1+\varepsilon)$.

Case 4.1. If $w(v, v^*) \geq 10\varepsilon \cdot \ell$ (the case where $w(\hat{v}, v^*) \geq 10\varepsilon \cdot \ell$ is symmetric), then we delete from $\mathcal{T}$ vertices v_1, v_2, v and all its incident edges, and add new edges $(u_1^1, u_2^1), (u_1^2, u_2^2), (u_1^1, u_1^2)$ and (u_1^1, u) (see Figure 7(a)).

Case 4.2. If $w(v, v^*), w(\hat{v}, v^*) \leq 10\varepsilon \cdot \ell$. Denote $\ell = w(v^*, u_1^1)$ and $\hat{\ell} = w(v^*, \hat{u}_1^1)$, and assume without loss of generality that $\ell \leq \hat{\ell}$. Then we delete all edges in the subtree rooted at $\mathcal{T}^*$ expect for the v^*-u_1^1 path, and add edges $(u_1^1, u_2^1), (u_1^2, u_2^2), (u_1^1, u_1^1)$, edges $(\hat{u}_1^1, \hat{u}_2^1), (\hat{u}_1^2, \hat{u}_2^2), (\hat{u}_1^1, \hat{u}_1^1)$ and edge $(u_1^1, \hat{u}_1^1)$ (see Figure 7(b)). If $\hat{\ell} \leq (1+10\varepsilon) \cdot \ell$, then we further add set $X_{i^*}(v^*) = \{u_1^1, u_1^2, \hat{u}_1^1, \hat{u}_1^2\}$ into collection $\mathcal{X}$, where $i^* = \lfloor \log_{1+\varepsilon} \ell \rfloor$.



(a) Case 4.1. (b) Case 4.2.

Figure 7: An illustration of edge replacement in Case 4.

This finally completes the description of the tree-modification process. Note that, in each of the cases described above, we replaced a set E of edges that do not connect a pair of terminals in the current tree $\mathcal{T}$ with a new set E' of edges that connect a pair of terminals, such that $(\mathcal{T} \setminus E) \cup E'$ is still a valid Steiner Tree, such that either

- $w(E') \leq (2 - 4\varepsilon_0) \cdot w(E)$; or
- $w(E') \leq 2 \cdot w(E)$, and we have added a set $X_i(v)$ into $\mathcal{X}$, such that $w(E) \leq 10 \cdot (1+\varepsilon)^i$; or
- $w(E') \leq 2 \cdot w(E)$, and we have added a set Y into $\mathcal{Y}$, such that $\mathsf{adv}(Y) \geq \varepsilon^{3/4} \cdot w(E)$; or
- $w(E') \leq 2 \cdot w(E)$, and we have not added sets into $\mathcal{X}$ or $\mathcal{Y}$, which may only happen in Case 3.3.4 where we performed a bad replacement.

First, it is easy to see that, the total cost of all edges where we perform bad replacements is at most $8 \cdot \varepsilon^{1/4} \cdot w(\mathcal{T}_{\mathsf{OPT}})$. Second, from the construction of the Set Cover instances $\{(\mathcal{W}_i, U_i)\}_{0 \leq i \leq L-1}$ in Step 2, and using similar arguments in the proof of Observation 4.4, it is easy to show that when we add a set $X_i(v)$ into collection $\mathcal{X}$, there is a set $W \in \mathcal{W}_i$, such that, for each $u \in X_i(v)$, there is a terminal $u' \in W$, such that $w(u, u') \leq \varepsilon \cdot (1+\varepsilon)^i$; and different sets $X_i(v)$ corresponds to different sets in $\mathcal{W}_i$. Therefore, according to the algorithm and the discussion above, the total weight of all edges in $\mathcal{T}^*$ in which we either perform a bad replacement or add a set into $\mathcal{X}$ or $\mathcal{Y}$ is at most

$$8 \cdot \varepsilon^{1/4} \cdot w(\mathcal{T}_{\mathsf{OPT}}) + \frac{6\varepsilon^{1/2} \cdot \mathsf{MST}}{\varepsilon^{1/4}} + 2^{20} \cdot \varepsilon_0 \cdot \mathsf{MST} \leq \frac{\mathsf{MST}}{3}.$$

Therefore, if we denote by $\mathcal{T}'$ the resulting tree we get from the above process, then

$$\mathsf{MST} \leq w(\mathcal{T}') \leq 2 \cdot \frac{w(\mathcal{T}_{\mathsf{OPT}})}{3} + (2 - 4\varepsilon_0) \cdot \frac{2 \cdot w(\mathcal{T}_{\mathsf{OPT}})}{3} \leq (2 - 2\varepsilon_0) \cdot w(\mathcal{T}_{\mathsf{OPT}}).$$

It follows that $w(\mathcal{T}_{\mathsf{OPT}}) \geq \mathsf{MST}/(2 - 2\varepsilon_0)$. This completes the proof of the correctness of the algorithm.

4.2 Proof of Theorem 4.1 In this section, we give a sublinear query algorithm for our Set Cover objective, and prove Theorem 4.1. Recall that we are given an instance $(U, \mathcal{W})$ of Set Cover, and the goal is to estimate the value of $\big(|U| - \mathsf{SC}(U, \mathcal{W}_{\neq 2})\big)$ where $\mathcal{W}_{\neq 2}$ is the collection of sets in $\mathcal{W}$ with size not equal to 2. We note that the goal of estimating the number of sets needed to cover a universe has been considered from the perspective of sublinear query algorithms [3, 4, 2]. However, these results do not apply to our setting as we need to estimate the difference between the universe size and the set cover size.

We first give an algorithm that outputs an estimate of $\big(|U| - \mathsf{SC}(U, \mathcal{W})\big)$, and then show how to modify the algorithm to prove Theorem 4.1.

An Algorithm for Estimating $\big(|U| - \mathsf{SC}(U, \mathcal{W})\big)$ The main result of this subsection is the following theorem.

THEOREM 4.2. *There is a polynomial-time randomized algorithm, that, given an instance $(U, \mathcal{W})$ of Set Cover and any constant $0 < \varepsilon < 1$, with probability $1 - O(n^{-2})$, returns a $(4, \varepsilon|U|)$-estimation of $\big(|U| - \mathsf{SC}(U, \mathcal{W})\big)$, by performing $O((|\mathcal{W}|^{3/2} + |\mathcal{W}|^{3/4}|U|) \cdot (\log n)^2/\varepsilon^3)$ membership queries to the instance $(U, \mathcal{W})$.*

Before we describe the algorithm in detail, we give a brief intuition. First we observe that, with high probability, all elements that appear in many (at least $\Omega(|\mathcal{W}| \log |\mathcal{W}|/\varepsilon|U|)$) sets in $\mathcal{W}$ can be covered by a random subset of $O(\varepsilon|U|)$ sets in $\mathcal{W}$, and so they can be ignored as we allow $\varepsilon|U|$ additive error in the estimation. Assume for simplicity that all elements that appear in fewer than $o(|\mathcal{W}| \log |\mathcal{W}|/\varepsilon|U|)$ sets in $\mathcal{W}$. Consider the optimal set cover $\mathcal{W}^*$ and assume the sets in $\mathcal{W}^*$ are arranged into a sequence. Now, if a set in $\mathcal{W}^*$ covers t elements that are not covered by the previous sets in the sequence, then it can be viewed as "contributing" $(t-1)$ to $\big(|U| - \mathsf{SC}(U, \mathcal{W})\big)$. We will show that, at a high-level, this "contribution" can be characterized as follows: if we consider the graph on U where there is an edge (u, u') iff u, u' appear in the same set in $\mathcal{W}$, then it can be shown that the size of any maximal matching in the graph is within a constant factor of $\big(|U| - \mathsf{SC}(U, \mathcal{W})\big)$. We then focus on this graph and utilize the algorithm from [12] to estimate the size of any maximal matching in it.

We now describe the algorithm in detail. For convenience, we denote $n = |\mathcal{W}|$ and $k = |U|$. Throughout this subsection, we use a parameter $\beta = \max\{k/n^{3/4}, 1\}$. In other words, when $k \geq n^{3/4}$, $\beta = k/n^{3/4}$, otherwise $\beta = 1$.

We define the *frequency* of an element $u \in U$ in the collection $\mathcal{W}$ to be the number of sets in $\mathcal{W}$ that contain u. We first partition the vertices into two subsets according to their frequency in $\mathcal{W}$ as follows. Let $\tilde{\mathcal{W}}$ be a random sub-collection of $\mathcal{W}$ that contains k/β sets. For every element $u \in U$, we compute the frequency u in $\tilde{\mathcal{W}}$ by performing membership queries on all pairs (u, W) with $W \in \tilde{\mathcal{W}}$. We then let U_{low} be the set of all elements in U with frequency at most $75\beta n \log n/(\varepsilon k)$ in $\tilde{\mathcal{W}}$, and let $U_{\text{high}} = U \setminus U_{\text{low}}$. The total number of queries that are needed to compute this partition is at most

$$k \cdot (k/\beta)) = O(k^2/\beta) = O\left(\frac{k^2}{\max\{k/n^{3/4}, 1\}}\right) = O\left(k^2 \cdot \min\left\{n^{3/4}/k, 1\right\}\right) = \underline{O(n^{3/2} + kn^{3/4})}.$$

Let $N = 50n\beta \log n/(\varepsilon k)$. We use the following observations that follow from the Chernoff Bound in Lemma 2.1.

OBSERVATION 4.5. *With probability $1 - n^{-2}$, all elements in U_{low} have frequency at most $2N$ in $\mathcal{W}$, and all elements in U_{high} have frequency at least N in $\mathcal{W}$.*

OBSERVATION 4.6. *Let $\hat{\mathcal{W}}$ be a random sub-collection of $\mathcal{W}$ that contains $\varepsilon k/(10\beta)$ sets. Then with probability $1 - n^{-2}$, every element in U_{high} is contained in some set in $\hat{\mathcal{W}}$.*

We now focus on the elements in U_{low}. We define $\mathcal{W}_{\text{low}} = \{W \cap U_{\text{low}} \mid W \in \mathcal{W}\}$. From Observation 4.5 with probability $1 - n^{-2}$, all elements in U_{low} have frequency at most $2N$ in $\mathcal{W}_{\text{low}}$. We define a graph H as follows. Its vertex set is U_{low}, and its edge set contains, for every pair $u, u' \in U_{\text{low}}$, an edge (u, u') iff there exists a set $W \in \mathcal{W}_{\text{low}}$ that contains both u and u'. We prove the following lemma.

LEMMA 4.1. *Let M be any maximal matching in H. Then*

$$\frac{|U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}})}{2} \leq |M| \leq |U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}).$$

Proof. On the one hand, we can construct a set cover of size at most $|U_{\text{low}}| - |M|$ as follows. For every pair $u, u' \in U_{\text{low}}$ that is matched in M, we take any set that contains both u, u'; for every $u'' \in U_{\text{low}}$ that is not matched in M, we take the set $\{u''\}$. Clearly, we have taken at most $|U_{\text{low}}| - |M|$ sets and they form a set cover, so $|M| \leq |U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}})$. On the other hand, consider an optimal set cover $\mathcal{F}$ with $|\mathcal{F}| = \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}})$. Note that each set $W \in \mathcal{F}$ contains at most one element that is not matched in M; otherwise there are $u, u' \in U_{\text{low}}$ that are both not matched in M while the edge (u, u') belongs to H by the definition of H, a contradiction to the maximality of M. Therefore, the number of vertices in H that are unmatched in M is at most $\mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}})$, which implies that $\frac{|U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}})}{2} \leq |M|$. ∎

We use the following result, which is implicit in Section 3 of [12].

THEOREM 4.3. ([12]) *Let G be a graph on Z vertices with average degree $\bar{d}$. If we are given an oracle that takes a vertex v of G as input and outputs all neighbors of v in G, then there is an algorithm, that, given any $0 < \varepsilon < 1$, with probability at least $1 - n^{-2}$ estimates the size of some maximal matching in G to within an additive factor of εZ, by performing $O(\bar{d} \log Z/\varepsilon^2)$ queries to the oracle.*

In order to use Theorem 4.3 to estimate the size of a maximal matching in H, we need to efficiently implement an oracle that, given any $u \in U$, finds all neighbors of u in H, which we do next.

A subroutine for finding all neighbors of a given vertex in H. From the definition of H, the neighbors of u are the elements u' in U_{low} that are contained in the same set as u. We find all neighbors of u in H as follows. We first perform membership queries on all pairs (u, W) with $W \in \mathcal{W}_{\text{low}}$ (a total of $O(n)$ queries), and find all sets in $\mathcal{W}_{\text{low}}$ that contains u. Then for each set $W \in \mathcal{W}_{\text{low}}$ that contains u, we perform membership queries on all pairs (u', W) with $u' \in U_{\text{low}}$ (a total of at most $2N \cdot k$ queries as u is contained in at most $2N$ sets), and figure out which elements are contained in W. The set of all neighbors of u in H is then obtained by taking the union of all sets W that contain u. In the whole subroutine, we have performed $2N \cdot k = O(\beta n \log n/\varepsilon)$ queries in total.

From Theorem 4.3 with the algorithm described above serving as the oracle, we can with high probability estimate the size of a maximal matching to within an additive factor of εk with $O(\bar{d} \cdot \beta n \log^2 n/\varepsilon^3)$ membership queries, where $\bar{d}$ is the average degree in H. However, $\bar{d}$ can be as large as k, in which case we can only get an $\tilde{O}(k \cdot n/\varepsilon^3)$ upper bound. To improve upon this, we will pre-process the instance $(U_{\text{low}}, \mathcal{W}_{\text{low}})$ before using the algorithm in Theorem 4.3.

We partition the collection $\mathcal{W}_{\text{low}}$ into $\mathcal{W}_1$ and $\mathcal{W}_2$ as follows. We use a parameter $Q = (k \log^{-2} n/\varepsilon n^{1/4})$. Let $\tilde{U}$ be a size-Q random subset of U_{low}. We let $\mathcal{W}_1$ contain all sets $W \in \mathcal{W}_{\text{low}}$ with $|W \cap \tilde{U}| \leq \log n$, and let $\mathcal{W}_2 = \mathcal{W}_{\text{low}} \setminus \mathcal{W}_1$. Using Chernoff bound and similar arguments in the proof of Observation 4.5, we can show that with probability $1 - n^{-2}$, every set in $\mathcal{W}_1$ contains at most $(100k \log n/Q)$ elements, and every set in $\mathcal{W}_2$ contains at least $(k \log n/100Q)$ elements. The number of queries that are needed for computing this partition is at most $(k \log^{-2} n/\varepsilon n^{1/4}) \cdot n \leq \tilde{O}(kn^{3/4}/\varepsilon)$.

We now consider the instances $(U_{\text{low}}, \mathcal{W}_1)$ and $(U_{\text{low}}, \mathcal{W}_2)$ separately. We define graphs H_1, H_2 for $(U_{\text{low}}, \mathcal{W}_1)$ and $(U_{\text{low}}, \mathcal{W}_2)$ respectively, in a similar way that H is defined for instance $(U_{\text{low}}, \mathcal{W}_{\text{low}})$. We use the following observation.

OBSERVATION 4.7. *Let M_1 be any maximal matching in H_1, and let M_2 be any maximal matching in H_2. Then there exists a maximal matching in H whose size is between $(|M_1| + |M_2|)/2$ and $|M_1| + |M_2|$.*

Proof. Note that $H = H_1 \cup H_2$. Assume without loss of generality that $|M_1| \geq |M_2|$. We construct a matching M in H as follows. We start with $M = M_1$. We add all edges of M_2 that do not share endpoint with any edges in M_1. Then we greedily add edges in H_2 that do not share endpoint with any of the current edges in M, until no edge can be added. Since M_1 is a maximal matching in H_1, from our algorithm, it is easy to verify that the resulting matching M is a maximal matching in H. Note that $|M| \geq |M_1| \geq (|M_1| + |M_2|)/2$.

It remains to show that $|M| \leq |M_1| + |M_2|$. We denote by M_2' the subset of edges in M_2 that do not share endpoints with edges in M_1. It suffices to show that in the last step we have greedily added at most $|M_2 \setminus M_2'|$

edges into M. Consider an edge e added to M in this step. Since $e \in E(H_2)$ and M_2 is a maximal matching in H_2, e shares an endpoint with some edge $e' \in M_2$, and e' has not been added to M in the previous step, so $e' \in M_2 \setminus M_2'$. We say that e' is the *blocker* of e. In fact, every edge $e' \in M_2 \setminus M_2'$ can be the blocker of at most one edge added in the last step, since if $e' = (u, u')$ and u is an endpoint of some edge in M_1, then every edge e blocked by e' has to contain u' as an endpoint, and thus there can be at most one such edge. ∎

We first consider the instance $(U_{\text{low}}, \mathcal{W}_1)$. Recall that each set in $\mathcal{W}_1$ has size at most $(100k \log n/Q)$, and every element in U_{low} is contained in at most $2N$ sets in $\mathcal{W}_1$. Therefore, every element in U_{low}, as a vertex in H_1, has at most $(100k \log n/Q) \cdot 2N = \tilde{O}(\beta n/Q)$ neighbors. We now apply the algorithm in Theorem 4.3 with parameter $\varepsilon/10$ together with the subroutine for finding all neighbors of a given vertex described above to obtain an estimate X_1 of the size of a maximal matching in H_1. Since the average degree in H_1 is $\tilde{O}(\beta n/Q)$, the number of queries performed by the algorithm is

$$O(\beta n \log n/\varepsilon) \cdot \tilde{O}(\beta n/Q) \cdot (\log n)/\varepsilon^2 = \tilde{O}(\beta^2 n^2/Q) = \tilde{O}\left(\frac{\left(\max\left\{k/n^{3/4}, 1\right\}\right)^2 \cdot n}{k/n^{1/4}}\right) = \underline{\tilde{O}(n^{3/2} + kn^{3/4})}.$$

We next consider the instance $(U_{\text{low}}, \mathcal{W}_2)$. Recall that every set in $\mathcal{W}_2$ has size at least $(k \log n/100Q)$ and every element is contained in at most $2N$ sets. Therefore,

$$|\mathcal{W}_2| \leq \frac{k \cdot 2N}{(k \log n/100Q)} = \frac{k \cdot 100\beta n \log n/(\varepsilon k)}{(k \log n/(100k \log^{-2} n/\varepsilon n^{1/4}))} = o\left(\max\left\{k, n^{3/4}\right\}\right).$$

Therefore, when $k > n^{3/4}$, we can simply use another $\tilde{O}(n)$ queries to estimate $|\bigcup_{W \in \mathcal{W}_2} W|$ to obtain an estimate of $\mathsf{SC}(U, \mathcal{W}_2)$ to within an additive εk factor. When $n^{3/4} \geq k$, we can then perform $k \cdot |\mathcal{W}_2| = \underline{O(kn^{3/4})}$ queries to obtain the entire instance $(U_{\text{low}}, \mathcal{W}_2)$, compute the graph H_2, and then compute the size $\underline{X_2}$ of a maximal matching in H_2.

Lastly, we return $X = |U_{\text{high}}| - \varepsilon k/10 + (X_1 + X_2)/2$ as our final output. From the above discussion, the number of queries we have performed is $\tilde{O}(n^{3/2} + kn^{3/4})$.

We next analyze the probability that X is a $(4, \varepsilon|U|)$-approximation of $|U| - \mathsf{SC}(U, \mathcal{W})$.

Bad Event ξ. We define ξ to be the bad event that either of the following happens: (i) there exists some element in U_{low} with frequency at more than $2N$ in $\mathcal{W}$ or there exists some element in U_{high} with frequency at most N in $\mathcal{W}$; (ii) there does not exist a subcollection of $\varepsilon k/(10\beta)$ sets in $\mathcal{W}$ that cover all elements in U_{high}; (iii) there exists some set in $\mathcal{W}_1$ that contains more than $(100k \log n/Q)$ elements, or there exists some set in $\mathcal{W}_2$ that contains fewer than $(k \log n/100Q)$ elements; and (iv) in the application of the algorithm from Theorem 4.3 to H_1, the output X_1 is not an estimate of the size of any maximal matching in H_1 to within an additive error of $\varepsilon k/10$. From Observation 4.5, Observation 4.6, Theorem 4.3 and the above discussion, $\Pr[\xi] = O(n^{-2})$.

The proof of Theorem 4.2 is concluded by the following claim.

CLAIM 4.2. *If event ξ does not happen, then X is a $(4, \varepsilon|U|)$-estimation of $|U| - \mathsf{SC}(U, \mathcal{W})$.*

Proof. Assume now that event ξ does not happen. Let M_1 be a maximal matching in H_1, such that $X_1 \leq |M_1| \leq X_1 + \varepsilon|U|/10$. Let M_2 be a maximal matching in H_2, such that $X_2 = |M_2|$. Let M be any maximal matching in H. From Observation 4.7,

$$\frac{X_1 + X_2}{2} \leq \frac{|M_1| + |M_2|}{2} \leq |M| \leq |M_1| + |M_2| \leq X_1 + X_2 + \frac{\varepsilon|U|}{10}.$$

Then from Lemma 4.1,

$$\frac{X_1 + X_2}{2} \leq |M| \leq |U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) \leq 2|M| \leq 2(X_1 + X_2) + \frac{\varepsilon|U|}{5}.$$

Let $\tilde{\mathcal{W}}$ be a sub-collection of $\varepsilon|U|/10$ sets in $\mathcal{W}$ that cover all elements in U_{high} (such a subcollection exists since event ξ does not happen). Let $\tilde{\mathcal{W}_{\text{low}}}$ be an optimal set cover of instance $(U_{\text{low}}, \mathcal{W}_{\text{low}})$. Clearly, $\tilde{\mathcal{W}} \cup \tilde{\mathcal{W}_{\text{low}}}$ is a feasible set cover of instance $(U, \mathcal{W})$. Therefore, $\mathsf{SC}(U, \mathcal{W}) \leq \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) + \varepsilon|U|/10$, and

$$\begin{aligned} X &= |U_{\text{high}}| - \frac{\varepsilon|U|}{10} + \frac{X_1 + X_2}{2} \leq |U_{\text{high}}| - \frac{\varepsilon|U|}{10} + |M| \leq |U_{\text{high}}| - \frac{\varepsilon|U|}{10} + |U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) \\ &= |U| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) - \frac{\varepsilon|U|}{10} \leq |U| - \mathsf{SC}(U, \mathcal{W}). \end{aligned}$$

On the other hand, since $\mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) \leq \mathsf{SC}(U, \mathcal{W})$,

$$|U| - \mathsf{SC}(U, \mathcal{W}) \leq |U| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) = |U_{\text{high}}| + |U_{\text{low}}| - \mathsf{SC}(U_{\text{low}}, \mathcal{W}_{\text{low}}) \leq |U_{\text{high}}| + 2|M|$$
$$\leq |U_{\text{high}}| + 2(X_1 + X_2) + \frac{\varepsilon|U|}{5} \leq 4 \cdot \left(|U_{\text{high}}| - \frac{\varepsilon|U|}{10} + \frac{X_1 + X_2}{2} \right) + \varepsilon|U| = 4X + \varepsilon|U|.$$

□

Handling the cardinality-2 sets We follow the same algorithm as decribed in Section 4.2. We first partition U into U_{low} and U_{high}, and then focus on instance $(U_{\text{low}}, \mathcal{W}_{\text{low}})$. We partition $\mathcal{W}_{\text{low}}$ into $\mathcal{W}_1$ and $\mathcal{W}_2$ and construct graphs H_1 and H_2. Note that when constructing these partitions, we do not (and are not able to) ignore the sets of size 2. Since every set in $\mathcal{W}_2$ contains more than 2 elements, computing an estimate of the size of a maximal matching in H_2 can be done in the same way. When processing the instance $(U_{\text{low}}, \mathcal{W}_1)$, in the subroutine of finding all neighbors of an element $u \in U_{\text{low}}$, we just need to ignore all sets of size 2 in $\mathcal{W}_1$ that contains u.

Regarding the proof that the output is a $(4, \varepsilon|U|)$-approximation of $|U| - \mathsf{SC}(U, \mathcal{W})$, Observation 4.6 shows that with probability $1 - n^{-2}$ there exists a subcollection $\tilde{\mathcal{W}}$ of at most $\varepsilon|U|/10$ sets in $\mathcal{W}$ that cover all elements in U_{high}. We need to modify it to show that with probability $1 - n^{-2}$ there exists a subcollection of $\varepsilon|U|/5$ sets in $\mathcal{W}_{\neq 2}$ that cover all elements in U_{high}, and this can be simply achieved by replacing all cardinality-2 sets in $\tilde{\mathcal{W}}$ with singleton sets that contain all elements that are covered by the cardinality-2 sets in $\tilde{\mathcal{W}}$. And the proof of Claim 4.2 now still goes through with this modification.

4.3 Implementation without Knowing the Terminal-Induced Metric Upfront

In this subsection, we complete the proof of Theorem 1.3 by showing that the algorithm described in Section 4.1 and Section 4.2 can in fact be implemented without knowing the terminal-induced metric upfront, at the cost of a slightly worse query complexity $\tilde{O}(n^{12/7} + n^{6/7} \cdot k)$.

Intuitively, since we do not have the terminal-induced metric upfront, we can no longer assume that we can start the process with a MST on terminals along with its hierarchical structure in our hand. However, as Step 2 and Step 3 in our algorithm only utilize local MST structure to find evidence, we do not really need the whole MST in order to implement them, but we can instead locally explore the MST structure (and its hierarchical structure) whenever needed. Therefore, we start by introducing two BFS-type subroutines that are crucial for implementing Steps 2 and 3. These subroutines are similar to the ones used in [24].

Auxiliary BFS-type Subroutines Since we are not able to query the distances between each pair of terminals, we are not able to precisely recover the graph H_i for each layer i. Thus, we will define a graph $\hat{H}_i$ which lies somewhere "in-between" H_i and H_{i+1} but makes it easier to explore locally.

Let $\varepsilon_1 = \varepsilon/10$. We first construct a graph $\bar{H}_i$ as follows: the vertices are terminals, and there is an edge between each pair of terminals if their distance is less than $\varepsilon_1(1+\varepsilon)^i$. We assign a random rank on each terminal, and let R_i be the lexicographically first maximal independent set based on this rank. We say the terminals in R are the *representative terminals*, and for each terminal u, there is a representative terminal u' at distance at most $\varepsilon_1(1+\varepsilon)^i$ away from u. We say *u is represented by u'*. We define $\hat{H}$ as follows: any terminal is connected to its representative terminal, and for any two representative terminals in R, they are connected in $\hat{H}$ if and only if their distance is at most $(1+3\varepsilon_1)(1+\varepsilon)^i$. The following observation directly follows from these definitions:

Observation 4.8. *Any component in H_i is also connected in $\hat{H}_i$, and any component in $\hat{H}_i$ is also connected in H_{i+1}.*

We define the *size of a component* in $\hat{H}$ as the number of representative terminals in $\hat{H}$. Next, we define another directed graph H_i^C as follows. The vertices of H_i^C are the components in $\hat{H}$, and for any two components S_1 and S_2, there is a directed edge from S_1 to S_2 if there is a representative terminal $u_1 \in S_1$ and a terminal $u_2 \in S_2$ such that their distance is at most $(1+3\varepsilon_1)(1+\varepsilon)^i$. We say that a *component S_1 can reach a component S_2* if there is a directed path from S_1 to S_2 in H_i^C. We have the following observation:

Observation 4.9. *If S_1 can reach S_2 in H_i^C, then S_1 and S_2 are inside the same component in H_{i+1}.*

We define U_i to be the components S in $\hat{H}_{i-1}$ such that the total size of the components that can be reached by S is at most $100L\log^2 n/\varepsilon_1$.

Throughout the process, we will always maintain the knowledge of whether a terminal is representative or not. At first, it is *unknown* for each terminal whether or not it is a representative. The subroutine $\mathsf{FIND}(u,i)$ takes as input a terminal u and a parameter i and outputs a representative u' in the following manner. We query the distance between u and all other terminals, if all terminals that are at most $\varepsilon_1(1+\varepsilon)^i$ away from u have lower rank than u or are known not to be a representative, then we output u. Otherwise let u_1 be the highest rank terminal among them, and we repeat this process on u_1, and continue in this manner, until we find a representative u'. We then mark u' as a representative, and all terminals that are at most $\varepsilon_1(1+\varepsilon)^i$ away from u' as non-representative. Note that u' might not represent u, but u and u' must be in in the same component in $\hat{H}_i$. The following observation gives an upper bound on the running time of FIND.

OBSERVATION 4.10. *With high probability, for any u and i, $\mathsf{FIND}(u,i)$ uses $\tilde{O}(k)$ queries.*

Proof. The exploration sequence is in fact a path in the DFS tree of the running of greedy parallel maximal independent set problem, and the depth of the DFS tree is $O(\log n)$ with high probability [1]. So with high probability, the length of the exploration sequence is $\tilde{O}(1)$ and the total number of queries we use is $\tilde{O}(k)$. ∎

Next, we give a BFS subroutine that takes as input a terminal u and an integer i. Intuitively, the subroutine explores the neighborhood of u in its level-i connected component up to a poly-logarithmic depth. Throughout, every terminal is either marked out, or in, or $\mathsf{representative}$, or active. Initially, terminal u is marked active and all other terminals are marked out. The procedure $\mathsf{BFS}(u,i)$ proceeds in rounds. In each round, we first pick an active terminal u' with the maximum rank (if there is no such terminal then the procedure is terminated). We run $\mathsf{FIND}(u',i)$ and let $\hat{u}$ be the output. We mark $\hat{u}$ as $\mathsf{representative}$. We then query the distance between $\hat{u}$ and all other terminals marked out, for each such terminal u'', if $w(\hat{u},u'') < \varepsilon_1(1+\varepsilon)^i$, then we mark u'' in. If $\varepsilon_1(1+\varepsilon)^i \le w(\hat{u},u'') < (1+3\varepsilon_1)(1+\varepsilon)^i$, we mark u'' active. This completes the description of a round. If the procedure did not terminate after $100L\log^2 n/\varepsilon_1$ rounds, then we artificially terminate the procedure and mark all active terminals in.

It is easy to observe that, after the procedure $\mathsf{BFS}(u,i)$ terminates, all terminals marked in or $\mathsf{representative}$ are certified to lie in the a component that is reachable from the component that contains u in $\hat{H}_i$ Clearly, if the procedure is not artificially terminated, then we have found all components in $\hat{H}_i$ that is reachable from the component containing u with all representative terminals inside it. So we can check if the component that contains u is inside U_{i+1} or not. If the procedure is artificially terminated, then u is contained in a component that is not in U_{i+1}. As the procedure $\mathsf{BFS}(u,i)$ runs for at most $100L\log^2 n/\varepsilon$ rounds and each round takes at most $\tilde{O}(k)$ queries, its query complexity is $\tilde{O}(k/\varepsilon_1)$.

We prove the following lemma.

LEMMA 4.2. *For any integers i and M, either (i) level i is light; or (ii) $|R_{i-1}| \le 2ML\log n/\varepsilon_1$; or (iii) $|U_i| \ge M$ holds.*

Proof. Assume that (i) and (ii) do not hold. We will show that $|U_i| \ge M$ must hold. Since $|R_{i-1}| \le 2ML\log n/\varepsilon_1$, the $2ML\log n/\varepsilon$ representative terminals are at distance at least $\varepsilon_1(1+\varepsilon)^{i-1}$ from each other, so $\mathsf{MST} \ge (2ML\log n/\varepsilon)\cdot\varepsilon(1+\varepsilon)^{i-1} = 2ML\log n(1+\varepsilon)^{i-1}$. On the other hand, since level i is not light, the total weight of all level-i edges in MST is at least $\mathsf{MST}/(L\log n) \ge 2M(1+\varepsilon)^{i-1}$. Note that $w_i(\mathcal{T}^*) \le |\mathcal{S}_i|(1+\varepsilon)^i$, so $|\mathcal{S}_i| \ge 2M/(1+\varepsilon)$. Lastly, from Observation 4.3, and the fact that any component of $\hat{H}_{i-1}$ that is not in $|U_i|$ is inside a large component in H_i by Observation 4.9, $|U_i| \ge \frac{2(1-O(1/\log n))M}{1+\varepsilon} > M$. ∎

We now proceed to describe the simulation of Steps 2 and 3 of our algorithm in the previous subsections. Note that if $k = O(n^{6/7})$, then we can simply perform $k^2 = O(n^{12/7})$ queries to obtain the terminal-induced metric and then perform Steps 2 and 3. The query complexity is $O(n^{12/7}) + \tilde{O}(n^{3/2} + n^{3/4}\cdot k) = \tilde{O}(n^{12/7} + n^{6/7}\cdot k)$. Therefore, we assume from now on that $k = \Omega(n^{6/7})$. [3]

[3] The main reason that we can run on graph $\hat{H}_i$ instead of H_i is the following: the algorithm we give in the previous section is in fact also true if we run layer i algorithms on graph H_{i+1} since the threshold we use for setting up the set cover instance, $3/5(1+\varepsilon)^i$ can be an arbitrary number between $1/2(1+\varepsilon)^i$ and $(1+\varepsilon)^i$, which means it also works for $3/5(1+\varepsilon)^{i+1}$. And since $\hat{H}_i$ is between H_i and H_{i+1} by Observation 4.8 and Observation 4.9, the analysis still works if we run on $\hat{H}_i$.

Simulation of Step 2 We now describe how to simulate Step 2 of the algorithm described in Section 4.1. Recall that, in Step 2, we have constructed, for each index i, an instance $(U_i, \mathcal{W}_i)$ of Set Cover for finding local evidence at level i, and designed an algorithm called AlgSetCover for estimating the value of $|U_i| - \mathsf{SC}(U_i, (\mathcal{W}_i)_{\neq 2})$. In particular, the sets in $\mathcal{W}_i$ correspond to Steiner nodes and the elements in U_i correspond to level-i connected components whose representative size is small (at most $L \log^2 n/\varepsilon$), and a set $W \in \mathcal{W}_i$ contains an element in U iff the Steiner node that W corresponds to is at distance at most $(3/5) \cdot (1+\varepsilon)^i$ from some representative of the component that the element in U corresponds to.

Fix an index i, and for convenience we denote $U = U_i$ and $\mathcal{W} = \mathcal{W}_i$. As we do not have the MST on terminals, we do not know the level-i connected components, which means we do not know the element in U but can only make queries to locally explore them. The simulation of Step 2 (and Step 3) finds the best tradeoff between the query complexity of this additional local exploration task and the previous algorithmic steps. In the remainder of this section, we use the parameter $M = n^{6/7} \log^2 n$.

We first find the first $(2ML \log n/\varepsilon_1)$ terminal in R_{i-1} by greedy MIS algorithm. The query complexity of this step is $\tilde{O}(Mk)$. If $|R_{i-1}| \leq (2ML \log n/\varepsilon)$, then we have already figured out all level-i connected components together with the representatives in each component, and therefore we can now run the algorithm AlgSetCover described before to obtain an estimate of the value of $|U| - \mathsf{SC}(U, \mathcal{W}_{\neq 2})$, whose query complexity is $\tilde{O}(n^{3/2} + n^{3/2} \cdot k)$. Assume from now on that $|R_{i-1}| > (2ML \log n/\varepsilon)$

From Lemma 4.2, either level i is light, or $|U| \geq M$. In order to determine which case happens, we will estimate the size of U. Specifically, we pick a random terminal u and run the procedure $\mathsf{BFS}(i-1, u)$. If the level-i connected component containing u is small, then we set $X(u)$ to be the inverse of the size of this component, otherwise we set $X(u) = 0$. It is easy to observe that X is a random variable supported on $[0, 1]$, and $\mathbb{E}[X] = |U|/k$. Therefore, from Chernoff Bound, if we repeat the process for $100k \log n/M$ random sampled terminals, then with probability $1 - n^{-10}$, we can estimate the value of $|U|$ to within an additive factor of $M/5$. Therefore, we can either correctly claim that $|U| < M$ or correctly claim that $|U| \geq M/2$. The query complexity is $\tilde{O}(k) \cdot (100k \log n/M) = \tilde{O}(k^2/M) = \underline{\tilde{O}(k \cdot n^{1/7})}$. If the claim is $|U| < M$, then from Lemma 4.2, level-i is light, and we will just ignore this layer by giving up local evidence on it. From now on we assume that $|U| > M/2$.

We now simulate the algorithm AlgSetCover in Section 4.2 with some modifications. We use another parameter $R = 50n^{1/7} \log n/\varepsilon$. First we partition the terminals into subsets T_{high} and T_{low} such that (i) for every $u \in T_{\text{high}}$, the number of Steiner nodes v at distance at most $(3/5) \cdot (1+\varepsilon)^i$ from u is at least R; and (ii) for every $u \in T_{\text{low}}$, the number of Steiner nodes v at distance at most $(3/5) \cdot (1+\varepsilon)^i$ from v is at most $2R$. Such a partition can be computed with $\underline{\tilde{O}(kn/R) = \tilde{O}(k \cdot n^{6/7})}$ queries. Note that, for each terminal in T_{high}, the element in U that corresponds to the level-i connected component that the terminal belongs to is contained in at least R sets. Since $|U| > M/2$, we can show via similar arguments that $\varepsilon|U|$ random sets will cover all these elements (as $M \cdot R > n/\varepsilon$). Therefore, we can ignore all level-i connected components that contain a terminal in T_{high}.

Next, we partition the Steiner vertices into subsets V_1 and V_2, using another parameter $P = n^{2/7}$ such that (i) for every vertex $v \in V_1$, the number of terminals in T_{low} at distance at most $(3/5) \cdot (1+\varepsilon)^i$ from v is at most $100P$; and (ii) for every vertex $v \in V_1$, the number of terminals in T_{low} at distance at most $(3/5) \cdot (1+\varepsilon)^i$ from v is at least P. Such a partition can be computed with $\tilde{O}(nk/P) = \underline{\tilde{O}(k \cdot n^{5/7})}$ queries. A similar argument shows

$$|V_2| \leq \frac{kR}{P} = \frac{k \cdot 50n^{1/7} \log n/\varepsilon}{n^{2/7}} = O(n^{6/7} \log n) = o(M) = o(|U|).$$

We now define U_{low} as the set of small level-i connected components that consist of only terminals in T_{low}. Let $\mathcal{W}_1, \mathcal{W}_2$ be the collections of sets naturally defined by Steiner nodes in V_1, V_2, respectively. We consider the set cover instances $(\mathcal{W}_1, U_{\text{low}})$ and $(\mathcal{W}_2, U_{\text{low}})$ separately.

We first estimate the value of $|U_{\text{low}}| - \mathsf{SC}(\mathcal{W}_2, U_{\text{low}})$. Since $|\mathcal{W}_2| = o(|U|)$, in order to obtain a $(2, \varepsilon|U|)$-estimate of $|U| - \mathsf{SC}(\mathcal{W}_2, U_{\text{low}})$, it is sufficient to estimate $|\bigcup_{W \in \mathcal{W}_2} W|$ to within an additive factor of $\varepsilon|U|$. This can be done in a similar way as estimating $|U|$. Specifically, we pick a random terminal $u \in U_{\text{low}}$ and run the procedure $\mathsf{BFS}(u, i)$. If the level-i connected component that contains u is small, then we set $X(u)$ to be the inverse of the size of this component; otherwise we set $X(u) = 0$. So the random variable X is supported on $[0, 1]$ and $\mathbb{E}[X] = |\bigcup_{W \in \mathcal{W}_2} W|/|T_{\text{low}}|$. From Chernoff bound, if we repeat the experiment for $100k \log n/(\varepsilon M)$ times, then we can obtain an estimate of $|\bigcup_{W \in \mathcal{W}_2} W|$ to within an additive factor of εM. The query complexity of this step is $\tilde{O}(k \cdot k/M) = \underline{\tilde{O}(k \cdot n^{1/7})}$.

We next estimate the value of $|U| - \mathsf{SC}(\mathcal{W}_1, U_{\text{low}})$. We proceed similarly as AlgSetCover, by defining an

auxiliary graph H on U and estimate its maximal matching size. Similar to Section 4.2, we only need to design a subroutine for finding all neighbors of a given element in H. This can be done as follows. Note that an element in H corresponds to a small level-i connected component. We first find all Steiner nodes in V_1 that is close to (at distance at most $(3/5)\cdot(1+\varepsilon)^i$ from) the component, and then find all terminals that are close to each of these Steiner nodes. Since any small components has at most $\tilde{O}(1)$ representatives, the number of such terminals is at most $R\cdot P=\tilde{O}((n^{1/7}/\varepsilon)\cdot n^{2/7})=\tilde{O}(n^{3/7}/\varepsilon)$. For each of these terminals, we run the procedure $\mathsf{BFS}(\cdot,i)$ on it to figure out if it indeed lies in a small component or not, and if the answer is yes, the component that contains the terminal is counted as a neighbor in H. The query complexity for all BFS procedures is $\tilde{O}(n^{3/7}\cdot k/\varepsilon)$. Lastly, via similar arguments, we can show that the query complexity of implementing the maximal matching estimation algorithm on H is $O(RP\cdot RPk)=\tilde{O}(n^{6/7}\cdot k)$.

We note that it is immediate to generalize above procedure to handle the cardinality-2 sets, since when estimating $|U|-\mathsf{SC}(\mathcal{W}_1,U_{\text{low}})$, we have figured out all elements in each explored set (and will be able to discard the set whenever it contains exactly two elements in U).

Simulation of Step 3 We now describe the simulation of Step 3, the four-vertex subroutine. Recall that, with the knowledge of terminal-induced metric, we constructed a laminar family and its partitioning tree $\mathcal{T}$ to represents its hierarchical structure, and we only aim to find a node $x_S\in V(\mathcal{T})$, such that x_S has exactly two children in $\mathcal{T}$ and each child of x_S also has exactly two children in $\mathcal{T}$. Consider such a node x_S at level i. Note that, if a child of x_S splits (into two child nodes) at a lower-than-$(i-\log_{1+\varepsilon}(1/\varepsilon_0))$ level, then the advantage obtained within this child node is at most $\varepsilon_0(1+\varepsilon)^i$, and it is safe to ignore it. Thus, we can run BFS on all terminals S for all levels between $i-\log_{1+\varepsilon}(1/\varepsilon_0)$ and i to figure out the hierarchical structure of S between these levels and calculate $\mathsf{adv}(S)$. The query time is $\tilde{O}(k)$.

The additional steps needed in simulating Step 3 are similar to that of Step 2, we first find the first $(2ML\log n/\varepsilon)$ terminals in R_{i-1}. If $|R_{i-1}|\le 2ML\log n/\varepsilon$, than we already figured out all level-i connected components, and we then simply proceed as before: sample $O(\log n/\varepsilon^{10})$ small components and calculate the advantage of them. Otherwise, similar to Step 2, we first estimates $|U|$. Either we correctly establish that $|U|<M$, in which case level i is light and can be safely ignored; or we correctly establish that $|U|>M/2$. Now we sample $O(k\log n/(\varepsilon^{10}M))$ terminals, and for each sampled terminal u, we first run $\mathsf{BFS}(u,i-1)$ to figure out the component S that contains u. If S is a small component, we calculate $\mathsf{adv}(S)$, and add it to B_i with probability $1/|S|$. The process is the same as sampling $\log n/\varepsilon^{10}$ small components in U and sum up the advantage of them. So B_i is a good approximation of A_i in this case as well. The total query complexity is $\tilde{O}(nk/M)=\tilde{O}(n^{1/7}k)$.

Altogether, the query complexity is $\tilde{O}(n^{12/7}+n^{6/7}\cdot k)$.

5 An $\tilde{\Omega}(nk)$ Lower Bound for $(2-\varepsilon)$-Approximate Steiner Tree

In this section, we provide the proof of Theorem 1.2 by showing that any randomized algorithm that computes a $(2-\varepsilon)$-approximate Steiner Tree performs at least $\Omega(nk)$ queries in the worst case. Throughout this section, we assume that $k\le n/100$.

We first construct a distribution on metric Steiner Tree instances (V,T,w) as follows. The vertex set V and the terminal set T are fixed (recall that $|V|=n$ and $|T|=k$). The terminal set is partitioned into $t=\left\lfloor\frac{k}{\lfloor 1/\varepsilon\rfloor}\right\rfloor$ sets $T=\bigcup_{1\le i\le t}T_i$, such that each set T_i contains either $\lfloor 1/\varepsilon\rfloor$ or $\lceil 1/\varepsilon\rceil$ terminals. This partitioning is fixed as well. The only randomized part is the weight-metric w. Let X be a set chosen uniformly at random from all size-t subsets of $V\setminus T$. We call vertices in X *crucial* vertices. We then choose a random one-to-one mapping from X to $[t]$, and for each $i\in[t]$, we call the vertex in X that is mapped to i the i-*crucial* vertex. The random metric w is defined according to the random set X as follows. For every $i\in[t]$, the weight between the i-crucial vertex in X and every terminal in T_i is 1, and the weight between any other pair of vertices in V is 2. It is easy to verify that w always satisfies the triangle inequality.

We call edges that are incident to crucial vertices *crucial edges*. Clearly, crucial edges form t disjoint stars. It is easy to verify that, although w is random, any two realizations of w are isomorphic, and so the metric Steiner Tree cost is always the same. In particular, the optimal Steiner tree contains all crucial edges and $(t-1)$ other edges connecting the star graphs formed by crucial edges, and so $\mathsf{ST}(V,T,w)=k\cdot 1+(t-1)\cdot 2=k+2t-2$. On the other hand, any Steiner tree that contains k' crucial edges has to contain at least $(t-1)+(k-k')$ other edges

in order to span all terminals, and its total cost is at least $k' \cdot 1 + \big((t-1)+(k-k')\big) \cdot 2 = 2k - k' + 2t - 2$. Therefore, in order to compute a Steiner tree of cost $(2-4\varepsilon) \cdot \mathsf{ST}(V, T, w)$, which is a Steiner tree of cost $(2-4\varepsilon)(k+2t-2)$, an algorithm has to finds at least $(2k+2t-2) - (2-4\varepsilon)(k+2t-2) = 4\varepsilon k - 2t \geq \varepsilon k$ edges.

Observe that, from the construction of w, if a terminal $u \in T_i$ has a weight-1 edge connecting to a Steiner vertex, then that Steiner vertex is the i-crucial vertex in X and every other terminal in T_i is also connected to it by weight-1 edges. For ease of analysis, we consider the following distribution of instances (V', T', w'). The terminal set T' contains, for each $i \in [t]$, a terminal $u_i \in T_i$, and it is fixed. The vertex set $V' = T' \cup (V \setminus T)$ and is also fixed. The metric w' is random and defined as follows. Let X be a set chosen uniformly at random from all size-t subsets of $V \setminus T$, and we then choose a random one-to-one mapping from X to $[t]$. We define i-crucial vertices similarly as before. For every $i \in [t]$, the weight between the i-crucial vertex in X and terminal u_i is 1, and the weight between any other pair of vertices in V' is 2. It is easy to observe that a size-t set X and a mapping from X to $[t]$ defines a metric (V, T, w) and a metric (V', T', w'), and a query in either metric can be simulated by a query in the other. Additionally, in order to find at least εk crucial edges in w, it is necessary to find $(\varepsilon k)/\lceil 1/\varepsilon \rceil \geq \varepsilon^2 k/2$ crucial edges in w'.

We say that an crucial edge (u, x) is *discovered* by an algorithm at some step iff the set of queries performed by the algorithm before this step uniquely identify the edge (u, x) to be a crucial edge. We say that a terminal is discovered iff its (unique) incident crucial edge is discovered, otherwise we say it is *undiscovered*. From Yao's minimax principle [7] and the above discussion, in order to prove Theorem 1.2, it suffices to prove the following lemma.

LEMMA 5.1. *Any deterministic algorithm that discovers in expectation at least $\varepsilon^2 k/2$ crucial edges in the distribution on w' defined above performs at least $\Omega(\varepsilon^2 nk)$ queries in expectation.*

In the remainder of this section, we provide the proof of Lemma 5.1. From now on, we only consider algorithms that perform queries to the metric w' instead of the metric w. We follow the framework used in the proof of Lemma 5.3 in [6], which shows that any randomized algorithm that outputs an approximate maximal matching performs at least $\Omega(n^2)$ queries in the worst case.

Consider a deterministic algorithm and the sequence of queries it performs. We partition the sequence into *phases* as follows. The first phase starts at the first query, a phase ends as soon as a crucial edge is discovered, and the next phase starts right after the previous phase ends. For each integer j, let Z_j be the random variable denoting the number of queries performed in the j-th phase. In order to prove Lemma 5.1, it suffices to show that $\mathbb{E}[\sum_{1 \leq j \leq \varepsilon^2 k} Z_i] = \sum_{1 \leq j \leq \varepsilon^2 k} \mathbb{E}[Z_i] = \Omega(\varepsilon^2 nk)$.

From the construction of w, the weight between any pair of Steiner vertices and the weight between any pair of terminals is always 2, so the only potentially useful queries are the ones between a terminal and a Steiner vertex. We define the *uncertainty* of a terminal $u \in T'$ at some step to be $(n-k)$ minus the number of queries involving u performed by the algorithm so far. We say that a phase is *bad*, iff at the start of the phase, there exists an undiscovered terminal whose uncertainty is below $3n/4$; and we say that a phase is *good*, iff at the start of the phase, the uncertainty of every undiscovered terminal is at least $3n/4$. The proof of Lemma 5.1 is concluded by the following claims.

CLAIM 5.1. *The expected number of queries in a good phase is at least $n/200$.*

Proof. We use the following lemma, which is similar to Lemma 5.4 in [6].

LEMMA 5.2. *Let $H = (A, B, E)$ be a bipartite graph with $|A| = a$ and $|B| = b$ (where $b > 10a$), such that every vertex in A has degree at least $2b/3$, then for every edge $e \in E$, the probability that e is contained in a uniformly at random chosen A-perfect matching in G is at most $2/b$ (here an A-perfect matching is a matching that matches all vertices of A).*

Proof. It is easy to verify from Hall's Theorem that H always contains a perfect matching. Consider an edge $(u, v) \in E$ where $u \in A$ and $v \in B$. Let M be an A-perfect matching that contains edge (u, v). We construct $b/2$ other A-perfect matchings as follows. Let B' be the set of vertices in $B \setminus \{v\}$ that is adjacent to u in H but is not an endpoint of any edge in M. Since $\deg_H(u) \geq 2b/3$ and $b > 10a$, we get that $|B'| \geq 2b/3 - 1 - b/10 \geq b/2$. For each vertex $v' \in B'$, we define $M_{v'}$ to be the matching obtained from M by replacing edge (u, v) with edge (u, v'). Clearly, $M_{v'}$ is an A-perfect matching for every $v \in B'$, so we obtained a collection of at least $b/2$ other A-perfect

matchings that do not contain edge (u, v), that we denote by $\mathcal{F}(M)$. On the other hand, for every pair M, M' of distinct A-perfect matchings in H that contains the edge (u, v), the collections $\mathcal{F}(M), \mathcal{F}(M')$ are disjoint. This is because for every matching $\hat{M} \in \mathcal{F}(M)$ and for every matching $\hat{M}' \in \mathcal{F}(M')$, $\hat{M}$ and $\hat{M}'$ must differ on an edge not incident to u, as M and M' do. Therefore, the number of perfect A-matchings in H is at least $(b/2)$ times the number of perfect A-matchings in H that contains the edge (u, v), and the lemma now follows. □

By definition, at the start of a good phase, the uncertainty of every undiscovered terminal is at least $3n/4$. Therefore, for the next $n/24$ queries, it is easy to verify that the graph induced by all unqueried edges satisfy the conditions of Lemma 5.2, and so the probability that any single query finds an edge of weight 1 is at most $2/(n-k) \leq 3/n$. Thus, with probability at least $(n/24) \cdot (3/n) = 1/8$, none of the first $n/24$ queries of a good phase finds a weight-1 edge. This implies that the expected number of queries in a good phase is at least $(1/8) \cdot (n/24) \geq n/200$. □

Claim 5.2. *If the algorithm performs less than $\varepsilon^2 nk/20$ queries, then the number of bad phases is at most $\varepsilon^2 k/4$.*

Proof. Since any useful query is between a terminal and a Steiner vertex, a useful query reduces the uncertainty of exactly one terminal by 1. Therefore, in order to have $\varepsilon^2 k/40$ bad phases, the reduction in the total uncertainty is at least $(\varepsilon^2 k/4) \cdot (n - k - 3n/4) \geq \varepsilon^2 nk/200$, which implies that the algorithm performs at least $\varepsilon^2 nk/200$ queries. □

From the above two claims, in order to find $\varepsilon^2 k/2$ crucial edges, either the algorithm performs at least $\varepsilon^2 nk/20$ queries, or the number of bad phases encountered by the query sequence is at most $\varepsilon^2 k/4$. Thus, in the first $\varepsilon^2 k/2$ phases, there are at least $\varepsilon^2 k/4$ good phases, implying that the expected number queries made by the algorithm is at least $(\varepsilon^2 k/4) \cdot (n/200) = \Omega(\varepsilon^2 nk)$. This completes the proof of Lemma 5.1, and therefore also completes the proof of Theorem 1.2.

6 Upper and Lower Bounds for α-Approximate Steiner Tree ($\alpha \geq 2$)

In this section we provide the proof of Theorem 1.5.

6.1 Upper Bound We start by presenting an $\tilde{O}(k^2/\alpha)$ query algorithm for any $\alpha \geq 2$. The query algorithm for any $\alpha \geq 2$ is very similar to the algorithm in the proof of Theorem 1 in [9], which shows a one-pass $\tilde{O}(n/\beta)$ streaming algorithm for estimating the metric MST cost to within factor β (for any $\beta > 1$). Note that we may assume without lose of generality that $\alpha = \Omega(\log^2 n)$, as otherwise we can simply query all terminal-terminal weight (which is $k^2 = \tilde{O}(k^2/\alpha)$ queries) and then compute the minimum spanning tree on T, which is a 2-approximate Steiner Tree (and therefore an α-approximate Steiner Tree as $\alpha \geq 2$).

Algorithm. Let $\beta = \alpha/(100 \log n)$. We first choose a uniformly at random size-$\lceil k/\beta \rceil$ subset of T, and denote it by T'. We then query all weights between pairs of terminals in T', and use the acquired information to compute the minimum spanning tree $\mathcal{T}'$ over T'. Lastly, for every terminal $u \notin T$, we query all weights between u and terminals in T', and let $f(u) = \arg_{u' \in T'} \min \{w(u, u')\}$. Finally, we output the tree $\mathcal{T}$ defined as $\mathcal{T} = \mathcal{T}' \cup \{(u, f(u)) \mid u \in T \setminus T'\}$.

On the one hand, observe that the algorithm only queries weights with one endpoint in T' and the other endpoint in T, so the number of queries performed by the algorithm is at most $k \cdot (k/\beta) = \tilde{O}(k^2/\alpha)$. On the other hand, it is easy to verify that the algorithm always outputs a spanning tree on T. The proof that the spanning tree $\mathcal{T}$ output by the algorithm above is with high probability an α-approximate Steiner Tree is similar to the analysis on pages 15-16 in [9], and is deferred to Appendix E.

6.2 Lower Bound We now prove an $\Omega(k^2/\alpha)$ lower bound for computing an α-approximate Steiner Tree for any $\alpha \geq 2$. In fact, we will show that, if all vertices in the metric space are terminals (so there are k terminals and no Steiner vertices), then for every $\alpha \geq 2$, computing a spanning tree of cost at most α times the minimum spanning tree cost requires at least $\Omega(k^2/\alpha)$ queries. Since the metric Steiner Tree cost is at most twice the minimum terminal spanning tree cost, this lower bound implies the lower bound in Theorem 1.5 (as we can construct a metric Steiner Tree instance, where all Steiner vertices are sufficiently far from all terminals and so any α-approximate Steiner tree may only terminal-terminal edges, and is therefore a terminal spanning tree).

It is easy to observe that it suffices to consider the case where $\alpha \geq 2$ is an integer and k is divisible by 100α. We generate a metric space w from the following distribution. The vertex set $V = T$ is fixed. Let $\mathcal{P}$ be a

partitioning of T, that is chosen uniformly at random from all partitioning of T that partition T into $t = k/100\alpha$ sets $T_1, \ldots, T_t$ of size 100α each. For every pair u, u' of terminals in the same part of $\mathcal{P}$, $w(u,u) = 1$, and for all other pairs u, u', $w(u,u) = 2\alpha$.

We call the weight-1 edges as *crucial* edges. It is easy to verify that, although metric w is random, any two realizations of w are isomorphic, and so the minimum spanning cost is always the same. In particular, the minimum spanning tree contains $(k-t)$ crucial edges and $(t-1)$ other edges connecting the trees incide each partition, and so $\mathsf{MST}(w) = (k-t)\cdot 1 + (t-1)\cdot 2\alpha = 1.02k - t - 2\alpha$. On the other hand, for any spanning tree that contains k' crucial edges and $(k-k'-1)$ other edges, its total cost is at least $k'\cdot 1 + (k-k'-1)\cdot 2\alpha = 2\alpha k - (2\alpha-1)k' - 2\alpha$. Therefore, in order for a spanning tree $\mathcal{T}$ to approximate the minimum spanning tree to within factor α, the tree $\mathcal{T}$ has to contain at least

$$\frac{2\alpha k - 2\alpha - \alpha\cdot(1.02k - t - 2\alpha)}{2\alpha - 1} = \frac{0.98\alpha k - 2\alpha + \alpha t + 2\alpha^2}{2\alpha} \geq 0.01k$$

crucial edges, and so there must be at least $0.01k$ vertices in T, such that $\mathcal{T}$ contains a crucial edge incident to it.

We say that an edge is *discovered* by an algorithm at some step iff the set of queries performed by the algorithm before this step uniquely identify the edge to be a weight-1 edge or a weight-2α edge. We say that a vertex u is *settled* iff the algorithm has discovered a crucial edge incident to u; otherwise we say that it is *unsettled*. Similarly, we say that a part T_i in the partitioning $\mathcal{P}$ is settled iff all its vertices are settled. From Yao's minimax principle [7] and the above discussion, in order to prove the lower bound of Theorem 1.5, it suffices to prove the following lemma.

LEMMA 6.1. *Any deterministic algorithm that settles at least* $0.01k$ *vertices in the distribution on* w *defined above performs at least* $\Omega(k^2/\alpha)$ *queries in expectation.*

In the remainder of this section, we provide a proof of Lemma 6.1. We use a similar approach as in Section 5. Consider a deterministic algorithm and the sequence of queries it performs. Assume without loss of generality that the algorithm terminates whenever it settles $0.01k$ vertices. We partition the sequence of queries it makes as follows. The first phase starts at the first query, a phase ends as soon as a previously unsettled vertex is settled, and the next phase starts right after the previous phase ends. For each $j \geq 1$, we let Z_j be the random variable denoting the number of queries performed in the j-th phase. In order to prove Lemma 6.1, it suffices to show that $\sum_{1\leq j\leq 0.01k} \mathbb{E}[Z_i] = \Omega(k^2/\alpha)$.

We classify all phases into good ones and bad ones as follows. We say that an unsettled vertex u is *well-discovered*, iff the number of parts T_i in $\mathcal{P}$ such that some edge in $E(u, T_i)$ has been discovered is at least $t/10$. We say that a phase is *type-1 bad*, iff at the start of this phase there exists an unsettled part $T_i \in \mathcal{P}$, and such that all discovered edges in $E(T_i, T\setminus T_i)$ touch at least $k/10$ vertices outside T_i. We say that a phase is *type-2 bad* iff at the start of this phase there exists a well-discovered vertex. If a phase is neither type-1 bad nor type-2 bad, then we say it is *good*. For ease of analysis, whenever the algorithm starts a type-1 bad phase due to some part T_i, we immediately reveal to the algorithm the weight of all edges with at least one endpoints in T_i, so the part T_i is settled right away; and whenever the algorithm starts a type-2 bad phase due to some well-discovered unsettled vertex u, we immediately reveal a crucial edge incident on u, so u and the other endpoint of the revealed edge are settled right away. We prove the following claims.

CLAIM 6.1. *The expected number of queries in a good phase is at least* $k/1600\alpha$.

Proof. Recall that the metric w is defined based on a partitioning $\mathcal{P}$ of T into $t = k/100\alpha$ subsets of size 100α each (that we call a *valid* partitioning). We say that a valid partitioning $\mathcal{P}$ *joins* a pair u, u' of vertices in T, iff u, u' lie in the same part of $\mathcal{P}$, otherwise we say that $\mathcal{P}$ *separates* the pair u, u'. We say that a valid partitioning $\mathcal{P}$ of T is *consistent* with the current queries, iff (i) every discovered crucial edge has both its endpoints lying in the same part of $\mathcal{P}$ and every discovered edge that is not crucial has its endpoints in different partitions; (ii) every unsettled vertex u is not well-discovered; and (iii) for every unsettled part T_i in $\mathcal{P}$, all discovered edges in $E(T_i, T\setminus T_i)$ touch at least $k/10$ vertices outside T_i. In other words, a partitioning $\mathcal{P}$ is consistent iff it provides consistent answers for all queries made by the algorithm, and the algorithm is not a bad phase given the current queries. Intuitively, from the algorithm's viewpoint, the up-to-date distribution (according to the answers to the queries performed so far) of the underlying partitioning should be the uniform distribution on all partitionings that are consistent with the current queries. We prove the following observation.

OBSERVATION 6.1. *At any time during the first $k/(10\alpha)$ queries of a good phase, for every pair u, u' of vertices in T such that u, u' are not both settled and the edge (u, u') has not been queried yet, the number of consistent partitionings that separate the pair u, u' is at least $k/800\alpha$ times the number of consistent partitionings that joins the pair u, u'.*

Proof. Consider a pair u, u' of vertices and a consistent partitioning $\mathcal{P}$ that joins the pair u, u'. Assume without loss of generality that u is unsettled. We construct a collection of other partitionings as follows. Let T_i be the part in $\mathcal{P}$ that contains u and u'. Let $T'(u)$ be the set of all unsettled terminals $\hat{u}$ such that (i) no edge in $E(\{\hat{u}\}, T_i)$ has been discovered; and (ii) no edge in $E(\{u\}, T_{i'})$ has been discovered, where $T_{i'}$ is the part in $\mathcal{P}$ that contains $\hat{u}$. We prove the following observation.

OBSERVATION 6.2. *At any time during the first $k/(10\alpha)$ queries of a good phase, for every unsettled vertex u, $|T'(u)| \geq k/2$.*

Proof. Before this good phase, u is also unsettled, so the number of vertices $\hat{u}$ such that $(u, \hat{u})$ has been discovered is at most $(t/10) \cdot (100\alpha) \leq k/10$. Let T_i be the part that u lies in, since T_i is not settled before this phase, it did not create a bad phase before, and so the number of vertices in $T \setminus T_i$ that are touched by discovered edges in $E(T_i, T \setminus T_i)$ is at most $k/10$. On the other hand, note that the algorithm settles at most 0.49 vertices before this phase, and has performed performed at most $k/(10\alpha)$ queries in this phase. By definition, $T'(u)$ contains all vertices $\hat{u}$ that does not satisfy any of the above conditions, so $|T'(u)| \geq k - k/10 - k/10 - 0.01k - k/(10\alpha) \geq k/2$. ∎

For each $\hat{u} \in T'(u)$, consider the partitioning $\mathcal{P}(\hat{u})$ obtained by exchanging the positions of vertices u and $\hat{u}$ (that is, if originally $u \in T_i$ and $\hat{u} \in T_{i'}$, then we move u to $T_{i'}$ and move $\hat{u}$ to T_i). Clearly, vertices u, u' are separated in $\mathcal{P}(\hat{u})$. We prove the following observation.

OBSERVATION 6.3. *For every $\hat{u} \in T'(u)$, $\mathcal{P}(\hat{u})$ is a consistent partitioning.*

Proof. Consider a vertex $\hat{u} \in T'(u)$. Let T_i be the set that contains u and let $T_{i'}$ be the set that contains $\hat{u}$. By definition of $T'(u)$, no edge in $E(\{\hat{u}\}, T_i)$ has been discovered, and no edge in $E(\{u\}, T_{i'})$ has been discovered. On the other hand, since u and $\hat{u}$ are not settled, no edge in $E(\{\hat{u}\}, T_{i'})$ has been discovered, and no edge in $E(\{u\}, T_i)$ has been discovered. Therefore, $\mathcal{P}(\hat{u})$ provides consistent answers with all queries made by the algorithm so far. On the other hand, it also implies that for every part T_j in $\mathcal{P}(\hat{u})$, all discovered edges in $E(T_j, T \setminus T_j)$ touch the same set of vertices as the corresponding part in $\mathcal{P}$. Lastly, it is also easy to verify that every unsettled vertex is still not discovered in $\mathcal{P}(\hat{u})$. Altogether, $\mathcal{P}(\hat{u})$ is a consistent partitioning with all current queries. ∎

For every $\hat{u} \in T'(u)$, we say that $\mathcal{P}(\hat{u})$ is a *host* of $\mathcal{P}$, so $\mathcal{P}$ has at least $k/2$ hosts. On the other hand, for every partitioning $\mathcal{P}'$ that separates u, u', there are at most 200α partitioning $\mathcal{P}$, such that $\mathcal{P}$ joins u, u' and $\mathcal{P}'$ is a host of $\mathcal{P}$ (since in $\mathcal{P}'$, u, u' belong to different parts, say $u \in T_i$ and $u' \in T_{i'}$ so it must be the case that either u and some vertex in $T_{i'}$ are exchanged or u' and some vertex in T_i are exchanged, a total of at most 200α possibilities). Therefore, the number of consistent partitionings that separate the pair u, u' is at least $(k/2)/(200\alpha) = k/400\alpha$ times the number of consistent partitionings that joins the pair u, u'. ∎

From Observation 6.1, among the first $k/800\alpha$ queries in a good phase, the probability that any single query finds a crucial edge is at most $400\alpha/n$, and so with probability $1/2$, none of them finds a crucial edge. This implies that the expected number of queries in a good phase is at least $(k/800\alpha) \cdot (1/2) = k/1600\alpha$. ∎

CLAIM 6.2. *If the algorithm performs less than $k^2/(10^7\alpha)$ queries, then the number of type-1 bad phases is at most $k/(10^5\alpha)$, and the number type-2 bad phases is at most $k/10^3$.*

Proof. Assume for contradiction that the number of type-1 bad phases is more than $k/(10^5\alpha)$. Then there are at least $k/(10^5\alpha)$ indices i, such that the number of edges in $|E(T_i, T \setminus T_i)|$ discovered by the algorithm is at least $k/10$. On the other hand since the algorithm never settles more than $0.01k$ vertices, it never discovers more than $0.01k$ crucial edges. Therefore, the number of queries performed by the algorithm is at least

$$\frac{1}{2} \cdot \frac{k}{10^5\alpha} \cdot \left(\frac{k}{10} - \frac{k}{10^5\alpha} \cdot 100\alpha - 0.01k \right) \geq \frac{k^2}{10^7\alpha},$$

a contradiction. Assume for contradiction that the number of type-2 bad phases is more than $k/10^3$, then there are more than $k/10^3$ well-discovered vertices. Therefore, the number of queries performed by the algorithm over all phases is at least $(k/10^3) \cdot (t/10 - k/(10^5\alpha)) \cdot (1/2) \geq k^2/(10^7\alpha)$, a contradiction. ∎

From the above two claims, we get that, in order to settle $0.01k$ vertices, either the algorithm performs at least $k^2/(10^7\alpha)$ queries, or the number of type-1 bad phases of its query sequence is at most $k/(10^5\alpha)$ (which settles at most $0.001k$ vertices in total) and the number of type-2 bad phases of its query sequence is at most $k/10^3$ (which settles at most $0.002k$ vertices in total), and so among the first $0.01k$ phases, there are at least $0.007k$ good phases, implying that the expected number queries made by the algorithm is at least $0.007k \cdot (k/1600\alpha) = \Omega(k^2/\alpha)$. This completes the proof of Lemma 6.1, and therefore also completes the proof of the lower bound in Theorem 1.5.

7 An $\tilde{\Omega}(n + k^{6/5})$ Query Lower Bound for $(2-\varepsilon)$-Estimation

In this section we provide the proof of Theorem 1.4, by showing that, for any $0 < \varepsilon < 1$, any randomized algorithm that with probability 2/3 estimates the Steiner Tree cost to within a factor of $2-\varepsilon$ performs $\Omega(n+k^{6/5})$ in the worst case. We will prove a lower bound of $\Omega(n)$ in Section 7.1 and a lower bound of $\Omega(k^{6/5})$ in Section 7.2. Combined together, they complete the proof of Theorem 1.4. Throughout the section, we assume that $1/(2\varepsilon) < k < n/2$.

7.1 An $\Omega(n)$ Lower Bound

We construct a distribution $\mathcal{D}$ of metric Steiner Tree instance (V, T, w) as follows. The set T contains k terminals and the vertices in $V \setminus T$ are denoted by $v_1, \ldots, v_{n-k}$. We define w_0 as the metric on V where $w_0(v, v') = 2$ for all pairs $v, v' \in V$. For each $1 \leq j \leq n-k$, we define a metric w_j as follows:

- For each terminal $u \in T$, $w_j(u, v_j) = 1$.
- For every other pair $v, v' \in V$, $w_j(v, v') = 2$.

In other words, the weight-1 edges in w_j form a star graph where v_j is its center and all terminals are its leaves. It is easy to verify that w_j is indeed a metric. We then define the instance $I_j = (V, T, w_j)$, and the distribution $\mathcal{D}$ is defined as follows: $\Pr[I_0] = 1/2$, and for each $1 \leq j \leq n-k$, $\Pr[I_j] = 1/(2(n-k))$.

Clearly, $\mathsf{ST}(I_0) = 2(k-1)$, and for each j, $\mathsf{ST}(I_j) = k$ as the star graph formed by all weight-1 edges is a Steiner Tree of cost k. Since $k > 1/(2\varepsilon)$, $\mathsf{ST}(I_0)/\mathsf{ST}(I_j) > 2-\varepsilon$ for all j, and so estimating the metric Steiner Tree cost of a random instance sampled from $\mathcal{D}$ to within factor $(2-\varepsilon)$ is equivalent to determining whether or not the random instance is I_0.

In order to correctly determining if a random instance sampled from $\mathcal{D}$ is I_0 with probability at least 2/3, it is necessary that the algorithm discovers a weight-1 edge on at least (1/3)-fraction of the instances $I_1, \ldots, I_{n-k}$. From Yao's minimax principle [7], the following claim implies a $\Omega(n-k) = \Omega(n)$ lower bound on the query complexity of any randomized algorithm, as $k \leq n/2$.

Claim 7.1. *Any deterministic algorithm that discovers a weight-1 edge on at least 1/3-fraction of the instances $I_1, \ldots, I_{n-k}$ performs at least $\Omega(n)$ queries in expectation.*

Proof. Observe that, in each of the instances $I_1, \ldots, I_{n-k}$, all weight-1 edges are incident to one Steiner vertex, that we call the *secret vertex*, and the secret vertex is $v_1, \ldots, v_{n-k}$ with probability $1/(n-k)$ each. Since a single query can only check one Steiner vertex (by querying any edge connecting it to a terminal), in order to find the secret vertex on at least 1/3-fraction of the instances $I_1, \ldots, I_{n-k}$, any algorithm needs to perform at least $(n-k)/3 = \Omega(n-k)$ queries in expectation. ∎

7.2 An $\Omega(k^{6/5})$ Lower Bound

We will construct a pair $\mathcal{D}_{\mathsf{Y}}, \mathcal{D}_{\mathsf{N}}$ of distributions on metric Steiner Tree instances, such that $\mathcal{D}_{\mathsf{Y}}$ is only supported on instances (V, T, w) with $\mathsf{ST}(V, T, w)$ close to Z, while $\mathcal{D}_{\mathsf{N}}$ is only supported on instances instances (V, T, w) with $\mathsf{ST}(V, T, w)$ close to $2Z$, where Z is some function of k that will be defined later. We let $\mathcal{D} = (\mathcal{D}_{\mathsf{Y}} + \mathcal{D}_{\mathsf{N}})/2$ be the average distribution of $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$. We show that, in order to report correctly with probability at least 2/3 whether an random instance sampled from $\mathcal{D}$ comes from $\mathcal{D}_{\mathsf{Y}}$ or $\mathcal{D}_{\mathsf{N}}$, any randomized algorithm has to perform at least $\tilde{\Omega}(k^{6/5})$ queries, thereby proving Theorem 1.4.

We now proceed to define the distributions $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$. We first define an *auxiliary instance* (V, T, w) as follows. For convenience, we let set T contain $k + k^{2/5}/\varepsilon$ terminals instead of k terminals. As we will see, this does not influence our lower bound, as $(k + k^{2/5}/\varepsilon)^{6/5} = \Theta(k^{6/5})$.

- The set T is partitioned into subsets $T = \left(\bigcup_{1\le j\le d} S_i\right) \cup \left(\bigcup_{1\le i\le d'} T_i\right)$, where $d = k^{2/5}$, $d' = k^{3/5}$; for each $1 \le j \le d$, $|S_j| = 1/\varepsilon$, and for each $1 \le i \le d'$, $|T_i| = k^{2/5}$.
  - We call sets $S_1, \ldots, S_d, T_1, \ldots, T_{d'}$ *groups*.
  - We denote $S = \bigcup_{1\le j\le d} S_j$, we call terminals in S *special* terminals, and we call terminals in $T \setminus S$ *regular* terminals. For each $1 \le j \le d$, we denote $S_j = \left\{s_{j,1}, \ldots, s_{j,1/\varepsilon}\right\}$. Note that $|S| = k^{2/5}/\varepsilon$.
- The set $V \setminus T$ contains $n - (k + k^{2/5}/\varepsilon)$ Steiner vertices and is further partitioned into $d' + 1$ subsets $V \setminus T = \bigcup_{0\le i\le d'} V_i$, where for each $1 \le i \le d'$, $|V_i| = k^{2/5}/\varepsilon$.
- The metric w is defined as follows:
  - For every pair u, u' of regular terminals that belong to the same group, $w(u, u') = 0$.
  - For each $1 \le i \le d'$, we fix an arbitrary perfect matching M_i between terminals in S and vertices in V_i. For every matched pair u, v', $w(u, v) = 1$.
  - The weight between every other pair of vertices in V is 2.

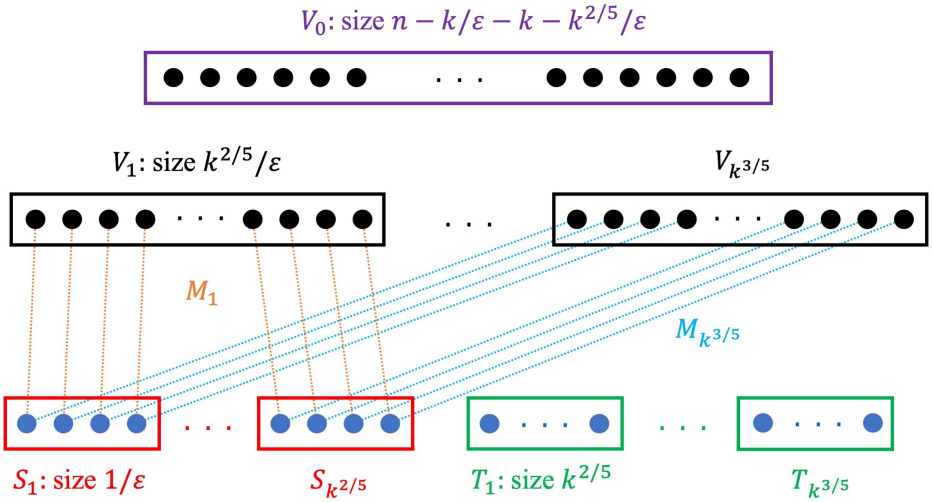


Figure 8: An illustration of the metric w. All terminals are shown in dark blue and all Steiner vertices are shown in black. Matchings $M_1, \ldots, M_{k^{3/5}}$ are shown in dashed lines and the matched pairs have weight 1. Pairs of teminals in the same T_i (green box) have weight 0. All other pairs have weight 2.

See Figure 8 for an illustration. It is easy to verify that w is indeed a metric.

We now use the auxiliary instance defined above to construct distributions $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$. Every instance with non-zero probability in either $\mathcal{D}_{\mathsf{Y}}$ or $\mathcal{D}_{\mathsf{N}}$ has the same vertex set $\hat{V}$ and the same terminal set $\hat{T}$, where $|\hat{V}| = |V|$ and $|\hat{T}| = |T|$. The set $\hat{V} \setminus \hat{T}$ of Steiner vertices is further partitioned into subsets $\hat{V} \setminus \hat{T} = \bigcup_{0\le i\le d'} \hat{V}_i$, where $V_0 = \hat{V}_0$, and for each $1 \le i \le d'$, $|\hat{V}_i| = |V_i|$. We say that a one-to-one mapping $f : \hat{V} \to V$ is *valid*, iff f maps terminals in $\hat{T}$ to terminals in T, f maps every vertex in $\hat{V}_0$ to itself in V_0, and for each $1 \le i \le d'$, f maps vertices in $\hat{V}_i$ to vertices in V_i. Let $\mathcal{F}$ be the set of all valid mappings.

We first define the distribution $\mathcal{D}_{\mathsf{N}}$. For each mapping $f \in \mathcal{F}$, we define an instance $I_f = (\hat{V}, \hat{T}, \hat{w}_f)$, where the metric $\hat{w}_f$ is defined as follows: for every pair $v, v' \in \hat{V}$, $\hat{w}_f(v, v') = w(f(v), f(v'))$. The distribution $\mathcal{D}_{\mathsf{N}}$ is simply defined to be the uniform distribution over all instances in $\mathcal{I}_{\mathsf{N}} = \{I_f \mid f \in \mathcal{F}\}$.

We now define the distribution $\mathcal{D}_{\mathsf{Y}}$. Consider a mapping $g : [d] \to [d']$. We first define another auxiliary metric w_g on V as follows.

- For each $1 \le j \le d$, we consider the matching $M_{g(j)}$ between S and $V_{g(j)}$. If we denote by $v^*_{g(j)}$ the Steiner vertex in $V_{g(j)}$ matched with terminal $s_{j,1}$, then for each $u \in S_j$, $w_g(u, v^*_{g(j)}) = 1$, and the weight in w_g between u and its matched Steiner vertex in $M_{g(j)}$ is 2.

- For every other pair $v, v' \in V$, $w_g(v, v') = w(v, v')$.

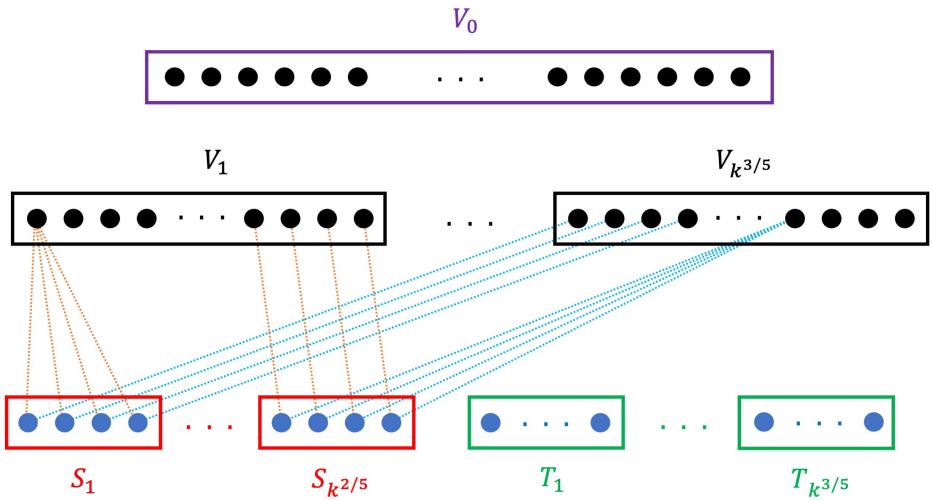


Figure 9: An illustration of the metric w_g (where $g(1) = 1$ and $g(k^{2/5}) = k^{3/5}$). The only difference between metrics w and w_g are the weight-1 pairs, which are shown in dashed lines.

See Figure 9 for an illustration. It is easy to verify that w_g is a metric. The only difference between metrics w and w_g are the weight-1 pairs. In particular, for each S_i, there is a star graph consisting of weight-1 edges in w_g that spans all terminals in S_i, while there is no such graph in w. These star graphs are the reason that the costs $\mathsf{ST}(V, T, w)$ and $\mathsf{ST}(V, T, w_g)$ are roughly separated by factor 2.

For every mapping $f \in \mathcal{F}$, we define a metric Steiner Tree instance $I_{(f,g)}$ as $I_{(f,g)} = (\hat{V}, \hat{T}, \hat{w}_{(f,g)})$ where the metric $\hat{w}_{(f,g)}$ is defined as: for every pair $v, v' \in \hat{V}$, $\hat{w}_{(f,g)}(v, v') = w_g(f(v), f(v'))$. The distribution $\mathcal{D}_{\mathsf{Y}}$ is simply defined to be the uniform distribution on all instances in $\mathcal{I}_{\mathsf{Y}} = \{I_{(f,g)} \mid f \in \mathcal{F}, g \in \mathcal{G}\}$, where $\mathcal{G}$ is the collection of all mappings from $[d]$ to $[d']$.

On the one hand, it is easy to verify that every instance with non-zero probability in $\mathcal{D}_{\mathsf{N}}$ is isomorphic to (V, T, w), and so it has the same Steiner Tree cost as (V, T, w). Similarly, it is easy to verify that every instance with non-zero probability in $\mathcal{D}_{\mathsf{Y}}$ is isomorphic to (V, T, w_g) for an arbitrary $g \in \mathcal{G}$ (and in fact all instances $\{(V, T, w_g)\}_{g \in \mathcal{G}}$ are isomorphic to each other), and so it has the same Steiner Tree cost as (V, T, w_g). We next show that $\mathsf{ST}(V, T, w)$ is roughly two times $\mathsf{ST}(V, T, w_g)$ for every g. Recall that $\mathcal{I}_{\mathsf{N}} = \{I_f \mid f \in \mathcal{F}\}$ and $\mathcal{I}_{\mathsf{Y}} = \{I_{(f,g)} \mid f \in \mathcal{F}, g \in \mathcal{G}\}$.

Claim 7.2. *Any instance in $\mathcal{I}_{\mathsf{N}}$ has Steiner Tree cost at least $2k^{2/5}/\varepsilon$. Any instance in $\mathcal{I}_{\mathsf{Y}}$ has Steiner Tree cost at most $k^{2/5}(1/\varepsilon + 4)$. Hence the cost of any instance in $\mathcal{I}_{\mathsf{N}}$ is more than $(2 - 8\varepsilon)$ times the cost of any instance in $\mathcal{I}_{\mathsf{Y}}$.*

Proof. For any instance in $\mathcal{I}_{\mathsf{N}}$, any Steiner node has weight 1 to at most 1 terminals, so the Steiner Tree cost equals the spanning tree of terminals, which is $2(k^{2/5} + k^{2/5}/\varepsilon - 1) > 2k^{2/5}/\varepsilon$. For any instance in $\mathcal{I}_{\mathsf{Y}}$, any group of special terminnal S_i have weight 1 to a common Steiner node, thus we can connect them at a cost of $1/\varepsilon$. Thus the Steiner Tree cost is at most $(1/\varepsilon)k^{2/5} + 2(k^{2/5} + k^{2/5} - 1) < (1 + 4\varepsilon)(k^{2/5}/\varepsilon)$. Moreover, it is easy to verify that the ratio of these costs is more than $(2 - 8\varepsilon)$ for any $\varepsilon > 0$. ∎

We then define distribution $\mathcal{D} = (\mathcal{D}_{\mathsf{Y}} + \mathcal{D}_{\mathsf{N}})/2$, and consider the following problem: Given an instance sampled from $\mathcal{D}$, estimate its value (Steiner Tree cost) to within a factor of $(2 - 8\varepsilon)$. From Claim 7.2, the problem is equivalent to the problem of determining a random instance (sampled from $\mathcal{D}$) comes from $\mathcal{D}_{\mathsf{Y}}$ or $\mathcal{D}_{\mathsf{N}}$. If a randomized algorithm reports correctly with probability at least 2/3, then we say that the algorithm *distinguishes between $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$*. Therefore, in order to prove Theorem 1.4, it suffices to prove that any algorithm that distinguishes between $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$ performs $\tilde{\Omega}(k^{6/5})$ queries in the worst case.

The remainder of this section is dedicated to the proof that any algorithm that distinguishes between $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$ performs $\tilde{\Omega}(k^{6/5})$ queries in the worst case. Before we give the detailed proof, we provide some intuition. From the construction of $\mathcal{D}_{\mathsf{Y}}$, $\mathcal{D}_{\mathsf{N}}$, distinguishing between $\mathcal{D}_{\mathsf{Y}}$ and $\mathcal{D}_{\mathsf{N}}$ is essentially distinguishing between the metric w and the metric w_g (for any g), where the identities of vertices are randomized. The main difference between metrics w and w_g is that, in w_g, there exist Steiner vertices that are connected to more than one (actually $1/\varepsilon$) special terminals with weight-1 edges. We call such Steiner vertices *secret vertices*. We now argue from a high level that finding a secret vertex requires $\Omega(k^{6/5})$ queries. We call edges of weight 0 or 1 *crucial* edges, since if a terminal u is found incident to a crucial edge, then we can immediately tell if u is special or regular.

On the one hand, note that in the metric w_g, there are $k^{2/5}$ secret vertices. So if we sample a random Steiner vertex v from $\bigcup_{1\le i\le d'} V_i$ (as the vertices in V_0 are irrelevant in distinguishing between metrics w and w_g), then the probability that v is a secret vertex is $O(k^{-3/5})$ as $|\bigcup_{1\le i\le d'} V_i| = \Omega(k)$, and so it takes $\Omega(k^{3/5})$ random samples to get a secret vertex. However, in order to certify that v is indeed a secret vertex, we need to find at least 2 crucial edges incident to it, which takes $\Omega(k)$ queries as every Steiner vertex is only incident to $O(1)$ crucial edges both in w and w_g. Altogether, it takes $\Omega(k^{3/5}) \cdot \Omega(k) = \Omega(k^{8/5})$ queries to discover a secret vertex in this way.

On the other hand, note that there are $k^{2/5}$ special terminals (in both w and w_g). So if we sample a random terminal u from T, then the probability that u is a special terminal is $O(k^{-3/5})$ as $|T| = \Omega(k)$, and so it takes $\Omega(k^{3/5})$ random samples to get a special terminal. In order to certify that u is indeed a special terminal, we need to find a crucial edge incident to it. Since each special terminal is only incident to $k^{3/5}$ crucial edges (in both w and w_g), this takes $\Omega(n/k^{3/5}) = \Omega(k^{2/5})$ queries. Altogether, it takes $\Omega(k^{3/5}) \cdot \Omega(k^{2/5}) = \Omega(k)$ queries to discover a special terminal. If the algorithm performs $o(k^{6/5})$ queries, it is only able to discover $o(k^{1/5})$ special vertices. As the identities of the terminals are randomized, from the Birthday Paradox, with high probability all discovered special vertices come from different groups in $S_1, \ldots, S_d$, so even if the algorithm has queried all edges incident to these discovered terminals, with high probability it will not find any vertex that is incident to more than one discovered special terminal, and so with high probability it will not find any secret vertex.

In the remainder of the section, we formalize the ideas described above, in a way similar to Section 5 and Section 6. We begin with some definitions. For convenience, we will think of the algorithm as performing queries on w, but it does not know the identities of the vertices (that is, it does know the set T and the partition $(V_0, V_1, \ldots, V_{d'})$ of $V \setminus T$, but it does not know which special terminal in set S_j is $s_{j,i}$, for any j, i). Over the course of the algorithm, we say a terminal u is *settled* at some step, iff the set of queries performed thus far uniquely identify u to be a special terminal or a regular terminal, i.e., we have discovered a crucial edge incident to it; otherwise we say it is *unsettled*. For convenience of the analysis, whenever a terminal u is settled, if u is a regular terminal, then we immediately reveal to the algorithm which T_i group it belongs to; if it is a special terminal, we will immediately reveal to the algorithm all crucial (weight-1) edges incident to it.

For each special terminal $u \in S$, we denote by $V(u)$ be set of Steiner nodes that are connected to u by some matching in $\{M_1, \ldots, M_{d'}\}$. Clearly, in w, vertex u is connected to all vertices in $V(u)$ via crucial (weight-1) edges, while in w_g, this is not always the case. We say that an edge is *discovered* if we can uniquely identify the weight of the edge. For any $0 < \alpha < 1$, we say an unsettled terminal u is *α-well-discovered* if at least one of the following four conditions is satisfied:

P1. There are at least $\alpha k^{2/5}$ groups T_i such that we have discovered at least one edge in $E(u, T_i)$.

P2. There are at least $\alpha k^{2/5}$ special terminals u' such that we discovered an edge in $E(u, V(u'))$.

P3. u is a regular terminal in the T_i, and there are at least αk terminals u' such that some edge in $E(u', T_i)$ has been discovered.

P4. u is a special terminal, and there are at least αk terminals u' such that some edge in $E(u', V(u))$ has been discovered.

The following lemma is the main technical tool for the proof of the $\Omega(k^{6/5})$ lower bound. Intuitively, it shows that, if the algorithm perform $o(k^{6/5})$ queries, then not only it settles or $(\varepsilon/200)$-well-discovers very few terminals, but also it has very limited knowledge upon the edges that it has not queried.

LEMMA 7.1. *Let* Alg *be any deterministic algorithm that performs at most $\varepsilon^2 k^{6/5}/10^9$ queries. Then:*

- *If the input to* Alg *is a random instance from* $\mathcal{D}_{\mathsf{N}}$*, then with probability at least* $9/10$*, the number of terminals that are either settled or* $(\varepsilon/200)$*-well-discovered is at most* $\varepsilon k^{4/5}/10^4$*, and at most* $\varepsilon k^{1/5}/10^4$ *of them are special terminals; and conditioned on the query sequence and its answers over the course of the algorithm, for every terminal that is neither settled nor* $(\varepsilon/200)$*-well-discovered, the probability that it is a special terminal is at most* $2/k^{3/5}$*.*

- *If the input to* Alg *is a random instance from* $\mathcal{D}_{\mathsf{Y}}$*, then with probability at least* $9/10$*,*
  - *the number of terminals that are either settled or* $(\varepsilon/200)$*-well-discovered is at most* $\varepsilon k^{4/5}/10^4$*;*
  - *at most* $\varepsilon k^{1/5}/10^4$ *of them are special terminals; and*
  - *all these special terminals belong to different* S_i *groups.*

  Moreover, conditioned on the query sequence and its answers over the course of the algorithm, for every unqueried edge between a Steiner vertex and a not-$(\varepsilon/100)$*-well-discovered terminal, the probability that the edge is a crucial (weight-*1*) edge and the Steiner vertex is a secret vertex is at most* $2/k^{8/5}$*.*

We provide the proof of Lemma 7.1 in Section 7.2.2 and Section 7.2.3, after we complete the proof of the $\Omega(k^{6/5})$ lower bound using it.

7.2.1 Completing the Proof of the $\Omega(k^{6/5})$ Lower Bound Recall that $\mathcal{D} = (\mathcal{D}_{\mathsf{Y}} + \mathcal{D}_{\mathsf{N}})/2$. In this subsection we use Lemma 7.1 to prove the following lemma.

LEMMA 7.2. *Any randomized algorithm that, given a random instance sampled from* $\mathcal{D}$*, reports correctly with probability at least* $2/3$ *that the instance comes from* $\mathcal{D}_{\mathsf{Y}}$ *or* $\mathcal{D}_{\mathsf{N}}$*, performs at least* $\varepsilon^2 k^{6/5}/10^9$ *queries in the worst case.*

From Yao's minimax principle [7], it suffices to consider only deterministic algorithms that report correctly on at least 2/3-fraction (in $\mathcal{D}$) of the instances.

We define a *transcript* to be the union of a query sequence and all its answers. We can define α-well-discovered vertices and settled terminals with respect to a transcript similarly.

Recall that each instance in $\mathcal{I}_{\mathsf{N}}$ is determined by a mapping $f \in \mathcal{F}$ and each instance in $\mathcal{I}_{\mathsf{Y}}$ is determined by a mapping $f \in \mathcal{F}$ and a mapping $g \in \mathcal{G}$, so $|\mathcal{I}_{\mathsf{Y}}| = |\mathcal{I}_{\mathsf{N}}| \cdot |\mathcal{G}|$. Let σ be the transcript produced by the algorithm when given a random instance from $\mathcal{D}$. Since the input is randomized, σ is a random variable. We say that σ is *consistent* with an instance I iff all answers to the queries in σ are matched with the corresponding weight-values in I. In order to prove Lemma 7.2, it suffices to show that, if σ always contains at most $\varepsilon^2 k^{6/5}/10^9$ queries, then with high probability, the ratio between the number of instances in $\mathcal{I}_{\mathsf{Y}}$ that are consistent with σ (that we call *consistent instances in* $\mathcal{I}_{\mathsf{Y}}$) and the number of instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ (that we call *consistent instances in* $\mathcal{I}_{\mathsf{N}}$) is still roughly $|\mathcal{G}|$. We prove this by showing that (i) for each consistent instance $I_f \in \mathcal{I}_{\mathsf{N}}$, almost all mappings $g \in \mathcal{G}$ give a consistent instance $I_{(f,g)}$; and (ii) for almost all consistent instances $I_{(f,g)} \in \mathcal{I}_{\mathsf{Y}}$, the instance $I_f \in \mathcal{I}_{\mathsf{N}}$ is also consistent.

We let T^* be the subset of terminals that are either settled or $(\varepsilon/200)$-well-discovered by σ (so T^* is a random variable as well). We will focus on queries incident to vertices in T^* and $S \setminus T^*$.

From Lemma 7.1, there are at most 1/10-fraction of the consistent instances in $\mathcal{I}_{\mathsf{N}}$ such that the desired properties (the number of settled or $(\varepsilon/100)$-well-discovered terminals is low, etc) do not hold; and there are at most 1/10-fraction of the consistent instances in $\mathcal{I}_{\mathsf{Y}}$ such that the desired properties do not hold. We call these instances *bad* instances, and call all other consistent instances *good* instances.

Consider now any mapping $g \in \mathcal{G}$, we define the following instance $I'_{(f,g)} \in \mathcal{I}_{\mathsf{Y}}$ defined by g and the T^* which is slightly different from $I_{(f,g)}$. For each $1 \le j \le d$, we consider the matching $M_{g(j)}$ between S_j and $V_{g(j)}$. If S_j contains more than 1 terminals in T^*, then $I_{(f,g)}$ is not well defined. If S_j contains one such terminal and assume it is s^*_j, then we first exchange the matching node of s^*_j and $s_{j,1}$ in $M_{g(j)}$. If S_j contains no such terminal, then we do not change the matching. After modifying the matching, for each $u \in S_j$, we make $w_g(u, s^*_{g(j)}) = 1$ and the weight in w_g between u and its matched Steiner vertex in $M_{g(j)}$ is 2. Note that after the change, we guarantee that we do not change the crucial edges incident on any terminal in T^*.

From consistent instances in $\mathcal{I}_{\mathsf{N}}$ to consistent instances in $\mathcal{I}_{\mathsf{Y}}$. From Lemma 7.1, for every terminal $u \in T \setminus T^*$, the probability (conditioned on σ and its answers) that u is special terminal is at most $2/k^{3/5}$. Then from Markov's Bound, with probability at least 0.98 (i.e., on at least 0.98-fraction of the good instances in $\mathcal{I}_{\mathsf{N}}$), the number of queries performed on edges in $E(S \setminus T^*, V \setminus T)$ is at most $\varepsilon k^{3/5}/100$. We now show that, for such a good instance $I_f \in \mathcal{I}_{\mathsf{N}}$, almost all mappings g give consistent instance $I'_{(f,g)} \in \mathcal{I}_{\mathsf{Y}}$.

Now for a random mapping $g \in \mathcal{G}$. If we view the algorithm as performing queries on w_g (without knowing the identities of the vertices), then from the perspective of the algorithm, the partitioning of the special terminals into groups $S_1, \ldots, S_{k^{2/5}}$ is random. Since T^* contains at most $\varepsilon k^{1/5}/10^4$ special terminals, with probability at least $1/10^4$, $|S_j \cap T^*| \leq 1$ for all $1 \leq j \leq d'$ hold. Moreover, if z queries are incident to terminals in $S_j \setminus T^*$, then with probability at least $z/k^{3/5}$, no query has been performed between $S_j \setminus T^*$ and $V_{g(j)}$. Thus with probability at least $1 - \varepsilon/100$, for each $1 \leq j \leq d'$, no query has been performed on $E(S_j \setminus T^*, V_{g(j)})$. If this happens, then it is easy to verify that $I'_{(f,g)}$ is a well defined and consistent instance in $\mathcal{I}_{\mathsf{Y}}$, as the difference between w and w_g, in particular a subset of edges in $\bigcup_{1 \leq j \leq d'} E(S_j \setminus T^*, V_{g(j)})$, has not been queried at all.

Altogether, there are at least 0.98-fraction of the good instances in $\mathcal{I}_{\mathsf{N}}$, such that for each such I_f, at least $(1 - \varepsilon/100 - 1/10^4) \leq (1 - 10^{-2})$-fraction of mappings $g \in \mathcal{G}$ can give consistent instances $I'_{(f,g)}$ in $\mathcal{I}_{\mathsf{Y}}$.

From consistent instances in $\mathcal{I}_{\mathsf{Y}}$ to consistent instances in $\mathcal{I}_{\mathsf{N}}$. From Lemma 7.1, for every unqueried edge between a Steiner vertex and a not-$(\epsilon/100)$-well-discovered terminal, the probability that the edge is a crucial (weight-1) edge and the Steiner vertex is a secret vertex is at most $2/k^{8/5}$. Therefore, for every $u \in S \setminus T^*$ and every $1 \leq i \leq d'$, the probability that u is a special terminal and belongs to a group S_j with $g(j) = i$ is at most $(2/k^{8/5}) \cdot (k^{2/5}) = 2/k^{6/5}$. For each $1 \leq i \leq d'$, if we denote by z_i the number of terminals $u \in S \setminus T^*$ such that some edge from $E(u, V_i)$ has been queried, then

$$\Pr\left[\exists j, \text{ s.t. } g(j) = i, \text{ and some edge in } E(S_j \setminus T^*, V_i) \text{ has been queried}\right] \leq \frac{2z_i}{\varepsilon k^{6/5}}.$$

Since the algorithm performs at most $\varepsilon^2 k^{6/5}/10^9$ queries, from Markov's Bound, with probability at least $1 - 10^{-6}$ (i.e., on at least $(1 - 10^{-6})$-fraction of the good instances in $\mathcal{I}_{\mathsf{Y}}$), for all pairs (i, j) with $g(j) = i$, no queries has been perform on $E(S_j \setminus T^*, V_i)$. For each consistent instance $I'_{(f,g)} \in \mathcal{I}_{\mathsf{Y}}$ with the above property, it is easy to verify that the corresponding instance I_f in $\mathcal{I}_{\mathsf{N}}$ is a consistent instance in $\mathcal{I}_{\mathsf{N}}$, as the difference between w and w_g, in particular a subset of edges in $\bigcup_{1 \leq j \leq d'} E(S_j \setminus T^*, V_{g(j)})$, has not been queried at all.

Altogether, there are at least $(1 - 10^{-6})$ fraction of the good instances in $\mathcal{I}_{\mathsf{Y}}$, such that for each $I'_{(f,g)}$ of them, the corresponding instance I_f is a consistent instance in $\mathcal{I}_{\mathsf{N}}$.

Form the above discussion, the number of consistent instances in $\mathcal{I}_{\mathsf{Y}}$ is at least $0.9 \cdot 0.98 \cdot (1 - 10^{-2})|\mathcal{G}| > 0.85|\mathcal{G}|$ times the number of consistent instances in $\mathcal{I}_{\mathsf{N}}$, and it is at most $|\mathcal{G}|/(0.9 \cdot (1 - 10^{-6})) \leq 1.12|\mathcal{G}|$ times the number of consistent instances in $\mathcal{I}_{\mathsf{N}}$. Therefore, the algorithm reports correctly with probability at most $\max\{1/(1 + 0.85), 1.12/(1 + 1.12)\} \leq 2/3$.

7.2.2 Proof of Lemma 7.1 for $\mathcal{D}_{\mathsf{N}}$

In this subsection we prove of the first half of Lemma 7.1. We start with the following claim.

Claim 7.3. *Let σ be any transcript, and let $\mathcal{D}_{\mathsf{N}}(\sigma)$ be the uniform distribution on all instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ. Then*

- *for every unqueried edge between a pair of terminals such that at least one is unsettled and not $(\varepsilon/100)$-well-discovered, the probability in $\mathcal{D}_{\mathsf{N}}(\sigma)$ that the edge is a crucial (weight-0) edge is at most $2/k^{2/5}$; and*
- *for every unqueried edge between a Steiner vertex and a unsettled and not-$(\varepsilon/100)$-well-discovered terminal, the probability in $\mathcal{D}_{\mathsf{N}}(\sigma)$ that the edge is a crucial (weight-1) edge is at most $2/k$.*

Proof. We first prove the first property. Consider a pair u, u' of terminals where terminal u is unsettled and not $(\varepsilon/100)$-well-discovered. Let I, I' be instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ, such that in I, u and u' belong to the same T_j group. We say that I' is *host* by I, iff I' can be obtained from I by exchanging the role of u with another unsettled regular terminal u'' that is not in the same group as u in I. It is clear that in any such instance I', the answer of the same query will 2 (that is, the edge is not a crucial edge). Suppose u and u' are in group T_j. The number of such instances I' equals the number of terminal u'' such that (i) $u'' \notin T_j$, and no edge

in $E(u'', T_j)$ has been queried; and (ii) no edge between u and the group u'' belongs to has been queried. Since u is not $(\varepsilon/100)$-well-discovered, Property P1, the number of u'' that violate (ii) is at most $\varepsilon k/100$, and from Property P3, the number of u'' that violate (i) is at most $\varepsilon k/100$. Therefore, I hosts at least $49k/50$ instances I'. On the other hand, for each instance I', the number of instances that hosts it is at most $k^{3/5}$, since $|T_j| \leq k^{3/5}$. Altogether, over all instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ, at most $50/(49k^{2/5}) \leq (2/k^{2/5})$-fraction of them have (u, u') as a crucial edge.

We now prove the second property. Consider an unsettled and not-$(\varepsilon/100)$-well-discovered terminal u and a Steiner vertex v. Let I, I' be instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ, such that in I, u is a special terminal and $v \in V(u)$. We say that I' is host by I if I' can be obtained from I by exchanging the role of u with a unsettled regular terminal u' in I. The number of such instances is the number of unsettled regular terminals u' in I such that (i) no edge in $E(u', V(u))$ has been queried; and (ii) no edge between u and the group that contains u' has been discovered. Since u is not $(\varepsilon/100)$-well-discovered, from Property P3, the number of regular terminals that violate (i) is at most $\varepsilon k/100$, and Property P1, the number of regular terminals that violate (ii) is at most $\varepsilon k^{3/5} \cdot k^{2/5}/100 = \varepsilon k/100$. Therefore, I hosts at least $49k/50$ instances. On the other hand, any instance I' is host by at most one instance since at most one crucial edge is incident to$v$. Altogether, over all instances in $\mathcal{I}_{\mathsf{N}}$ that are consistent with σ, at most $50/49k \leq (2/k)$-fraction of them have (u, v) as a crucial edge. ∎

In the remainder of this subsection, we will refer to $(\varepsilon/200)$-well-discovered vertices as *well-discovered vertices*, for convenience. We call queries between two terminals *regular* queries, and queries between a terminal and a Steiner node *special* queries. We say a query is *good* iff it discovers a previously-unknown crucial edge. In other words, either (i) the query is a regular query, such that at least one endpoint is not well-discovered, and the answer is 0, or (ii) the query is a special query such that the terminal is not $(\varepsilon/100)$-well-discovered, and the answer is 1. From Claim 7.3, the probability that a regular query is good is at most $2/k^{2/5}$, and the probability that a special query is good is at most $2/k$. Therefore, if the algorithm performs at most $\varepsilon^2 k^{6/5}/10^9$ queries, then from Markov's Bound, with probability $1/50$, the number of good regular queries is at most $\varepsilon^2 k^{2/5}/10^6$, and the number of good special query is at most $\varepsilon^2 k^{3/5}/10^6$.

For every settled terminal, if it is not settled by a good query, it must become well-discovered before it become settled. Thus if we can upper bound the number of terminals that are well-discovered without but not settled at some step, then we can upper bound the number of settled or well-discovered terminals as well. For ease of analysis, we always assume that there are at most $\varepsilon k^{4/5}/10^4$ terminals and $\varepsilon k^{1/5}/10^4$ special terminals that are settled or well-discovered, and we think of the algorithm as being immediately terminated once this condition no longer holds.

There are four possiblities for a terminal to become well-discovered, and we analyze them seperately.

Possibility 1: through P3. Let u be such a vertex, so u is a regular terminal and the group that T_i contains it has $\varepsilon k/200$ incident edges discovered. For each such edge, it is either discovered by a query or because the its other endpoint is settled or well-discovered. Since there are at most $\varepsilon k^{4/5}/10^4$ settled or well-discovered terminals, there are at least $0.004\varepsilon k$ edges incident on T_i that are discovered by queries. Thus, if the total number of queries is at most $\varepsilon^2 k^{6/5}/10^9$, then there are at most $\varepsilon k^{1/5}/10^6$ groups such that the terminals in this group is well-discovered through P3. Therefore, the number of terminals that become well-discovered through P3 is at most $\varepsilon k^{4/5}/10^6$. Note that all these terminals are regular terminals.

Possibility 2: through P4. Let u be such a vertex, so u is a special terminal and the set $V(u)$ has $\varepsilon k/200$ incident edges discovered. By the same argument, for at most $\varepsilon k^{4/5}/10^4$ of these edges, the other endpoint is a settled or well-discovered terminal, and all the others are discovered by queries. Therefore, the number of such terminals is at most $\varepsilon k^{1/5}/10^6$, and all of them are special terminals.

Possibility 3: through P1 but not P3/P4. Let u be such a vertex, so there are at least $\varepsilon k^{2/5}/200$ groups T_i such that some edge in $E(u, T_i)$ has been discovered. Note that each such edge is discovered either by a query, or because we have discovered all terminals in T_i. By previous analysis, there are at most $\varepsilon k^{1/5}$ of them. Therefore, at least $0.004\varepsilon k^{2/5}$ edges incident on u are queried.

Possibility 4: through P2 but not P3/P4. Let u be such a vertex, so there are at least $\varepsilon k^{2/5}/200$ special terminals u' such that some edge in $E(u, V(u'))$ has been discovered. Note that such an edge is discovered either by a query, or because u' has already been settled. Therefore, at least $0.004\varepsilon k^{2/5}$ edges incident to u are queried.

From the analysis in Possibilities 3 and 4, for any terminal that becomes well-discovered through P1 or P2 but not P3 or P4, at least $0.004\varepsilon k^{2/5}$ of its incident edges have been queried. Therefore, there are at most $\varepsilon k^{4/5}/(4 \cdot 10^6)$ such vertices. However, such terminals could be either regular or special, and we still need to

upper bound the number of such special terminals. Let u be any such terminal, and consider the moment when exactly $0.004\varepsilon k^{2/5}$ of its incident edges are queried. From Claim 7.3, for any Steiner node v, the probability that (u, v) is a crucial (weight-1) edge is at most $2/k$. It follows that the probability that u is a special terminal is at most $(2/k) \cdot (k/k^{3/5}) = 2/k^{3/5}$. By Markov's Bound, the number of special terminals that are well-discovered is at most $\varepsilon k^{1/5}/10^5$ with probability at least $1/40$.

Altogether, with probability $1 - 1/40 - 1/50 \geq 9/10$, the number of settled or well-discovered terminals is at most $(\varepsilon^2 k^{4/5} + \varepsilon k^{4/5} + \varepsilon k^{1/5} + \varepsilon k^{4/5})/10^6 < \varepsilon k^{4/5}/10^4$, and the number of settled or well-discovered special terminals is at most $(\varepsilon^2 k^{1/5} + \varepsilon k^{1/5})/10^6 + \varepsilon k^{1/5}/10^5 < \varepsilon k^{1/5}/10^4$.

7.2.3 Proof of Lemma 7.1 for $\mathcal{D}_{\mathsf{Y}}$ In this subsection we provide the proof of the second half of Lemma 7.1.

Recall that Steiner vertices that are incident to more than one crucial edges are called *secret vertices*. We assume for now that, over the course of the algorithm,

- for every Steiver vertex, we have discovered at most one crucial edge incident to it;
- no secret vertex has more than $\varepsilon k/10^4$ of its incident edges discovered;
- the number of terminals that are settled or well-discovered is at most $\varepsilon k^{4/5}/10^4$; and
- the number of special terminals that are settled or well-discovered is at most $\varepsilon k^{1/5}/10^4$.

We will show at the end of this subsection that, if any of the above condition is not satisfied, then the algorithm has to perform at least $\varepsilon^2 k^{6/5}/10^9$ as well.

We start with the following claim, whose is very similar to Claim 7.3, and is omitted here.

Claim 7.4. *Let σ be any transcript, and let $\mathcal{D}_{\mathsf{Y}}(\sigma)$ be the uniform distribution on all instances in $\mathcal{I}_{\mathsf{Y}}$ that are consistent with σ. Then*

- *for every unqueried edge between a pair of terminals such that at least one is unsettled and not $(\varepsilon/100)$-well-discovered, the probability in $\mathcal{D}_{\mathsf{Y}}(\sigma)$ that the edge is a crucial (weight-0) edge is at most $2/k^{2/5}$; and*
- *for every unqueried edge between a Steiner vertex and a unsettled and not-$(\varepsilon/100)$-well-discovered terminal, the probability in $\mathcal{D}_{\mathsf{Y}}(\sigma)$ that the edge is a crucial (weight-1) edge is at most $2/k$.*

Using Claim 7.4 and the same arguments in the proof of Lemma 7.1 for $\mathcal{D}_{\mathsf{N}}$, we can prove that the number of terminals that are settled or well-discovered is at most $\varepsilon k^{4/5}/10^4$, and at most $\varepsilon k^{1/5}/10^4$ of them are special terminals. In order to prove that all settled or well-discovered special terminals belong to different S_j groups, we use the following two claims.

Claim 7.5. *Let u^* be a special terminal that is either settled or $(\varepsilon/200)$-well discovered. Then for every other not-$(\varepsilon/100)$-well-discovered terminal u and any other Steiner vertex v, the probability that u is a special terminal in the same S_j group as u^* and (u, v) is a crucial edge is at most $2/k^{7/5}$.*

Proof. Let I, I' be instances in $\mathcal{I}_{\mathsf{N}}$, let u^* be a special terminal that is either settled or $(\varepsilon/200)$-well discovered in both I and I', let u be a special vertex in the same S_j group as u^* in I, and let v be a Steiner vertex such that (u, v) is a crucial edge in I. We say I' is *host* by I, iff there are two terminals u', u'', such that u' is special and u'' is regular, and I' can be obtained from I by giving the role of u' to u, giving the role of u to u'' and giving the role of u'' to u'. The vertex u' can be any not-$(\varepsilon/100)$-well-discovered terminal in S_j with no edge to $V(u)$ discovered. Since u is not $(\varepsilon/100)$-well-discovered, and we have assumed that each secret Steiner vertex has at most $\varepsilon k/10^4$ of its incident edges discovered, there are at least $0.99k$ choices for u'. On the other hand, u'' can be any terminal that has no edge discovered to $V(u')$ and the secret Steiner node of group S_j in I. By the same arguement, there are at least $0.99k$ such terminal. Since we have assumed that at most $\varepsilon k^{1/5}/10^4$ special terminals are $(\varepsilon/100)$-well-discovered, an instance I hosts at least $0.99^2 k^2 \cdot (k^{2/5}/\varepsilon) > 0.98k^{12/5}/\varepsilon$ instances I'. On the other hand, for any instance I', the terminal u' must be the terminal such that $v \in V_{u'}$, and the terminal u'' must in the same group as u^*. So there are at most k/ε instance I that hosts I'. Altogether, the probability that all events happend is at most $(k/\varepsilon)/(0.98k^{12/5}/\varepsilon) \leq 2/k^{7/5}$. ∎

CLAIM 7.6. *Let u be a not-$(\varepsilon/100)$-well-discovered terminal and let v be a Steiner vertex. Then the probability that (u,v) is a crucial (weight-1) edge and v is an secret vertex is at most $2/k^{8/5}$.*

Proof. Recall that we have assumed that v has at most one crucial edge connecting to a settled or a well-discovered terminal discovered. By definition, any $(\varepsilon/100)$-well discovered terminal is also a well discovered terminal. So there v v has at most one crucial edge connecting to a settled or a $(\varepsilon/100)$-well-discovered terminal discovered. We distinguish between the following two cases.

Case 1: There does not exist a settled or $(\varepsilon/100)$-well discovered terminal u', such that (u',v) is a crucial edge and has been discovered. For any consistent instance I such that u and v has weight 1 and v is an secret Steiner node. Suppose $u \in S_i$ and let $S_i = \{s_{i,1}, \ldots, s_{i,1/\varepsilon}\}$, and suppose $v \in V(s_{i,1})$. By assumption, no terminal in S_i is $(\varepsilon/100)$-well discovered. We say an instance I' is hosted by I if all special terminals not in S_i and their weights to the Steiner nodes are the same as I, and the set $V(s_{i,t})$ in I' is identical to the set $V(s_{i,t})$ in I for all $1 \le t \le 1/\varepsilon$.

We first count the number of such consistent instances I'. For each $1 \le t \le 1/\varepsilon$, we define T_t^* as the set of regular terminals that with no edges to $V(s_{i,t})$ discovered. Since no terminal in S_i is $(\varepsilon/100)$-well discovered, every set T_t^* has size at least $(1-\varepsilon/100)k$. For each $1 \le t \le 1/\varepsilon$, consider any group of terminals $t_1 \ldots, t_{1/\varepsilon}$ such that $t_{j'} \in T'_{j'} \cap T'_j$. Any $t_{j'}$ can have weight one to all Steiner nodes in $V_{s_{j'}}$ and V_{s_j}, which means we can make any terminal in V_{s_j} an secret terminal. Thus such group can construct at least $k^{3/5}/\varepsilon$ consistent instances I'. On the other hand, since any $|T_{j'}| \ge (1-\varepsilon/100)k$, the number of such group is at least $((1-49\varepsilon/50)k)^{1/\varepsilon} > (49/50)\cdot k^{1/\varepsilon}$. Thus, I hosts at least $(49/50)k^{1/\varepsilon} \cdot (k^{3/5}/\varepsilon)$ instances I'.

Now we count how many instance can host an instance I'. For any instance I', an instance I that hosts it can only change the terminals in S_i that has weight 1 to v, and so one of the terminal in S_i should be u. Moreover, every terminal in S_i should has weight 1 to v. The total number of such instance is at most $k^{1/\varepsilon} \cdot 1/\varepsilon$ since u could replace any terminal in S_i. Thus, the probability that v and u has weight 1 and v is an secret Steiner node is at most $(k^{1/\varepsilon-1} \cdot (1/\varepsilon))/((49/50)k^{1/\varepsilon}k^{3/5} \cdot (1/\varepsilon)) < 2/k^{8/5}$.

Case 2: There does not exist a settled or $(\varepsilon/100)$-well discovered terminal s_{j^*}, such that (s_{j^*},v) is a crucial edge and has been discovered. Note that $s_{j^*} \neq u$. For any instance I such that u and v has weight 1. We say an instance I' is hosted by I as the same definition as the first case, except that now u must still in S_i, and moreover, we exchange one Steiner node in V_{j^*} with V_j for some j. We first count the number of I' hosted by I. We define T'_j the same as the first case. Suppose $v \in V_{\ell^*}$, for any $1 \le \ell \le k^{3/5}$ and $\ell \neq \ell^*$, we define v^*_ℓ as the only one vertex in $V_j^* \cap V_\ell$, and T^*_ℓ as the set of terminals that does not have weight one to v^*_ℓ. We also define $v^*\ell^* = v$ and $T^*_{\ell^*}$ as the set of terminals that does not have weight one to $v^*_{\ell^*}$. Now for any $1 \le \ell \le k^{3/5}$, for any group of terminals $t_1, \ldots, t_{1/\varepsilon}$ such that for any $j' \neq j^*$, $t_{j'} \in T'_{j'} \cap T^*_\ell$ and $t_{j^*} = s_{j^*}$, we can make v^*_ℓ the secret Steiner node of this group. Moreover, to do so, we can exchange v^*_ℓ from $V_{s_{j^*}}$ with any $V_{s_{j'}}$ since we will not change the weight between any pair of vertices by doing so. Since the algorithm only perform at most $\varepsilon^2 k^{6/5}/10^9$ queries and settled or $(\varepsilon/100)$-well discovered at most $O(k^{4/5})$ terminals, there are $(1-o(1))k^{3/5}$ number of index ℓ such that $|T^*_\ell| > (1-\varepsilon/100)k$. For such ℓ, the number of groups $t_1, \ldots, t_{1/\varepsilon}$ is at least $0.99k^{1/\varepsilon-1}$. So the total number of instances $\mathcal{I}'$ hosted by I is at least $(1-o(1)k^{3/5} \cdot k^{1/\varepsilon-1} \cdot (1/\varepsilon)$. On the other hand, for any instance I', it is hosted by at most $k^{1/\varepsilon-2} \cdot (1/\varepsilon)$ instance I since we cannot exchange s_{ℓ^*} and u must be a special terminal in S_i. This implies that v and u has weight 1 and v is a secret Steiner node is at most $(k^{1/\varepsilon-2} \cdot (1/\varepsilon))/((1-o(1))0.99k^{1/\varepsilon-1}k^{3/5} \cdot 1/\varepsilon) < 2/k^{8/5}$. ∎

Remember that a special query is called a *good* query iff it discovers a crucial edge between a not-well-discovered terminal and a Steiner node. If a special terminal is ever settled, then it is either due to a good special query incident to it, or because it becomes well-discovered at some step. We analyze the probability that a special query (u,v) is a good query, and u is in the same group S_i with some terminal u' that is already settled or $(\varepsilon/200)$-well discovered. By Claim 7.6, the probability that the query is a good query and v is an secret Steiner node is at most $2/k^{8/5}$. On the other hand, if the query is a good query but v is not an secret terminal, it means $v \in V(u)$. By Claim 7.5, the probability that it is a good query and u and a fix settled or $(\varepsilon/200)$-well discovered u^* in the same group S_i is at most $2/k^{7/5}$. Since there are at most $\varepsilon k^{1/5}/10^4$ such u^*. So the probability that a special is a good query and u is in the same group S_i with some terminal u' that is already settled or $(\varepsilon/200)$-well discovered is at most $\varepsilon/10^4 k^{6/5}$. With probablity at least $1-\varepsilon/1000$, there is no such query through out the algorithm by Markov's Bound.

Now we consider the well-discovered terminals. When a terminal is well-discovered but not settled, it is still not $(\varepsilon/100)$-well discovered. By Claim 7.6, the probability that it is in the same group S_i with some terminal u' that is already settled or $(\varepsilon/200)$-well discovered is at most $(2/k^{8/5}) \cdot (\varepsilon k^{1/5}/10^4) \cdot k^{3/5} < \varepsilon/100k^{4/5}$. Since there are at most well-discovered $\varepsilon k^{4/5}/10^4$, no well-discovered terminal belongs to the same group with some terminal u' that is already settled or well-discovered.

Finally, we need to prove the assumption that there is no secret Steiner node has at least $\varepsilon k/10^4$ edges discovered. Since we only performed at most $\varepsilon^2 k^{6/5}/10^9$ queries, there are at most $\varepsilon k^{1/5}/10^4$ terminals that are queried at least $\varepsilon k/10^5$ times. By Claim 7.6, the probability that such terminal is secret is at most $2/k^{8/5} \cdot k = 2/k^{3/5}$. Thus with proability at least $1 - k^{-2/5}$, all these terminals are not secret. This finishes the proof of Lemma 7.1.

A An $O(nk)$-Query $(5/3)$-Approximation Algorithm

In this section, we explain how the previous work [27] and [28] lead to an $O(nk)$-query algorithm for computing a $(5/3)$-approximate Steiner Tree. In fact, their results imply the following theorem.

THEOREM A.1. *For any instance (V, T, w) of Steiner Tree problem, there exists a Steiner tree $\mathcal{T}^*$ with weight at most $(5/3) \cdot \mathsf{ST}(V, T, w)$, such that every edge in $\mathcal{T}^*$ is incident on some vertex in T.*

We refer to such Steiner trees as *good trees*. From the above theorem, it is not hard to observe that querying all terminal related distances is sufficient to find a $(5/3)$-approximate Steiner Tree, and the query complexity is $O(nk)$.

We now explain how the results in previous work [27] and [28] imply Theorem A.1.

We start by introducing some definitions. Let $\mathcal{T}$ be a tree, let v be a vertex of $\mathcal{T}$, and let $v_1, \ldots, v_d$ be the neighbors of v. For each $1 \le i \le d$, we delete edges $(v, v_1), \ldots, (v, v_{i-1}), (v, v_{i+1}), \ldots, (v, v_d)$, and define $\mathcal{T}_i$ to be the connected component of the remaining graph that contains v, so $\mathcal{T}_i$ is a subtree of $\mathcal{T}$ that contains v. We say that subtrees $\mathcal{T}_1, \ldots, \mathcal{T}_d$ are obtained by *splitting* $\mathcal{T}$ *at* v.

Consider an instance (V, T, w) and let $\mathcal{T}$ be a Steiner tree. Let $c > 1$ be an integer. We say that $\mathcal{T}$ is a *c-Steiner tree*, iff when we split $\mathcal{T}$ at all terminals, then each resulting subtree contains at most c terminals. It is easy to verify that any 2-Steiner Tree is a terminal spanning tree. We now show that every 3-Steiner tree can be converted into a good tree with at most the same cost. Let $\mathcal{T}$ be a 3-Steiner tree. Assume without loss of generality that every Steiner vertex has degree at least 3 (as otherwise we can suppress such a vertex and get another Steiner tree with at most the same cost). We now claim that $\mathcal{T}$ does not contain any Steiner-Steiner edge. Assume not, then such a pair of Steiner vertices must both belong to some subtree obtained by splitting $\mathcal{T}$ at all terminals, and such a subtree contains at least $(3 + 3 - 1 - 1) = 4$ terminals, a contradiction. It was shown in [27] and [28] that, for any instance (V, T, w), there exists a 3-Steiner tree with cost at most $(5/3) \cdot \mathsf{ST}(V, T, w)$, and the ratio 5/3 here cannot be improved. Theorem A.1 now follows.

B Proof of Claim 3.1

Let v_1, v_2, v_3 be three vertices in V. Assume $v_1 \in V_{x_1}$, $v_2 \in V_{x_2}$, and $v_3 \in V_{x_3}$, where x_1, x_2, x_3 are nodes in tree ρ. We denote by ℓ_1, ℓ_2, ℓ_3 the levels of x_1, x_2, x_3, respectively, and assume w.l.o.g. that $\ell_1 \ge \ell_2$. Let x_1' be a leaf of ρ that lies in the subtree of ρ rooted at x_1, and we define leaves x_2', x_3' similarly.

We first show that w_{N} is a metric on V by showing that $w_{\mathsf{N}}(v_1, v_2) \le w_{\mathsf{N}}(v_1, v_3) + w_{\mathsf{N}}(v_2, v_3)$. By definition, $w_{\mathsf{N}}(v_1, v_2) = \mathrm{dist}_\rho(x_1', x_1) + \mathrm{dist}_\rho(x_1', x_2)$. Let $\hat{x}$ be the lowest common ancestor of nodes x_1 and x_2 in ρ. Assume that y is the unique vertex on the $x_1' - x_2'$ path that is closest (under dist_ρ) to x_3. We distinguish between the following three cases, depending on the location of y.

Case 1: y lies between (excluding) $\hat{x}$ and (including) x_1'. On the one hand, $w_{\mathsf{N}}(v_2, v_3) \ge \mathrm{dist}_\rho(x_2, x_3)$; on the other hand, $w_{\mathsf{N}}(v_1, v_3) \ge \mathrm{dist}_\rho(x_1, x_3) + 2 \cdot \min\{\mathrm{dist}_\rho(x_1', x_1), \mathrm{dist}_\rho(x_3', x_3)\}$.

If y lies between (excluding) $\hat{x}$ and (including) x_1, then

$$\begin{aligned}
&w_{\mathsf{N}}(v_1,v_3)+w_{\mathsf{N}}(v_2,v_3) \geq \mathrm{dist}_\rho(x_2,x_3)+\mathrm{dist}_\rho(x_1,x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(y,x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\mathrm{dist}_\rho(y,x_3)+2\cdot\min\{0,\mathrm{dist}_\rho(x_3',x_3)-\mathrm{dist}_\rho(x_1',x_1)\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(y,x_3),\mathrm{dist}_\rho(y,x_3)+\mathrm{dist}_\rho(x_3',x_3)-\mathrm{dist}_\rho(x_1',x_1)\}\\
&\geq\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(y,x_3),0\}\\
&\geq\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)\\
&=\mathrm{dist}_\rho(x_1',x_1)+\mathrm{dist}_\rho(x_1',x_2)=w_{\mathsf{N}}(v_1,v_2).
\end{aligned}$$

If y lies between (excluding) x_1 and (including) x_1', then

$$\begin{aligned}
&w_{\mathsf{N}}(v_1,v_3)+w_{\mathsf{N}}(v_2,v_3) \geq \mathrm{dist}_\rho(x_2,x_3)+\mathrm{dist}_\rho(x_1,x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(y,x_1)+2\cdot\mathrm{dist}_\rho(y,x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\mathrm{dist}_\rho(y,x_3)+2\cdot\min\{\mathrm{dist}_\rho(y,x_1),\mathrm{dist}_\rho(x_3',x_3)-\mathrm{dist}_\rho(y,x_1')\}\\
&\geq\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(y,x_3),\mathrm{dist}_\rho(y,x_3)+\mathrm{dist}_\rho(x_3',x_3)-\mathrm{dist}_\rho(y,x_1')\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(y,x_3),0\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)\\
&=\mathrm{dist}_\rho(x_1',x_1)+\mathrm{dist}_\rho(x_1',x_2)=w_{\mathsf{N}}(v_1,v_2).
\end{aligned}$$

Case 2: y lies between (excluding) $\hat{x}$ and (including) x_2'. The analysis in this case is symmetric to that of Case 1, with an additional observation that $\mathrm{dist}_\rho(x_1,x_1') \leq \mathrm{dist}_\rho(x_2,x_2')$ (as $\ell_1 \geq \ell_2$).

Case 3: $y=\hat{x}$. In this case, from the definition of w_{N}, $w_{\mathsf{N}}(v_2,v_3) \geq \mathrm{dist}_\rho(x_2,x_3)$, and $w_{\mathsf{N}}(v_1,v_3) \geq \mathrm{dist}_\rho(x_1,x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}$. Therefore,

$$\begin{aligned}
&w_{\mathsf{N}}(v_1,v_3)+w_{\mathsf{N}}(v_2,v_3) \geq \mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(\hat{x},x_3)+2\cdot\min\{\mathrm{dist}_\rho(x_1',x_1),\mathrm{dist}_\rho(x_3',x_3)\}\\
&\geq\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(\hat{x},x_3),\mathrm{dist}_\rho(x_3',x_3)+\mathrm{dist}_\rho(\hat{x},x_3)-\mathrm{dist}_\rho(x_1',x_1)\}\\
&\geq\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)+2\cdot\min\{\mathrm{dist}_\rho(\hat{x},x_3),0\}\\
&=\mathrm{dist}_\rho(x_1,x_2)+2\cdot\mathrm{dist}_\rho(x_1',x_1)\\
&=\mathrm{dist}_\rho(x_1',x_1)+\mathrm{dist}_\rho(x_1',x_2)=w_{\mathsf{N}}(v_1,v_2).
\end{aligned}$$

This completes the proof that w_{N} is a metric on V.

We now proceed to show that w_{Y} is a metric on V by showing that $w_{\mathsf{Y}}(v_1,v_2) \leq w_{\mathsf{Y}}(v_1,v_3)+w_{\mathsf{Y}}(v_2,v_3)$. We distinguish between the following cases.

Case 1: $v_1,v_2,v_3 \in S$. In this case,

$$w_{\mathsf{Y}}(v_1,v_2)=\mathrm{dist}_\rho(v_1,v_2) \leq \mathrm{dist}_\rho(v_1,v_3)+\mathrm{dist}_\rho(v_2,v_3)=w_{\mathsf{Y}}(v_1,v_3)+w_{\mathsf{Y}}(v_2,v_3).$$

Case 2: At most one of v_1,v_2,v_3 lies in S. In this case,

$$w_{\mathsf{Y}}(v_1,v_2)=w_{\mathsf{N}}(v_1,v_2) \leq w_{\mathsf{N}}(v_1,v_3)+w_{\mathsf{N}}(v_2,v_3)=w_{\mathsf{Y}}(v_1,v_3)+w_{\mathsf{Y}}(v_2,v_3).$$

Case 3: Exactly two of v_1,v_2,v_3 lie in S. We further consider the following subcases.

Case 3.1: $v_1,v_2 \in S$, and $v_3 \notin S$. In this case,

$$w_{\mathsf{Y}}(v_1,v_2) \leq w_{\mathsf{N}}(v_1,v_2) \leq w_{\mathsf{N}}(v_1,v_3)+w_{\mathsf{N}}(v_2,v_3)=w_{\mathsf{Y}}(v_1,v_3)+w_{\mathsf{Y}}(v_2,v_3).$$

Case 3.2: $v_2,v_3 \in S$, and $v_1 \notin S$. The analysis in this case uses almost identical arguments as Case 1 for showing that w_{N} is a metric (since there we only uses the fact that $w_{\mathsf{N}}(x_2,x_3) \geq \mathrm{dist}_\rho(x_2,x_3)$).

Case 3.3: $v_1,v_3 \in S$, and $v_2 \notin S$. The analysis in this case uses almost identical arguments as Case 2 for showing that w_{N} is a metric (since there we only uses the fact that $w_{\mathsf{N}}(x_1,x_3) \geq \mathrm{dist}_\rho(x_1,x_3)$).

This completes the proof that w_{Y} is a metric on V.

C Proof of Observation 3.2

Let $\mathcal{T}'$ be an optimal Steiner Tree of instance (S, T, w_{N}), and assume for contradiction that $\mathcal{T}'$ contains an edge $(u_x, u_{x'})$ where $u_x, u_{x'} \notin T$ (or equivalently $x, x' \notin L(\rho)$). Assume without loss of generality that the level of x in ρ is at least the level of x'. Let $\tilde{x}$ be any leaf in ρ that is a descendant of x. Consider now the unique path in $\mathcal{T}'$ connecting $u_{\tilde{x}}$ to u_x, that we denote by P.

Assume first that the vertex $u_{x'}$ does not belong to P. Let $\mathcal{T}$ be the tree obtained from $\mathcal{T}'$ by replacing the edge $(u_x, u_{x'})$ with edge $(u_{\tilde{x}}, u_{x'})$. It is easy to verify that $\mathcal{T}$ is a Steiner Tree. Moreover, from the definition of w_{N},

$$w_{\mathsf{N}}(u_x, u_{x'}) = \text{dist}_\rho(x, x') + 2 \cdot \text{dist}_\rho(x, \tilde{x}) > \text{dist}_\rho(x, x') + \text{dist}_\rho(x, \tilde{x}) = w_{\mathsf{N}}(u_{\tilde{x}}, u_{x'}),$$

which implies that $w(\mathcal{T}) < w(\mathcal{T}')$, a contradiction to the assumption that $\mathcal{T}'$ is an optimal Steiner Tree of the instance (S, T, w_{N}).

Assume now that vertex $u_{x'}$ belongs to P. Similarly, let $\mathcal{T}$ be the tree obtained from $\mathcal{T}'$ by replacing the edge $(u_x, u_{x'})$ with the edge $(u_{\tilde{x}}, u_x)$. It is easy to verify that $\mathcal{T}$ is a Steiner Tree. Moreover, from the definition of w_{N},

$$w_{\mathsf{N}}(u_x, u_{x'}) = \text{dist}_\rho(x, x') + 2 \cdot \text{dist}_\rho(x, \tilde{x}) > \text{dist}_\rho(x, \tilde{x}) = w_{\mathsf{N}}(u_{\tilde{x}}, u_x),$$

which implies that $w(\mathcal{T}) < w(\mathcal{T}')$, again a contradiction to the assumption that $\mathcal{T}'$ is an optimal Steiner Tree of the instance (S, T, w_{N}). This completes the proof of the observation.

D Proof of Observation 3.3

First of all, it is easy to see that there exists an optimal Steiner Tree such that every Steiner vertex has degree at least 3, since we can compress degree-2 Steiner vertices without increasing the cost.

Consider now a tree $\mathcal{T}'$ such that:

1. $\mathcal{T}'$ is an optimal Steiner Tree such that every Steiner vertex has degree at least 3;
2. on top of 1, $\mathcal{T}'$ minimizes the number of Steiner vertices;
3. on top of 1 and 2, $\mathcal{T}'$ maximizes the sum of levels of all its Steiner vertices; and
4. on top of 1, 2, and 3, $\mathcal{T}'$ minimizes the number of edges incident to Steiner vertices.

For each Steiner vertex u_x in $\mathcal{T}'$, we denote by $d_0(x)$, $d_1(x)$, $d_2(x)$ the number of neighbors of u_x in sets $T_0(x)$, $T_1(x)$, $T_2(x)$, respectively. Consider now a Steiner vertex u_x in $\mathcal{T}'$. Let $u_{x'}$ be the parent of u_x, and let u_{x_1}, u_{x_2} be the children of u_x, where $u_{x_1} \in S_1(x)$ and $u_{x_2} \in S_2(x)$. We distinguish between the following cases.

Case 1: One of $d_0(x), d_1(x), d_2(x)$ is 0. Assume first that $d_0(x) = 0$. Then if $d_1(x) \geq d_2(x)$, we can replace Steiner vertex u_x with u_{x_1} (that is, delete from $\mathcal{T}'$ the vertex u_x and all its incident edges, and replace them with vertex u_{x_1} and edges in $\{(u, u_{x_1}) \mid (u, u_x) \in E(\mathcal{T}')\}$), without increasing the total cost. In this way, we either reduce the number of Steiner vertices by 1, or increase the sum of levels of all Steiner vertices, a contradiction to either 2 or 3. The case where $d_1(x) \leq d_2(x)$ is symmetric. Assume now that $d_1(x) = 0$ (the case where $d_2(x) = 0$ is symmetric). Then if $d_2(x) \geq d_0(x)$, we can replace u_x with u_{x_2} either reducing the number of Steiner vertices or increasing the sum of levels of all Steiner vertices, leading to a contradiction to 2 or 3; if $d_2(x) < d_0(x)$, we can replace u_x with $u_{x'}$, reducing the total cost, leading to a contradiction to 1.

Case 2: One of $d_1(x), d_2(x)$ is at least 2. Assume w.l.o.g. that $d_1(x) \geq 2$. Let u, u' be two vertices of $T_1(x)$ that are adjacent to u_x in $\mathcal{T}'$. We can replace the edge (u, u_x) with edge (u, u'), and it is easy to verify from the definition of w_{N} that this does increase the total cost, leading to a contradiction to 4.

Case 3: $d_1(x) = d_2(x) = 1$ and $d_0(x) > 2$. In this case, we can replace u_x with $u_{x'}$, reducing the total cost, leading to a contradiction to 1.

Altogether, we get that $d_1(x) = d_2(x) = 1$ and $d_0(x)$ is either 1 or 2, completing the proof of property (i) in the observation. We now focus on proving property (ii). Let $\mathcal{T}'$ be the tree defined above. Let u_1 (u_2, resp.) be the neighbor of u_x in $T_1(x)$ ($T_2(x)$, resp.). Assume for contradiction that in there exists some vertex $u \in T_1(x)$ such that $u \notin W_1$. From similar arguments in the proof of Observation 3.2, we can replace the edge (u_1, u_x) with edge (u_1, u), obtaining another optimal Steiner Tree with less edges incident to Steiner vertices, a contradiction to 4. Assume for contradiction that in there exists some vertex $u \notin T_1(x)$ such that $u \in W_1$. We root tree W_1 at u_1, and it is easy to see that there exists some pair u', u'' of vertices in W_1, such that $u' \in T_1(x)$, $u'' \notin T_1(x)$

and u' is the parent of u''. Similarly, we can replace the edge (u', u'') with edge (u_x, u''), obtaining another Steiner Tree with strictly lower cost, a contradiction to 1. Therefore, $T_1(x) \subseteq V(W_1) \subseteq S_1(x)$. The proof of $T_2(x) \subseteq V(W_2) \subseteq S_2(x)$ is symmetric.

Last, we prove property (iii) in the observation. Let $\mathcal{T}'$ be the tree defined above. Consider a Steiner vertex u_x in $\mathcal{T}'$. We denote by $u_{x'}$ the parent of u_x, and denote by $u_{\hat{x}}$ the other child of $u_{x'}$. Assume for contradiction that both u_x and $u_{x'}$ belong to $\mathcal{T}'$. From property (i), $u_{x'}$ has a neighbor in $T(x)$, that we denote by u. From property (ii), since $u \in T(x)$, vertex u belongs to either W_1 or W_2. However, $u_{x'} \notin R(x)$, so u does not belong to either W_1 or W_2, a contradiction to the fact that $u_{x'}$ and u are connected by an edge. Assume for contradiction that both u_x and $u_{\hat{x}}$ belong to tree $\mathcal{T}'$. Let u be the neighbor of u_x that belongs to the same connected component of $\mathcal{T}' \setminus \{u_x\}$ as $u_{\hat{x}}$. From property (i) and (ii) on $u_{\hat{x}}$, u does not belong to the corresponding subgraphs $\hat{W}_1, \hat{W}_2$ for $u_{\hat{x}}$. Let $\hat{u}$ be any leaf of $T(\hat{x})$. Via similar arguments, we can replace the edge (u_x, u) with the edge $(u_x, \hat{u})$, obtaining another Steiner Tree with strictly lower cost, a contradiction to 1. Assume for contradiction that none of $u_x, u_{x'}, u_{\hat{x}}$ belongs to $\mathcal{T}'$. Via similar arguments, it is easy to verify that we can add either u_x or $u_{\hat{x}}$ to $\mathcal{T}'$ and deleting some edges, obtaining another Steiner Tree with strictly lower cost, a contradiction to 1.

E Analysis of the Algorithm in Section 6.1

In this section, we show that the spanning tree $\mathcal{T}$ output by the algorithm in Section 6.1 is with high probability an α-approximate Steiner Tree. We denote by MST the minimum spanning tree cost on T. Therefore, it suffices to show that, with high probability, $w(\mathcal{T}) \leq (\alpha/2) \cdot \mathsf{MST}$, as MST is at most twice the minimum Steiner Tree cost. Recall that $\beta = \alpha/(100 \log n)$.

Let $\mathcal{T}^*$ be a minimum spanning tree on T. Let $\pi = (u_1, u_2, \ldots, u_{2k-2})$ be an Euler-tour of $\mathcal{T}^*$, and for each $1 \leq t \leq 2k-2$, we let $R_{\pi,t} = \{u_i \mid t \leq i \leq t + 20\beta \log n\}$. We define a bad event ξ as follows.

Bad event ξ. Let ξ be the event that there exists some t, such that $R_{\pi,t} \cap T' = \emptyset$. We now show that $\Pr[\xi] = O(n^{-9})$. Since each edge of $\mathcal{T}^*$ appears at most twice in the set $\{(u_i, u_{i+1}) \mid u_i \in R_{\pi,t}\}$, $R_{\pi,t}$ contains at least $10\beta \log n$ distinct vertices. Therefore, the probability that a random subset of $\lceil n/\beta \rceil$ vertices in V does not intersect with $R_{\pi,t}$ is at most $(1 - (10\beta \log n/k))^{k/\beta} \leq n^{-10}$. Taking the union bound over all $1 \leq t \leq 2k-2$, we get that $\Pr[\xi_1] = O(n^{-9})$. Note that, if the event ξ does not happen, then every consecutive window of π of length $20\beta \log n$ contains at least one element of T'.

Let $u_{i_1}, u_{i_2}, \ldots, u_{i_{t'}}$ be the vertices of π that belongs to T' (t' may be larger than $|T'|$ since we keep all copies of the same vertex), then for each $1 \leq j \leq t'$, $|i_j - i_{j+1}| \leq 20\beta \log n$.

Let u be any terminal in $T \setminus T'$. Assume that the first appearance of u in π is between u_{i_j} and $u_{i_{j+1}}$, so $w(u, T') \leq \min\left\{w(u, u_{i_j}), w(u, u_{i_{j+1}})\right\} \leq w(u_{i_j}, u_{i_{j+1}})$, which is at most the total weight of all edges in π between u_{i_j} and $u_{i_{j+1}}$. So from triangle inequality, $\mathrm{dist}_\pi(u, T') \leq \sum_{i_j \leq t \leq i_{j+1}-1} w(v_t, v_{t+1})$. As $|i_j - i_{j+1}| \leq 20\beta \log n$ holds for all $1 \leq j \leq t'$, $\sum_{u \in T \setminus T'} w(u, f(u)) \leq 20\beta \log n \cdot w(\mathcal{T}^*) \leq 20\beta \log n \cdot \mathsf{MST}$. Finally, since $w(\mathcal{T}') \leq \mathsf{MST}$, we conclude that $w(\mathcal{T}) \leq 21\beta \log n \cdot \mathsf{MST} \leq (\alpha/2) \cdot \mathsf{MST}$. Altogether, we conclude that, with probability $1 - O(n^{-9})$, the spanning tree $\mathcal{T}$ output by the algorithm in Section 6.1 is an α-approximate Steiner Tree.

Acknowledgements

We thank anonymous reviwers for helpful comments and for pointing to us the previous work [28] and [27].

References

[1] Manuela Fischer and Andreas Noever, *Tight Analysis of Parallel Randomized Greedy MIS*, ACM Trans. Algorithms, 6:1–6:13, 2020.

[2] Grunau, Christoph and Mitrović, Slobodan and Rubinfeld, Ronitt and Vakilian, Ali, *Improved local computation algorithm for set cover via sparsification*, Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2993–3011, 2020.

[3] Har-Peled, Sariel and Indyk, Piotr and Mahabadi, Sepideh and Vakilian, Ali, *Towards tight bounds for the streaming set cover problem*, Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 371–383, 2016.

[4] Indyk, Piotr and Mahabadi, Sepideh and Rubinfeld, Ronitt and Vakilian, Ali and Yodpinyanee, Anak, *Set cover in sub-linear time*, Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 2467–2486, 2018.

[5] Dubhashi, Devdatt P and Panconesi, Alessandro, *Concentration of measure for the analysis of randomized algorithms*, 2009, Cambridge University Press
[6] Assadi, Sepehr and Chen, Yu and Khanna, Sanjeev, *Sublinear algorithms for (Δ+ 1) vertex coloring*, Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, 767–786, 2019.
[7] Yao, Andrew Chi-Chin, *Probabilistic computations: Toward a unified measure of complexity*, 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), 222–227, 1977.
[8] Goldreich, Oded and Goldwasser, Shari and Ron, Dana, *Property testing and its connection to learning and approximation*, Journal of the ACM (JACM), 45, 4, 653–750, 1998.
[9] Chen, Yu and Khanna, Sanjeev and Tan, Zihan, *Sublinear Algorithms and Lower Bounds for Estimating MST and TSP Cost in General Metrics*, arXiv preprint arXiv:2203.14798, 2022.
[10] Goldreich, Oded and Ron, Dana, *Property testing in bounded degree graphs*, Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, 406–415, 1997.
[11] Guy E. Blelloch and Jeremy T. Fineman and Julian Shun, *Greedy sequential maximal independent set and matching are parallel on average*, 24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '12, Pittsburgh, PA, USA, June 25-27, 2012, 308–317, 2012.
[12] Soheil Behnezhad, *Time-Optimal Sublinear Algorithms for Matching and Vertex Cover*, 62nd IEEE Annual Symposium on Foundations of Computer Science, 873–884, 2021.
[13] Zelikovsky, Alexander, *Better approximation bounds for the network and Euclidean Steiner tree problems*, Technical Report CS-96-06, Department of Computer Science, University of Virginia 1996.
[14] Goemans, Michel X and Olver, Neil and Rothvoß, Thomas and Zenklusen, Rico, *Matroids and integrality gaps for hypergraphic steiner tree relaxations*, Proceedings of the forty-fourth annual ACM symposium on Theory of computing, 1161–1176, 2012.
[15] Byrka, Jaroslaw and Grandoni, Fabrizio and Rothvoß, Thomas and Sanita, Laura, *An improved LP-based approximation for Steiner tree*, Proceedings of the forty-second ACM symposium on Theory of computing, 583–592, 2010.
[16] Garey, Michael R and Johnson, David S, *Computers and intractability*, 1979.
[17] Gilbert, Edgar N and Pollak, Henry O, *Steiner minimal trees*, SIAM Journal on Applied Mathematics, 16, 1, 1–29, 1968.
[18] Karpinski, Marek and Zelikovsky, Alexander, *New approximation algorithms for the Steiner tree problems*, Journal of Combinatorial Optimization, 1, 1, 47–65, 1997.
[19] Zelikovsky, Alexander Z, *An 11/6-approximation algorithm for the network Steiner problem*, Algorithmica, 9, 5, 463–470, 1993.
[20] Robins, Gabriel and Zelikovsky, Alexander, *Tighter bounds for graph Steiner tree approximation*, SIAM Journal on Discrete Mathematics, 19, 1, 122–134, 2005.
[21] Hauptmann, Mathias and Karpiński, Marek, *A compendium on Steiner tree problems*, Inst. für Informatik 2013.
[22] Chlebík, Miroslav and Chlebíková, Janka, *The Steiner tree problem on graphs: Inapproximability results*, Theoretical Computer Science, 406, 3, 207–214, 2008.
[23] Traub, Vera and Zenklusen, Rico, *Local search for weighted tree augmentation and Steiner tree*, Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 3253–3272, 2022.
[24] Czumaj, Artur and Sohler, Christian, *Estimating the weight of metric minimum spanning trees in sublinear time*, SIAM Journal on Computing, 39, 3, 904–922, 2009.
[25] Chazelle, Bernard and Rubinfeld, Ronitt and Trevisan, Luca, *Approximating the minimum spanning tree weight in sublinear time*, SIAM Journal on computing, 34, 6, 1370–1379, 2005.
[26] Borchers, Al and Du, Ding-Zhu, *The k-Steiner ratio in graphs*, SIAM Journal on Computing, 26, 3, 857–869, 1997.
[27] Zelikovsky, Alexander Z, *An 11/6-approximation algorithm for the network Steiner problem*, Algorithmica, 9, 5, 463–470, 1993.
[28] Du, Ding-Zhu, *On component-size bounded Steiner trees*, Discrete applied mathematics, 60, 1-3, 131–140, 1995.