Near-Optimal Distributed Tree Embedding

Mohsen Ghaffari
MIT
ghaffari@csail.mit.edu

Christoph Lenzen
MIT
clenzen@csail.mit.edu

Abstract

Tree embeddings are a powerful tool in the area of graph approximation algorithms. Roughly speaking, they transform problems on general graphs into much easier ones on trees. Fakcharoenphol, Rao, and Talwar (FRT) [STOC'04] present a probabilistic tree embedding that transforms n-node metrics into (probability distributions over) trees, while *stretching* each pairwise distance by at most an $O(\log n)$ factor in expectation. This $O(\log n)$ stretch is optimal.

Khan et al. [PODC'08] present a distributed algorithm that implements FRT in $O(\mathrm{SPD} \log n)$ rounds, where SPD is the *shortest-path-diameter* of the weighted graph, and they explain how to use this embedding for various distributed approximation problems. Note that SPD can be as large as $\Theta(n)$, even in graphs where the hop-diameter D is a constant. Khan et al. noted that it would be interesting to improve this complexity. We show that this is indeed possible.

More precisely, we present a distributed algorithm that constructs a tree embedding that is essentially as good as FRT in $\tilde{O}(\min\{n^{0.5+\varepsilon}, \mathrm{SPD}\} + D)$ rounds, for any constant $\varepsilon > 0$. A lower bound of $\tilde{\Omega}(\min\{n^{0.5}, \mathrm{SPD}\} + D)$ rounds follows from Das Sarma et al. [STOC'11], rendering our round complexity near-optimal.

1 Introduction and Related Work

Metric embeddings are a versatile technique in centralized approximation algorithms. Given an arbitrary metric space on n points—i.e., a weighted graph on n nodes with distances being the metric—metric embeddings transform it into a "nicer" metric space while incurring only a small *distortion*. A basic example is Bourgain's theorem [5], which shows that it is possible to embed into ℓ_2 with $O(\log n)$ distortion. The general approach for using metric embeddings in approximation algorithms is as follows: (1) using the embedding, transform the given problem instance to one in a more convenient metric space (i.e., nicer graph); (2) solve the simpler problem; (3) transform the solution back to one of the original instance. See [14, 18] for surveys.

Tree embeddings are a particularly useful branch of metric embeddings, which "transform" general graphs into trees. This is especially attractive, because finding solutions on trees is often quite easy, if not trivial. Not surprisingly, the approach has a caveat: one cannot hope for a small distortion when (deterministically) embedding into a tree; for example, transforming an n-node cycle to any tree will incur an $\Omega(n)$ distortion on at least one edge. But not all the hope is lost, as there is still the option of embedding probabilistically. Indeed, a beautiful line of work [1–3, 9] shows this to be feasible, obtaining successively better distortions, and ending with the algorithm of Fakcharoenphol, Rao, and Talwar (FRT) [9], which achieves $O(\log n)$ distortion. More precisely, the FRT algorithm maps any n-point metric into a tree drawn from a certain probability distribution so that each pairwise distance is *stretched* by a factor $O(\log n)$ in expectation. This $O(\log n)$ distortion is existentially optimal, as demonstrated by expander graphs [3].

The fact that graph problems are often easier on trees is not particular to centralized algorithms. Hence, it is natural to expect that *tree embeddings* should be helpful in distributed algorithms as well. Actualizing this intuition, Khan et al. [15] showed how to implement FRT distributedly and use it to get distributed approximation algorithms for a number of graph problems. The distributed tree embedding of Khan et al. works in $O(\mathrm{SPD} \log n)$ rounds of the CONGEST model, where SPD denotes the *shortest-path-diameter*. CONGEST is the standard model for distributed computation, in which each node can send one B-bit size message per round to each of its neighbors; typically, $B = O(\log n)$. The *shortest-path-diameter* SPD is the minimum integer $h \in \mathbb{N}^+$ such that for any pair of nodes v and u, there is a least-weight path between v and u that has at most h hops. Note that SPD can be much larger than the hop diameter D, up to factor $\Theta(n)$; see, e.g., Figure 1.

Khan et al. noted that it would be interesting to improve the round complexity. This is particularly intriguing in light of the developments in the general area of distributed approximation algorithms. On the lower bound side, an elegant sequence of papers [7, 8, 21] show $\tilde{\Omega}(D + \sqrt{n})$ rounds to be necessary for a wide range of (approximation) problems, including those for which tree embeddings can be useful. Here, D is the hop diameter of the network graph. On the upper bound side, in the last couple of years approximation algorithms with round complexity $\tilde{O}(D+\sqrt{n})$, or close to it, have been developed for a number of different graph problems [11–13, 16, 17, 19]. Consequently, it is intriguing to ask

"Is there an $\tilde{O}(D+\sqrt{n})$-round tree embedding algorithm?"

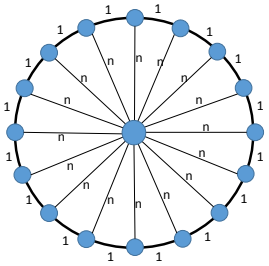

Figure 1: An example where $\mathrm{SPD} \gg D$. Here $D = 2$ and $\mathrm{SPD} \approx n/2$. Numbers on edges show their weights.

We consider answering this question important as a positive result would add tree embeddings to the set of our $\tilde{O}(D + \sqrt{n})$-round tools and extend the range of problems in the $\tilde{O}(D + \sqrt{n})$-round category to those which can be solved via tree embedding.

1.1 Our Contribution

We show that there is a distributed algorithm that provides a probabilistic tree embedding that is essentially as good as FRT—i.e., with only a constant factor larger stretch—in almost $\tilde{O}(D + \sqrt{n})$ rounds.

Theorem 1.1 (INFORMAL). *For any $\varepsilon > 0$, a probabilistic tree embedding with expected stretch of $O(\log n/\varepsilon)$ can be computed in $\tilde{O}(\min\{n^{0.5+\varepsilon}, \mathrm{SPD}\} + D)$ rounds of the* CONGEST *model.*

The formal version specifying how the embedding is represented distributedly is presented in Theorem 4.11. As mentioned, this result is near-optimal in both stretch and round complexity: the former must be $\Omega(\log n)$ [3], and we explain in Appendix A that [7] yields that the latter must be $\tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\} + D)$.

1.2 Overview

Here, we explain the key ideas of our construction. For brevity, the description of FRT is deferred to Section 3.

FRT = Least Elements (LE) Lists: Given a weighted graph $G = (V, E, W)$, computing FRT's probabilistic tree embedding of G boils down to the following: (1) choose a permutation π of V uniformly at random; (2) for each node $v \in V$ and each distance d, find the node w within distance d of v that appears first in the permutation π.

For each node v, this generates a list of nodes with one entry for each distance d. Note that the π-indices of the nodes in the list are decreasing as a function of d. The list can be compressed by only keeping the entry with the minimum distance d for each node w in the list. The compressed lists are called Least Elements (LE) lists [6]. See Figure 2 for an example.

dist.	node
0	9
2	8
3	5
4	3
6	1

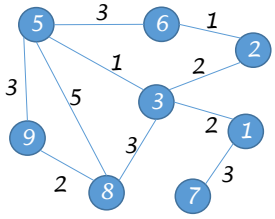

Figure 2: The LE list for node 9. Nodes are labeled 1 to 9 randomly, edges are labeled by their weights.

Distributed LE-list Computation: Khan et al. [15] present a neat method for computing LE lists distributedly. Their algorithm runs in iterations; in each iteration, each node sends its whole current (compressed) list to its neighbors. Initially, each list contains only the node itself at distance 0. In each round, each node updates its lists using the received ones and the distances from the sending neighbors. After at most SPD iterations, we get the correct LE lists. A key observation of [15] is that, due to the random order π, in each iteration, each list will contain at most $O(\log n)$ entries, with high probability (w.h.p.). Thus, each of the at most SPD iterations can be performed in $O(\log n)$ rounds, which translates to a total of $O(\mathrm{SPD} \log n)$ rounds.

A Strawman Idea: LE lists can be determined easily if we have all-to-all distances. Since we know how to get a multiplicative approximation of all-to-all distances in time close to $\tilde{O}(D + \sqrt{n})$ [16], a natural idea is to use these approximates to construct an FRT-like embedding. However, this idea does not go far, mainly because multiplicative approximates do not satisfy (any approximation of) the triangle inequality.

Our Approach In a Nutshell: We construct a *virtual* graph G' whose distances approximate those of G and has shortest-path diameter $S_{G'}$ at most $\tilde{O}(\sqrt{n})$. Note that distances in a graph always satisfy the triangle inequality, which enables us to apply the FRT construction. However, since G' is a virtual graph, we cannot readily use the algorithm by Khan et al. [15] to achieve a time complexity of $\tilde{O}(S_{G'}) = \tilde{O}(\sqrt{n})$.

We resolve this by entangling the construction of G' with the computation of LE lists. More concretely, we pick the first $\Theta(\sqrt{n})$ nodes in the random order π of FRT, call them $\mathcal{S}$, find factor $\varrho = O(1/\varepsilon)$ approximations of distances among nodes of $\mathcal{S}$. This part uses the spanner construction of Baswana and Sen [4] and its adaptation in [16]. We set the G'-distances among $\mathcal{S}$ equal to these approximations by adding direct virtual edges of the corresponding weight, while the original edges of G are added to G' with weight raised by factor ϱ. As now the approximated distances between nodes in $\mathcal{S}$ are *exact* G'-distances, we can directly compute the G'-LE lists of the nodes in $\mathcal{S}$. Here it is crucial that the nodes in $\mathcal{S}$ are prioritized by π, implying that their lists will only contain nodes in $\mathcal{S}$. Furthermore, for any pair of nodes for which a least-weight path has at least roughly $\sqrt{n} \log n$ hops, w.h.p. one such path has a node from $\mathcal{S}$ within its first $O(\sqrt{n} \log n)$ hops. Thus, it suffices to run the LE list computation, jump-started with the complete lists on $\mathcal{S}$, only for $\tilde{O}(\sqrt{n})$ iterations. The fact that we have entangled the (random) construction of G' with the (random) permutation π of FRT creates some probabilistic dependency issues. However, by opening up the analysis of FRT and the LE list computation, we show that this does affect neither the stretch nor the running time by more than factor 2.

2 Preliminaries

Model: The network is represented by a connected, simple, and weighted graph $G = (V, E, W)$ of $n := |V|$ nodes, with edge weights $W : E \to \mathbb{N}$ bounded polynomially in n. Initially each node knows the weights of its incident edges. We use the CONGEST model [20]: communication occurs in synchronous rounds, where in each round $B = O(\log n)$ bits can be sent in each direction of each edge.

Each node independently picks an $O(\log n)$-bit *identifier* (*ID* in short), uniformly at random. These are the identifiers that we will use in the remainder of the paper. We use random IDs because they readily give us a uniformly random ordering π of nodes. We use notation $v < w$ to mean that the random ID of node v is smaller than that of node w. It is easy to see that, with high probability, the random ID picked by each node is unique, and we assume throughout the paper that this holds true. Here, we use the phrase "with high probability" (w.h.p.) to indicate that an event occurs with probability at least $1 - 1/n^c$, for an arbitrary constant $c > 0$ fixed in advance.

Graph-Theoretic Notations:

- For a node v, denote the set of its neighbors by $\mathcal{N}(v)$.
- For a path $p = (v_0, \ldots, v_h)$, define its length $\ell(p) := h$ and its weight $W(p) := \sum_{i=1}^{h} W(v_{i-1}, v_i)$.
- For $v, w \in V$, denote by P_{vw} the set of all paths starting at v and ending at w.
- The hop distance of $v, w \in V$ is defined as $\mathrm{hd}(v, w) := \min_{p \in P_{vw}}\{\ell(p)\}$.
- The (hop) diameter of G is $D := \max_{v,w \in V}\{\mathrm{hd}(v, w)\}$.
- The (weighted) distance of $v, w \in V$ is given by $\mathrm{wd}(v, w) := \min_{p \in P_{vw}}\{W(p)\}$.
- The weighted diameter of G is $\mathrm{WD} := \max_{v,w \in V}\{\mathrm{wd}(v, w)\}$.
- The shortest-path diameter of G is $\mathrm{SPD} := \max_{v,w \in V}\{\min\{\ell(p) | p \in P_{vw} \wedge W(p) = \mathrm{wd}(v, w)\}\}$.

For brevity, we use the conventions that "diameter" denotes the hop diameter, but "distance" refers to *weighted* distances. When talking of graphs other than G, we subscript the above notations by the respective graph.

3 Recap: FRT, Least Element Lists, and Spanners

3.1 The FRT Probabilistic Tree Embedding

Given a weighted graph $G = (V, E, W)$, FRT randomly constructs a tree $T = (V_T, E_T, W_T)$ such that there is a mapping $\mathfrak{M}$ from V to leaves of T, and for each pair of nodes $u, v \in V$, we have

- $\mathrm{wd}(u, v) \le \mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(v))$, and
- $\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u))] \le \mathrm{wd}(u, v) \cdot O(\log n)$.

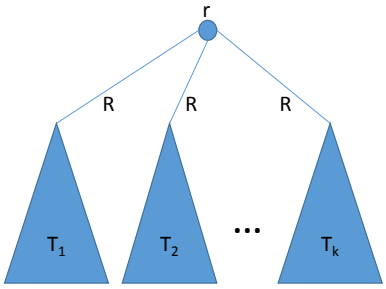

Figure 3: FRT's recursion.

FRT–An Intuitive Description: The construction can be viewed as a tree of hierarchical decompositions. The key decomposition step is as follows: Pick $R \in [\mathrm{WD}/4, \mathrm{WD}/2]$ uniformly at random. For each node w, define its (exclusive) ball

$$B(w) := \{v \in V \mid \mathrm{wd}(v, w) \le R \wedge \forall w' \in V : \mathrm{wd}(v, w') \le R \Rightarrow w \le w'\}.$$

Recall from Section 2 that the notation $w \le w'$ means w has a smaller random ID compared to w'. We recursively create a tree embedding T_i for each subgraph of G induced by a nonempty ball $B(w)$. Finally, add a root node r and connect the roots of trees T_i to r via edges of weight R.

FRT–A Formal Description: The whole structure of the FRT tree can be succinctly described as follows. Choose a uniformly random $\beta \in [1, 2)$. Denote by $L := \lceil \log \mathrm{WD} \rceil + 1$ the maximum level of the tree. For each $v \in V$ and each $i \in \{0, \ldots, L\}$, let node $v_i \in V$ minimize the ID among the set $\{w \in V \mid \mathrm{wd}(v, w) \le \beta 2^{i-1}\}$.

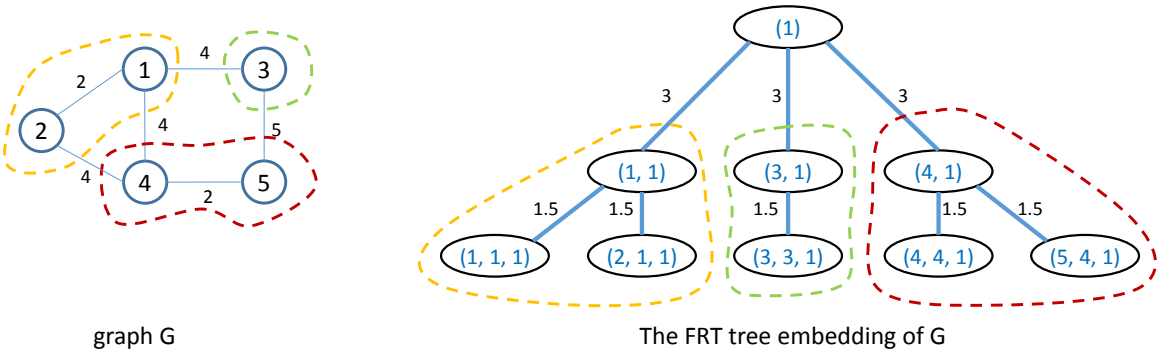

Figure 4: A simple example of an FRT tree.

Note that v_i is a function[1] of v. In particular, $v_0 = v$ and $v_L = \min V$ is the first node in the order π. The node set V_T of the tree is $\{(v_i, \ldots, v_L) \,|\, v \in V \wedge i \in \{0, \ldots, L\}\}$. Note that each different sequence $(v_i, \ldots, v_L)$ starting with v_i denotes a distinct "copy" of $v_i \in V$. For each tree node $(v_i, \ldots, v_L)$ with $i < L$, its parent is the tree node $(v_{i+1}, \ldots, v_L)$, and the weight of the edge connecting them is $\beta 2^i$. Finally, we have $\mathfrak{M}(v) := (v_0, \ldots, v_L)$. Thus, the node set V is mapped to the leaf set of the tree. Figure 4 shows an example.

The fact that, for each $v, w \in V$, $\mathrm{wd}(v, w) \leq \mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(w))$ can be easily verified. The main result of Fakcharoenphol, Rao, and Talwar [10] is the probabilistic upper bound.

Theorem 3.1 ([10], Theorem 2)**.** *For the embedding described above, it holds for each $v, w \in V$ that $\mathbb{E}[d_T(\mathfrak{M}(v), \mathfrak{M}(w))] \in O(\log n) \cdot \mathrm{wd}(v, w)$.*

3.2 Least Element Lists

The FRT embedding can be implicitly encoded via a data structure called *Least Element lists* [6].

Least Element (LE) Lists: For each node v, its LE list is

$$L_v := \{(w, \mathrm{wd}(v, w)) \in V \times \mathbb{N}_0 \,|\, \nexists u \in V : (u < w \wedge \mathrm{wd}(v, u) < \mathrm{wd}(v, w))\}.$$

Given the LE lists, each node v can easily compute the nodes v_i, for $i \in \{1, \ldots, L\}$, from its LE list L_v. This is because, for any given distance d, node v can recover the node of smallest ID within distance d as the node w of smallest ID satisfying that $(w, \mathrm{wd}(v, w)) \in L_v$ and $\mathrm{wd}(v, w) \leq d$. Moreover, these lists allow us to determine the next hop on a least-weight path from v to v_i locally.

Observation 3.2. *If $(w, \mathrm{wd}(v, w)) \in L_v$ for $w \neq v$, then $u \in \mathcal{N}(v)$ exists s.t. $(w, \mathrm{wd}(v, w) - W(v, u)) \in L_u$. Hence, if for each $w \in V$, $w \neq v$, so that $(w, \mathrm{wd}(v, w)) \in L_v$ we choose $p_v(w) \in \mathcal{N}(v)$ with $(w, \mathrm{wd}(v, w) - W(v, p_v(w))) \in L_{p_v(w)}$, then the edges $(v, p_v(w))$ form a shortest-path tree rooted at w.*

3.3 Distributed Computation of Least Element Lists

Next, we explain the distributed algorithm of Khan et al. [15] for computing LE lists:

1. Each $v \in V$ initializes $L_v^{(0)} := \{(v, 0)\}$. Set $i := 0$.
2. All nodes v do the following *iteration* in parallel:

[1] A better notation might be $c_i(v)$, indicating that this is the level i center of node v. However, for brevity we write $v_i = c_i(v)$.

(a) Send $L_v^{(i)}$ to all neighbors.
(b) Set $L_v^{(i+1)} := L_v^{(i)}$.
(c) For all $w \in \mathcal{N}(v)$ and $(u, d) \in L_w^{(i)}$, set $L_v^{(i+1)} := L_v^{(i+1)} \cup \{(u, d + W(v, w))\}$.
(d) Scan $L_v^{(i+1)}$ in ascending order of distances (i.e., second entries) and delete each entry for which the ID (i.e., the first entry) is not larger than all IDs previously encountered in the scan.
(e) Set $i := i + 1$.

while $\exists v \in V$ so that $L_v^{(i)} \neq L_v^{(i-1)}$.

3. Each $v \in V$ returns $L_v^{(i)} = L_v$.

From the definition of LE lists, the following observations regarding this algorithm are straightforward.

Observation 3.3. *If $(w, \mathrm{wd}(v, w)) \in L_v$, then $(w, \mathrm{wd}(v, w))$ is not deleted from $L_v^{(i)}$ during its scan.*

Observation 3.4. *For $i \in \mathbb{N}_0$, suppose $(w, \mathrm{wd}(v, w)) \in L_v$ and $(w, \mathrm{wd}(u, w)) \in L_u^{(i)}$ for $u \in \mathcal{N}(v)$ on a least-weight path from v to w. Then $(w, \mathrm{wd}(v, w))) \in L_v^{(i+1)}$.*

Observations 3.2, 3.3, and 3.4 essentially imply the following lemma, whose proof appears in Appendix B.

Lemma 3.5. *The above algorithm computes correct LE lists. It terminates after at most* $\mathrm{SPD} + 1$ *iterations.*

Remark 3.6. *If node $v \in V$ also memorizes which neighbor sent the (first) message causing it to insert an entry into L_v, the least-weight paths indicated by the respective pointers have minimal hop count, and the trees implied by Observation 3.2 have minimal depth (which is bounded by* SPD*).*

The remaining analysis essentially boils down to showing that the lists are always short, i.e., have $O(\log n)$ entries w.h.p. in each iteration. We remark that our analysis fixes a technical issue with the analysis in [15]; however, the key idea is the same. We refer to Appendix B for details.

Lemma 3.7 ([15], Lemma 5)**.** *For each $v \in V$ and each $i \in \mathbb{N}_0$, $|L_v^{(i)}| \in O(\log n)$ w.h.p.*

The results explained above can be put together to prove the following theorem.

Theorem 3.8. *The LE lists can be computed within $O(\mathrm{SPD} \log n)$ rounds w.h.p.*

3.4 Spanners and Skeletons

In our algorithm, we will make use of known techniques for constructing spanners and skeletons. Here, we briefly review these tools. We note that the description is adapted to best suit our application.

For $\varrho \geq 1$, a (multiplicative) ϱ-spanner of a graph $H = (V_H, E_H, W_H)$ is a graph $H_S = (V_H, E_S, W_S)$ with $E_S \subseteq E_H$, $W_S(e) = W_H(e)$ for all $e \in E_S$, and $\mathrm{wd}_{H_S}(v, w) \leq \varrho\, \mathrm{wd}_H(v, w)$ for all $v, w \in V_H$. Ideally, we want both ϱ and $|E_S|$ to be as small as possible.

We will make use of a spanner construction on a certain virtual graph. For $\mathcal{S} \subseteq V$, the h-hop $\mathcal{S}$-skeleton of G is defined as the weighted graph $G_{\mathcal{S},h} := (\mathcal{S}, E_{\mathcal{S},h}, W_{\mathcal{S},h})$ with $E_{\mathcal{S},h} := \{\{s, t\} \in \binom{\mathcal{S}}{2} \mid \mathrm{hd}(s, t) \leq h\}$ and $W_{\mathcal{S},h}(s, t) := \min\{W(p) \mid p \in P_{st} \wedge \ell(p) \leq h\}$ for each $\{s, t\} \in E_{\mathcal{S},h}$. In words, the graph has node set $\mathcal{S}$ and for each $s, t \in \mathcal{S}$ an edge of weight $\mathrm{wd}^{(h)}(s, t)$ iff s and t are in hop distance at most h.

For the purposes of this paper, we confine ourselves to the special case that each node is sampled into $\mathcal{S}$ independently with probability $1/\sqrt{n}$ and that $h := c\sqrt{n} \log n$ for a sufficiently large constant c. Under these constraints, the skeleton preserves weighted distances, w.h.p.

Lemma 3.9 ([16], Lemma 4.6)**.** *If nodes are sampled into $\mathcal{S}$ with independent probability $1/\sqrt{n}$ and $h := c\sqrt{n} \log n$ for a sufficiently large constant c, then $\mathrm{wd}_{G_{\mathcal{S},h}}(s, t) = \mathrm{wd}(s, t)$ for all $s, t \in \mathcal{S}$ w.h.p.*

The hop-distance h up to which we need to explore paths in G to "find" the edges in $E_{G_{\mathcal{S},h}}$ is in $\tilde{O}(\sqrt{n})$. Furthermore, because $|\mathcal{S}| \in \tilde{O}(\sqrt{n})$, $G_{\mathcal{S},h}$ can be encoded using $\tilde{O}(n)$ bits. We can further reduce this to $\tilde{O}(n^{0.5+\varepsilon})$ bits by constructing a spanner of $G_{\mathcal{S},h}$, sacrificing a factor of $O(1/\varepsilon)$ in the accuracy of the distances.

Theorem 3.10 ([16], Theorem 4.10)**.** *Suppose nodes are sampled into $\mathcal{S}$ independently with probability $1/\sqrt{n}$ and $h := c\sqrt{n}\log n$ for a sufficiently large constant c. Then, for any $k \in \mathbb{N}$, w.h.p. a $(2k-1)$-spanner of $G_{\mathcal{S},h}$ can be computed and made known to all nodes in $\tilde{O}(n^{1/2+1/(2k)} + D)$ rounds. Furthermore, for each $s, t \in \mathcal{S}$, there is a unique $p \in P_{st}$ of weight $W(p) \leq (2k-1)\,\mathrm{wd}(s,t)$ so that each node on p knows that it is on p as well as which of its neighbors is the next node on p (as a function of the identifier of t).*

This is achieved by simulating the Baswana-Sen spanner construction [4] on $G_{\mathcal{S},h}$ using a truncated version of the multi-source Bellman-Ford algorithm. For details and proofs we refer to [16].

4 Fast Distributed Tree Embedding

In this section, we explain our algorithm. We first give an intuitive explanation of what are our key ideas. Then, we present the algorithm and its correctness, running time, and approximation analysis.

4.1 Key Ideas

When computing LE lists according to the algorithm by Khan et al. [15], information spreads along least-weight paths. For most of the nodes, however, the induced shortest-path trees (cp. Observation 3.2) will be fairly shallow: Let $\mathcal{S}$ be the set of nodes which their ID is in the first $1/\sqrt{n}$ fraction of range of possible IDs. Recall from Section 2 that we assign IDs uniformly and independently at random. Thus, $\Pr[v \in S] = 1/\sqrt{n}$. When following a least-weight path, w.h.p. one will encounter a node in $\mathcal{S}$ after at most $O(\sqrt{n}\log n)$ hops. Such a node will never have any entries corresponding to nodes of larger IDs. By the union bound and Observation 3.2, we get that the trees rooted at nodes in $V \setminus \mathcal{S}$ have depth $O(\sqrt{n}\log n)$ w.h.p.

Observation 4.1. *Let $\mathcal{S}$ be the set of nodes with ID in the first $1/\sqrt{n}$ fraction of the ID range. For each $v \in V \setminus \mathcal{S}$, the depth of a shortest-path tree rooted at v whose nodes are the set $\{w \in V \mid (v, \mathrm{wd}(w,v)) \in L_w\}$ is $O(\sqrt{n}\log n)$ w.h.p.*

For nodes in $\mathcal{S}$, there is no such guarantee. In fact, if a graph contains a very light path of $n-1$ hops and otherwise only edges of large weight, it is certain that the tree rooted at the minimum-ID node has depth $\Omega(n)$, even if the hop diameter D is very small. Nonetheless, the property that on any path a node from $\mathcal{S}$ will be encountered within $O(\sqrt{n}\log n)$ hops w.h.p. still applies. Hence, once the LE lists of the nodes in $\mathcal{S}$ are determined, the algorithm from the Section 3.3 will complete after $O(\sqrt{n}\log n)$ additional iterations w.h.p.

Observation 4.2. *If for some $i \in \mathbb{N}_0$ and all $s \in \mathcal{S}$ it holds that $L_s^{(i)} = L_s$, then $L_v^{(i')} = L_v$ for all $v \in V$ and $i' \in i + O(\sqrt{n}\log n)$ w.h.p.*

In summary, the problem boils down to computing the LE lists for the small subset $\mathcal{S} \subset V$ quickly. We do not know how to do this exactly in sublinear time, i.e., $\tilde{o}(n)$ rounds. Consequently, we will make use of approximation. Recall that, since the IDs are uniformly random, $\mathcal{S}$ is a uniformly random subset of V containing each node with independent probability $1/\sqrt{n}$. This is exactly the precondition required in the skeleton and spanner constructions given by Lemma 3.9 and Theorem 3.10. Thus, in $\tilde{O}(n^{1/2+\varepsilon} + D)$ rounds, we can make *ϱ-approximate* distances, for $\varrho \in O(1/\varepsilon)$, between nodes in $\mathcal{S}$ global knowledge. More precisely, these approximations are induced by the distance metric of a spanner of the $O(\sqrt{n}\log n)$-hop $\mathcal{S}$-skeleton. Hence, we can compute *exact* LE lists of this spanner locally. Using these lists instead of those for G *approximately* preserves distances.

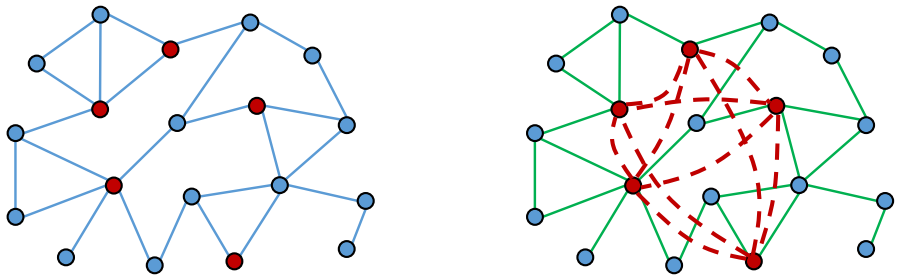

Figure 5: Virtual graph construction. Red nodes are in $\mathcal{S}$, i.e., their ID is in the top $1/\sqrt{n}$ fraction of the ID range. Dotted red lines indicate virtual edges in G', whose weights are the ϱ-approximation of their endpoints' distance in G. Green edges show edges from G that are not connecting two nodes from $\mathcal{S}$. In G', their weight is by factor ϱ larger than in G. Hence, for each $s, t \in \mathcal{S}$, the virtual edge $\{s, t\}$ is a least-weight path from s to t in G'.

Next, we want to use these lists and Observation 4.2 to complete the computation of LE lists for the remaining set $V \setminus \mathcal{S}$ quickly. However, unfortunately we did not compute LE lists for G. To address this issue, we consider the virtual graph $G' = (V, E', W')$, where $E' := E \cup \binom{\mathcal{S}}{2}$, and $W'(e')$ is the distance of s and t in the spanner iff $e' = \{s, t\}$ for some $s, t \in \mathcal{S}$ and $W(e') = \varrho W(e)$ otherwise. In G', the spanner distances between $s, t \in \mathcal{S}$ are the exact distances, and for any $v, w \in V$, $\mathrm{wd}(v, w) \leq \mathrm{wd}_{G'}(v, w) \leq \varrho\, \mathrm{wd}(v, w)$. Intuitively, without distorting distances by more than factor ϱ, we have ensured that the LE lists of nodes in $\mathcal{S}$ we determined from the spanner are their *exact* LE lists in G', and by Observation 4.2, we can compute the LE lists of nodes in $V \setminus \mathcal{S}$ quickly. Finally, we would like to argue that the computed lists are those of an FRT embedding of G', and because G' satisfies that $\mathrm{wd}(v, w) \leq \mathrm{wd}_{G'}(v, w) \leq \varrho\, \mathrm{wd}(v, w)$ w.h.p., the overall expected distortion is $O(\varrho \log n)$ in expectation.

Is it that simple? Almost, but not quite. The issue with the above simplistic reasoning is that it ignores dependencies. Since G' depends on $\mathcal{S}$, the permutation of V induced by the ranks is not independent of the topology of G', and therefore it is not obvious that the bound on the expected distortion of the FRT algorithm applies. Similarly, the the statement of Lemma 3.7 that (intermediate) LE lists are w.h.p. of size $O(\log n)$ relies on the independence of the permutation from the topology of G'. Both of these issues can be resolved, by argueing about $\mathcal{S}$ and $V \setminus \mathcal{S}$ separately; the total orders the IDs induce on $\mathcal{S}$ and $V \setminus \mathcal{S}$, respectively, are independent of what nodes are in $\mathcal{S}$ and thus the topology of G'.

4.2 Our Algorithm: Constructing the Virtual Graph G' and Computing its LE Lists

Here we describe our algorithm. This algorithm uses a parameter $k \in \mathbb{N}$ that can be set arbitrarily. For our main result, one should think of k as $k = \lceil 1/\varepsilon \rceil$.

1. Construct any BFS tree, determine its depth $\hat{D} \in \Theta(D)$ and n (the latter by aggregation on the tree), and make these values known to all nodes.
2. Put the nodes whose random ID is in the first $1/\sqrt{n}$ fraction of the ID range in set $\mathcal{S}$.
3. Set $h := c\sqrt{n} \log n$ for a sufficiently large constant c and $\rho := 2k - 1$. Construct a ρ-spanner $\tilde{G}$ of $G_{\mathcal{S},h}$ in $\tilde{O}(n^{1/2+1/(2k)} + D)$ rounds, using the algorithm given by Theorem 3.10.
4. Define the virtual graph $G' := (V, E', W')$ as follows:
   - $E' := E \cup \binom{\mathcal{S}}{2}$.
   - For each $s, t \in \mathcal{S}$, set $W'(s, t) := W_{\tilde{G}}(s, t)$.
   - For each $e \in E' \setminus \binom{\mathcal{S}}{2}$, set $W(e') := \rho W(e)$.

5. Each $s \in \mathcal{S}$ locally computes $L_v^{\mathcal{S}}$, its LE list for the metric on $\mathcal{S}$ induced by distances in $\tilde{G}$.
6. Each node $s \in \mathcal{S}$ initializes $L_s^{(0)} := L_v^{\mathcal{S}}$. Nodes $v \in V \setminus \mathcal{S}$ initialize $L_v^{(0)} := \{(v, 0)\}$. The algorithm from Section 3.3 is run on G, however with the lists $L_v^{(0)}$ initialized as just specified.
7. Return the computed lists.

4.3 Correctness Analysis

We first show that the algorithm computes the desired LE lists.

Lemma 4.3. *W.h.p.,* $\mathrm{wd}_{\tilde{G}}(s,t) = \mathrm{wd}_{G'}(s,t)$ *for all* $s, t \in \mathcal{S}$.

Proof. Theorem 3.10 guarantees that w.h.p., $\tilde{G}$ is a ρ-spanner of $G_{\mathcal{S},h}$. By Lemma 3.9, w.h.p. $\mathrm{wd}_{G_{\mathcal{S},h}}(s,t) = \mathrm{wd}(s,t)$ for all $s, t \in \mathcal{S}$. In the following, we condition on both events occuring. Therefore, for any $s, t \in \mathcal{S}$,

$$\mathrm{wd}_{G'}(s,t) \leq \mathrm{wd}_{\tilde{G}}(s,t) \leq \rho\, \mathrm{wd}_{G_{\mathcal{S},h}}(s,t) = \rho\, \mathrm{wd}(s,t).$$

It remains to prove that $\mathrm{wd}_{G'}(s,t) \geq \mathrm{wd}_{\tilde{G}}(s,t)$. To this end, consider any path $p = (v_0 = s, v_1, \ldots, v_{\ell(p)} = t)$ in G'. It decomposes into subpaths $p = p_1 \circ p_2 \circ \ldots \circ p_m$ (for some $m \leq \ell(p)$) so that each p_i, $i \in \{1, \ldots, m\}$, starts and ends at a node in $\mathcal{S}$ and all its internal nodes are in $V \setminus \mathcal{S}$. Therefore, either $p_i = (s_i, t_i)$ for some $s_i, t_i \in \mathcal{S}$ and $W'(p_i) = \mathrm{wd}_{\tilde{G}}(s_i, t_i)$, or $p_i = (s_i, \ldots, t_i)$ consists of edges from E only. The latter implies that

$$W'(p_i) = \rho W(p_i) \geq \rho\, \mathrm{wd}(s_i, t_i) = \rho\, \mathrm{wd}_{G_{\mathcal{S},h}}(s_i, t_i) \geq \mathrm{wd}_{\tilde{G}}(s_i, t_i).$$

Thus, in both cases, $W'(p_i) \geq \mathrm{wd}_{\tilde{G}}(s_i, t_i)$. By repeated application of the triangle inequality ($\mathrm{wd}_{\tilde{G}}$ is a metric), we conclude that

$$\mathrm{wd}_{\tilde{G}}(s,t) \leq \sum_{i=1}^{m} \mathrm{wd}_{\tilde{G}}(s_i, t_i) \leq \sum_{i=1}^{m} W'(p_i) = W'(p).$$

Since p was an arbitrary s-t path in G', we conclude that indeed $\mathrm{wd}_{G'}(s,t) \geq \mathrm{wd}_{\tilde{G}}(s,t)$. □

Corollary 4.4. *W.h.p., the LE lists computed locally in Step 5 are the LE lists for* G' *of nodes in* $\mathcal{S}$.

Corollary 4.5. *W.h.p., the above algorithm returns LE lists for the graph* G' *specified in Step 4.*

4.4 Running Time Analysis

Clearly, the first step of the algorithm requires $O(D)$ rounds. The other steps that do not solely consist of local computations are Steps 3 and 6. By Theorem 3.10, the time complexity of Step 3 is $\tilde{O}(\sqrt{n}^{1/2+1/(2k)} + D)$ w.h.p. Hence, it remains to analyze the time complexity of Step 6.

Lemma 4.6. *Step 6 of the above algorithm performs* $O(\sqrt{n} \log n)$ *iterations of the LE list algorithm w.h.p.*

The proof is given in Appendix C; essentially, the lemma follows from the fact that on each path, a node from $\mathcal{S}$ is encountered every $\tilde{O}(\sqrt{n})$ hops.

Due to this lemma, it is sufficient to show that in each iteration, the lists contain $O(\log n)$ entries w.h.p.

Lemma 4.7. *For each iteration of the LE list algorithm during Step 6, all lists have* $O(\log n)$ *entries w.h.p.*

Proof. For each node $v \in V$ and each index $i \in \mathbb{N}_0$, we split its list $L_v^{(i)}$ into two parts. The head of the list $H_v^{(i)}$ consists of entries (s, d) with $s \in \mathcal{S}$ and its tail $T_v^{(i)}$ consists of entries (v, d) with $v \in V \setminus \mathcal{S}$. Consider the following virtual graph:

- Take a copy of G'.
- Add a copy of each node in $\mathcal{S}$ and connect it to the original by a 0-weight edge.
- Connect each copy of a node $s \in \mathcal{S}$ to each original node $t \in \mathcal{S} \setminus \{s\}$ by an edge of weight $W'(s,t)$.

Now initialize, for each $s \in \mathcal{S}$, $L^{(0)}_{s'} := \{(s,0)\}$, where s' is the copy of s. For all original nodes $v \in V$, set $L^{(0)}_v := \emptyset$. Observe that

(i) after one iteration of the LE list algorithm, each original node has the same head $H^{(0)}_v$ as according to the initialization in Step 6 of the algorithm;
(ii) no message from a copy of a node ever causes a change in the lists in later rounds; and
(iii) the permutation of $\mathcal{S}$ induced by the IDs is uniform and independent of the topology.

The third observation implies that Lemma 3.7 applies,[2] i.e., the list heads have $O(\log n)$ entries w.h.p. The first two observations imply that the list heads are identical to those of the iterations of the LE list algorithm performed in Step 6 (shifted by one round). Hence, $|H^{(i)}_v| \in O(\log n)$ w.h.p. for all nodes $v \in V$ and rounds i.

Now consider the list tails. Suppose the list construction algorithm was run on G, but with $L^{(0)}_s := \emptyset$ for all $s \in \mathcal{S}$. Since the ranks induce a uniformly random permutation on $V \setminus \mathcal{S}$ (that is independent of G), Lemma 3.7 applies and, in each iteration, the respective lists have $O(\log n)$ entries w.h.p. We claim that the tails $T^{(i)}_v$, $v \in V$, $i \in \mathbb{N}_0$, are always prefixes of these lists. This follows because if an entry of an (intermediate) head list causes deletion of an entry from a tail list, it can never happen that the deleted entry would affect neighbors' lists in future iterations (the head entry causing the deletion always takes precedence).

To complete the proof, for each iteration i we take the union bound over all nodes and the events that the head and tail lists are short, implying that, w.h.p., $|L^{(i)}_v| = |H^{(i)}_v| + |T^{(i)}_v| \in O(\log n)$ for all nodes $v \in V$. □

We summarize our results on the LE list computation for G'; see Appendix C for details.

Theorem 4.8. *W.h.p., the algorithm computes the LE lists of the virtual graph G' defined in Step 5 and (if suitably implemented) terminates in $\tilde{O}(n^{1/2+1/(2k)} + D)$ rounds.*

4.5 Approximation Analysis

So far, we have shown that our algorithm computes LE lists for G' and does so fast. However, we cannot apply Theorem 3.1 to show that these LE lists represent a virtual tree sampled from the (distribution of trees given by) the FRT embedding. Since the construction of G' depends on the choice of $\mathcal{S}$, and this choice depends on the random IDs, G' and the permutation on V induced by the IDs are not independent. We now adapt the analysis of [10] to our setting and show how to remedy the probabilistic dependencies created by our algorithm. In the following, denote by T the FRT tree specified by the LE lists on G'.

Lemma 4.9. *For each $v, u \in V$, we have that $\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u))] \in O(\log n) \cdot \mathrm{wd}_{G'}(v,u)$.*

Proof Sketch. For any $\mathcal{S} \subset V$, denote by $\mathcal{E}_{\mathcal{S}}$ the event that $\mathcal{S}$ is the set of the nodes with random IDs in the first $1/\sqrt{n}$ fraction of the ID range. It suffices to show that

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) \,|\, \mathcal{E}_{\mathcal{S}}] \in O(\log n) \cdot \mathrm{wd}_{G'}(v,u).$$

We condition on $\mathcal{E}_{\mathcal{S}}$ and an arbitrary outcome of the spanner construction (which uses independent randomness); this fixes G'. Note that the total orders the IDs induce on each of the sets $\mathcal{S}$ and $V \setminus \mathcal{S}$, respectively, are still uniformly random and *independent* of G'.

Fix $u, v \in V$. We say w *settles* u and v on level i, iff it is the node with the smallest ID so that

$$\min\{\mathrm{wd}_{G'}(w,v), \mathrm{wd}_{G'}(w,u)\} \leq \beta 2^{i-1}. \tag{1}$$

[2] Technically speaking, we use a slightly more general version of the lemma, in which a subset of the nodes may be anonymous; the reasoning remains identical. Here, all nodes but the copies of nodes in $\mathcal{S}$ are anonymous.

We say w *cuts* u and v on level i, iff it settles them on level i and also

$$\beta 2^{i-1} \leq \max\{\mathrm{wd}_{G'}(w,v), \mathrm{wd}_{G'}(w,u)\}. \tag{2}$$

It is not hard to show that if $(v_{i+1}, \ldots, v_L)$ is the least common ancestor of v and u, then $\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) < \beta 2^{i+1} < 2^{i+2}$ and either v_i or u_i cuts v and u. Hence, if we denote by $\mathcal{E}_{w,i}$ the event that $w \in V$ cuts u and v on level i, we have that

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) \,|\, \mathcal{E}_{\mathcal{S}}] < \sum_{w \in V} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} = \sum_{w \in \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} + \sum_{w \in V \setminus \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2}.$$

Both sums are handled analogously; let us consider only the first one here. Sort the nodes $w \in \mathcal{S}$ in ascending order w.r.t. $\min\{\mathrm{wd}_{G'}(w,v), \mathrm{wd}_{G'}(w,u)\}$ and let w_k be the k^{th} node in this order. We rewrite $P[\mathcal{E}_{w_k,i}]$ as

$$P[(1) \text{ and } (2) \text{ hold for } w_k \text{ and } i] \cdot P[w_k \text{ settles } u \text{ and } v \text{ on level } i \,|\, (1) \text{ and } (2) \text{ hold for } w \text{ and } i].$$

As the random order on $\mathcal{S}$ is uniform and independent of G', the second, conditional probability is $1/k$. Concerning the first probability, recall that (1) and (2) hold exactly if $\beta 2^{i-1} \in [\mathrm{wd}_{G'}(w,v), \mathrm{wd}_{G'}(w,u)]$. Here, W.l.o.g. we have assumed that $\mathrm{wd}_{G'}(w,v) < \mathrm{wd}_{G'}(w,u)$. Computation shows that

$$\sum_{i=1}^{L} P[(1) \text{ and } (2) \text{ hold for } w_k \text{ and } i] \cdot 2^{i+2} = \int_{\mathrm{wd}_{G'}(w,v)}^{\mathrm{wd}_{G'}(w,u)} 2^3 \, dx = 8(\mathrm{wd}_{G'}(w,u) - \mathrm{wd}_{G'}(w,v)) \leq 8\,\mathrm{wd}_{G'}(v,u),$$

where in the final step we applied the triangle inequality. We conclude that

$$\sum_{w \in W} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} \leq \sum_{k=1}^{|W|} \frac{8}{k} \cdot \mathrm{wd}_{G'}(v,u) < 8H_n \cdot \mathrm{wd}_{G'}(v,u) \in O(\log n) \cdot \mathrm{wd}_{G'}(v,u). \qquad \square$$

Full proofs of the lemma and the statements below are given in Appendix C. As distances in G' are w.h.p. at most by factor $O(\log n/\varepsilon)$ larger than in G, we conclude that the embedding given by the LE lists for G' is also good for G.

Corollary 4.10. *For each $v, u \in V$, we have that $\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u))] \in O(k \log n) \cdot \mathrm{wd}(v,u)$.*

We arrive at our main result, which was informally stated in Theorem 1.1.

Theorem 4.11. *For any $0 < \varepsilon \leq 1$, it is possible to sample from a distribution of probabilistic tree embeddings with expected stretch $O(\log n/\varepsilon)$ in $\tilde{O}(\min\{n^{0.5+\varepsilon}, \mathrm{SPD}\} + D)$ rounds w.h.p. in the* CONGEST *model. The computed embedding is given distributedly, in the form of corresponding LE lists. Moreover, if not all least-weight paths in G induced by the LE lists have $\tilde{O}(\sqrt{n})$ hops, the subtree of the virtual tree induced by the set $\mathcal{S}$ of nodes whose (uniformly random) ID is in the first $1/\sqrt{n}$ fraction of the ID range is known to all nodes, and for each edge $\{s,t\}$ in this subtree there is a unique s-t-path in G whose weight does not exceed the weight of the virtual edge and whose nodes know that they are on this path.*

We remark that instead of just constructing the tree embedding in form of the LE lists, this result also makes sure that the embedding can be used for approximation algorithms efficiently. For instance, it is essential that we can, e.g., select a root-leaf path in the virtual tree and map it back to a corresponding path in G in $\tilde{O}(\min\{n^{0.5+\varepsilon}, \mathrm{SPD}\} + D)$ rounds. Note that this operation is very basic, as it will be required whenever seeking to connect two leaves in different subtrees. Reconstructing such a path hop by hop using LE lists may take SPD time, which is too slow if $\mathrm{SPD} \gg \sqrt{n}$. Fortunately, if SPD is large, for each path all but a prefix of $\tilde{O}(\sqrt{n})$ hops corresponds to a route in the constructed skeleton spanner, and the additional information collected during the spanner construction stage is sufficient to quickly determine the remaining edges in G by announcing the (endpoints of) the subpath in the skeleton spanner to all nodes.

References

[1] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, Feb. 1995.

[2] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 184–, 1996.

[3] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 161–168, 1998.

[4] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures and Algorithms*, 30(4):532–563, 2007.

[5] J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

[6] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441–453, 1997.

[7] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 363–372, 2011.

[8] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 331–340, 2004.

[9] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 448–455, 2003.

[10] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[11] M. Ghaffari. Near-optimal distributed approximation of minimum-weight connected dominating set. In *the Pro. of the Int'l Colloquium on Automata, Languages and Programming (ICALP)*, 2014.

[12] M. Ghaffari and F. Kuhn. Distributed minimum cut approximation. In *Proc. of the Int'l Symp. on Dist. Comp. (DISC)*, pages 1–15, 2013.

[13] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 355–364, 2012.

[14] P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*, 37:46, 2004.

[15] M. Khan, F. Kuhn, D. Malkhi, G. Pandurangan, and K. Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, pages 263–272, 2008.

[16] C. Lenzen and B. Patt-Shamir. Fast routing table construction using small messages: Extended abstract. In *Proc. of the Symp. on Theory of Comp. (STOC)*, pages 381–390, 2013.

[17] C. Lenzen and B. Patt-Shamir. Improved distributed steiner forest construction. In *the Proc. of the Int'l Symp. on Princ. of Dist. Comp. (PODC)*, 2014.

[18] J. Matoušek. *Lectures on discrete geometry*, volume 212. Springer, 2002.

[19] D. Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proc. of the Symp. on Theory of Comp. (STOC)*, 2014, to appear.

[20] D. Peleg. *Distributed Computing: A Locality-sensitive Approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[21] D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed MST construction. In *Proc. of the Symp. on Found. of Comp. Sci. (FOCS)*, pages 253–, 1999.

A Lower Bound

Stating a lower bound for a distributed algorithm sampling from a distribution of tree embeddings is somewhat akward, as it is not at all clear how the distributed data structure representing the embedding is supposed to look like. However, some assumptions must be made, as technically any embedding is given by the distributed "data structure" consisting of the nodes' local views of the topology and their own randomness. Fortunately, the hardness barrier at running time roughly $\sqrt{n}$ (or, more accurately, $\min\{\sqrt{n}, \mathrm{SPD}\}$), is very strong, in the sense that it applies to a wide range of problems. The following corollary of [7] is based on reduction to the (approximate) lightest s-t path problem, which certainly should be trivial to solve on the virtual tree given any "reasonable" tree embedding. Translating the solution back to G should be trivial, too, as we are interested in the weight of the s-t-path in the virtual tree only. Consequently, while the lower bound we state has a long list of prerequisites, violating any of them results in embeddings that are most likely not useful.

Corollary A.1. *Suppose $\mathcal{A}$ is an algorithm in the* CONGEST *model that, for any graph G, constructs a virtual tree embedding $\mathfrak{M} : V \to V_T$ with the following properties:*

- *Each node in V_T has a unique identifier of $\operatorname{polylog} n$ bits.*
- *The depth of the virtual tree is polynomial in n.*
- *Each node $v \in V$ knows the path from its image $\mathfrak{M}(v)$ to the root of the tree (in terms of the identifiers and including edge weights).*
- *Distances in the virtual tree are never smaller than in the original graph.*
- *The expected stretch of each edge satisfies a non-trivial (i.e., polynomial in n) bound.*

Then, for arbitrary n and $\mathrm{SPD} \in \{D, \dots, n-1\}$, there is a graph G satisfying that $D \in O(\log n)$ and $\mathcal{A}$ runs for $\tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\} + D)$ rounds in expectation.

Proof. Observe that a lower bound of $\Omega(D)$ is trivial, i.e., it suffices to prove that $\tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\})$ rounds are necessary.

Suppose an algorithm $\mathcal{A}$ as specified runs on all graphs of n nodes with $D \leq \mathrm{SPD} \leq n-1$ and $D \in O(\log n)$ in $T = T(n, \mathrm{SPD}, D)$ expected rounds. We use $\mathcal{A}$ to approximate the weight of the lightest s-t-path on such graphs, for arbitrary $s, t \in V$, as follows. We run $\mathcal{A}$, which takes T (expected) rounds. Then, we find a shortest *unweighted* s-t-path in $O(D)$ rounds (constructing a BFS tree). Communicating over this path, s and t perform a binary search to determine the least common ancestor of their images under $\mathfrak{M}$ in the virtual tree; using binary search, this takes $D \operatorname{polylog} n = \operatorname{polylog} n$ rounds (due to the properties of $\mathcal{A}$ and because $D \in O(\log n)$). Afterwards, s sends the distance of $\mathfrak{M}(s)$ to its least common ancestor in the virtual tree to t, which takes $O(\log n)$ rounds in expectation, as $\mathrm{wd}(s,t) \leq \mathrm{WD} \in \operatorname{poly} n$ and we assume that the s-t-distance in the virtual tree is only by an expected factor of $\operatorname{poly} n$ larger. Now t can output the distance of s and t in the virtual tree, which is an expected polynomial approximation of $\mathrm{wd}(s,t)$.

The total running time of this approach is $T + \operatorname{polylog} n$ rounds. However, in [7] it is shown that for any n, there is a $D \in O(\log n)$ so that for any $D \leq \mathrm{SPD} \leq n-1$ there is an n-node graph of diameter D and shortest-path-diameter SPD in which the (expected) running time of any algorithm approximating s-t-distance within an expected polynomial factor must be $\tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\})$.[3] We conclude that

$$T(n, \mathrm{SPD}, D) \in \tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\}) - \operatorname{polylog} n = \tilde{\Omega}(\min\{\sqrt{n}, \mathrm{SPD}\}). \qquad \square$$

B Additional Details Concerning Least Element Lists and Spanners

Proof of Lemma 3.5. By initialization, each $v \in V$ satisfies that $(v, \mathrm{wd}(v,v)) \in L_v^{(0)}$. By Observations 3.2, 3.3, and 3.4, correct entries (i.e., those that show up in final lists) concerning v will spread along shortest

[3] To be precise, the construction in [7] satisfies that $\mathrm{SPD} \in \Theta(\sqrt{n})$. By adding some irrelevant nodes, this trivially generalizes to $D \leq \mathrm{SPD} \in O(\sqrt{n})$.

paths by one hop per iteration. In particular, the termination condition cannot be satisfied before the phase $i_0 \in \mathbb{N}_0$ satisfying that $(v, \mathrm{wd}(v,w)) \in L_v \Rightarrow (w, \mathrm{wd}(v,w)) \in L_v^{(i_0)}$. Observe that the scanning process and the definition of L_v imply that in fact $(v, \mathrm{wd}(v,w)) \in L_v \Leftrightarrow (w, \mathrm{wd}(v,w)) \in L_v^{(i_0)}$, and thus also that $(v, \mathrm{wd}(v,w)) \in L_v \Leftrightarrow (w, \mathrm{wd}(v,w)) \in L_v^{(i_0+1)}$. Hence the algorithm terminates after iteration $i_0 + 1$ with the correct output. Since information spreads along least-weight paths, it holds that $i_0 \leq \mathrm{SPD}$. □

With Lemma 3.5 in place, it remains to (i) show that each iteration can be performed quickly in the CONGEST model and (ii) termination can be checked efficiently. Regarding (ii), one can use the standard technique of continuously convergecasting indications of changes to any LE list over a BFS tree. If the root does not receive any such notification for the sum of the depth of the tree (which is at most D) and the time required for a loop iteration, it is safe to signal termination to all nodes over the BFS tree. Overall, the asymptotic cost of this mechanism is $O(D)$ rounds. We remark that Khan et al. [15] use a more elaborate, problem-specific mechanism, but it has no advantage over the generic approach.

Concerning (i), note that each entry in some $L_v^{(i)}$ can be encoded using $O(\log n)$ bits. Therefore, it suffices to show that $|L_v^{(i)}| \in O(\log n)$ w.h.p.; by the union bound, this entails that each iteration can be completed within $O(\log n)$ rounds w.h.p. To this end, we will have to analyze the behavior of the list variables during the course of the algorithm more closely. The distance variables for entries in $L_v^{(i)}$ represent the minimal weights of paths of at most i hops.

Definition B.1 (Bounded-Hop Distances)**.** *For $i \in \mathbb{N}_0$ and $v, w \in V$ with $\mathrm{hd}(v,w) \leq i$, define* $\mathrm{wd}^{(i)}(v,w) := \min\{W(p) \,|\, p \in P_{vw} \wedge \ell(p) \leq i\}$.

Note that $\mathrm{wd}^{(i)}(v,w) = \mathrm{wd}(v,w)$ iff there is a least-weight path from v to w of at most i hops, and $\mathrm{wd}^{(\mathrm{SPD})} = \mathrm{wd}$. We now can pinpoint the content of the variables $L_v^{(i)}$.

Lemma B.2. *For all $i \in \mathbb{N}_0$, $v, w \in V$ and each $d \in \mathbb{N}_0$, it holds that*

$$(w,d) \in L_v^{(i)} \Leftrightarrow \Big(\mathrm{hd}(v,w) \leq i \wedge d = \mathrm{wd}^{(i)}(v,w) \wedge (\nexists w' \in V : \mathrm{wd}^{(i)}(v,w') \leq \mathrm{wd}^{(i)}(v,w) \wedge w' < w)\Big)$$

(after the scans in iteration i are complete).

Proof. We show the claim by induction; it trivially holds for $i = 0$. Now suppose that it holds for $i \in \mathbb{N}_0$ and consider $i + 1$.

To show the implication "⇒", let us assume that $(w,d) \in L_v^{(i+1)}$. Clearly, $\mathrm{hd}(v,w) \leq i + 1$ and $d \geq \mathrm{wd}^{(i+1)}(v,w)$, since entries containing w spread by at most one hop each round and "accumulate" the weight of all traversed edges. Now consider a neighbor u of v on a least-weight path from v to w of at most $i + 1$ hops. By the induction hypothesis, either $(w, \mathrm{wd}^{(i)}(u,w)) \in L_u^{(i)}$, or there is an entry $(w', \mathrm{wd}^{(i)}(u,w')) \in L_u^{(i)}$ with $\mathrm{wd}^{(i)}(u,w') \leq \mathrm{wd}^{(i)}(u,w)$ and $w' < w$. In the latter case, we get the contradiction that v would delete (w,d) from $L_v^{(i+1)}$, since $\mathrm{wd}^{(i)}(u,w') + W(v,u) \leq \mathrm{wd}^{(i)}(u,w) + W(v,u) = \mathrm{wd}^{(i+1)}(v,w) \leq d$ and $w' < w$. Therefore, $(w, \mathrm{wd}^{(i)}(u,w), w) \in L_u^{(i+1)}$, and by the same argument, $d \leq \mathrm{wd}^{(i+1)}(v,w)$ (i.e., $d = \mathrm{wd}^{(i+1)}(v,w)$, since we showed that $d \geq \mathrm{wd}^{(i+1)}(v,w)$). To complete this direction, it thus remains to prove that $\nexists w' \in V : \mathrm{wd}^{(i+1)}(v,w') \leq \mathrm{wd}^{(i+1)}(v,w) \wedge w' < w$. Assuming for contradiction that this is the case, suppose that u is a neighbor of v on a least-weight path from v to w' of at most $i + 1$ hops. By the induction hypothesis, there is some w'' (maybe w' itself) satisfying that $(w', \mathrm{wd}^{(i)}(u,w')) \in L_u^{(i)}$, $\mathrm{wd}(u,w'') \leq \mathrm{wd}(u,w') = \mathrm{wd}(v,w') - W(v,u) \leq \mathrm{wd}(v,w) - W(v,u)$, and $w'' \leq w' < w$. Again, we arrive at the contradiction that (w,d) would be deleted from $L_v^{(i+1)}$.

To show the implication "⇐", assume that

$$\mathrm{hd}(v,w) \leq i + 1 \wedge (\nexists w' \in V : \mathrm{wd}^{(i+1)}(v,w') \leq \mathrm{wd}^{(i+1)}(v,w) \wedge w' < w).$$

Suppose u is a neighbor of v on a shortest path from v to w of at most $i+1$ hops. Then $\mathrm{hd}(u,w) \leq i$ and $\nexists w' \in V : \mathrm{wd}^{(i)}(u,w') \leq \mathrm{wd}^{(i)}(u,w) \wedge w' < w)$, since otherwise w' would violate the assumption that $\nexists w' \in V : \mathrm{wd}^{(i+1)}(v,w') \leq \mathrm{wd}^{(i+1)}(v,w) \wedge w' < w$. Thus, by the induction hypothesis, $(u, \mathrm{wd}^{(i)}(u,w)) \in L_u^{(i)}$, implying $(v, \mathrm{wd}^{(i+1)}(v,w)) \in L_v^{(i+1)}$ before the scan operation in iteration $i+1$. If this entry would get deleted, this would mean that there is an entry $(w', d) \in L_v^{(i+1)}$ with $d \leq \mathrm{wd}^{(i+1)}(u,w)$ and $w' < w$. However, by the previous arguments, this entry would satisfy that $d = \mathrm{wd}^{(i+1)}(v,w')$ and therefore w' would violate the assumption that $\nexists w' \in V : \mathrm{wd}^{(i+1)}(v,w') \leq \mathrm{wd}^{(i+1)}(v,w) \wedge w' < w$. This completes the induction. □

We remark that this lemma yields an alternative proof for Lemma 3.5, by observing that the algorithm can only terminate if all list variables have their final values and that $\mathrm{wd}^{(\mathrm{SPD})}(v,w) = \mathrm{wd}^{(\mathrm{SPD}+1)}(v,w) = \mathrm{wd}(v,w)$ for all $v, w \in V$.

We now can re-prove a key lemma from [15], stated here as Lemma 3.7. Our proof fixes a problem with the one from [15]: the authors argue that $L_v^{(i)}$ is v's LE list for the subgraph induced by least-weight paths of i hops that start at v, but this is not the case. More specifically, i-hop distances do not induce a metric if $i < \mathrm{SPD}$, as they do not satisfy the triangle inequality. Hence, Lemma B.2 shows that, in general, $L_v^{(i)}$ is not the LE list of v for *any* subgraph of G.

Proof of Lemma 3.7 (corrected). Fix $v \in V$ and $i \in \mathbb{N}_0$. Denote by $N \in \mathbb{N}$ the number of nodes $w \in V$ so that $\mathrm{hd}(v,w) \leq i$. We claim that the random variable $|L_v^{(i)}|$ is the sum of independent Bernoulli variables X_j, $j \in \{1,\dots,N\}$, where $P[X_j = 1] = 1/j$. From this the statement the lemma then follows by observing that

$$\mathbb{E}[|L_v^{(i)}|] = \mathbb{E}\left[\sum_{j=1}^{N} X_j\right] = \sum_{j=1}^{N} \frac{1}{j} \in O(\log N) \subseteq O(\log n)$$

and applying Chernoff's bound.

To see that the claim is true, order the N nodes in hop distance at most i from v in ascending order according to $\mathrm{wd}^{(i)}(v,\cdot)$. By Lemma B.2, for each such node w, there can be at most one entry in $L_v^{(i)}$, namely $(w, \mathrm{wd}^{(i)}(v,w))$, and this is the case if and only if w is smaller than all previous nodes (comparing random IDs). Since the IDs induce a uniformly random order, the probability for this is precisely $1/j$ for the j^{th} node, and this is independent from the order of the previous nodes. This proves that indeed $|L_v^{(i)}| = \sum_{j=1}^{N} X_j$, concluding the proof. □

Proof of Theorem 3.8. By Lemma 3.5, the algorithm is correct and requires at most $\mathrm{SPD}+1$ iterations. By Lemma 3.7 and the union bound, w.h.p. $|L_v^{(i)}| \in O(\log n)$ for all $v \in V$ and $i \in \{0,\dots,\mathrm{SPD}+1\}$.[4] Therefore, globally fixing a sufficiently large value $R \in \Theta(\log n)$ based on n, each iteration can be completed in R rounds w.h.p. Using the standard mechanism of detecting termination over a BFS tree (which can be constructed and used to determine n in $O(D)$ rounds), we obtain an implementation running in $(\mathrm{SPD}+1)R + O(D) + R = O(\mathrm{SPD} \log n)$ rounds w.h.p. □

Proof of Lemma 3.9. Fix $s, t \in \mathcal{S}$ and a least-weight s-t-path $p = (s = v_0, v_1, \dots, t = v_{\ell(p)})$. Determine node by node whether $v_1, v_2, \dots$ are in $\mathcal{S}$ until the first node in $\mathcal{S}$ is encountered. The probability for this taking more than h hops is $(1 - 1/\sqrt{n})^h \approx e^{-c \log n} < 1/n^c$. Since c is sufficiently large, w.h.p. there is a sampled node among the first h nodes. Repeating this argument inductively and applying the union bound, it follows that the sampled nodes are no more than h hops apart w.h.p. We conclude that $G_{\mathcal{S},h}$ contains an s-t-path of weight $\mathrm{wd}(s,t)$ w.h.p. Since trivially $\mathrm{wd}_{G_{\mathcal{S},h}}(s,t) \geq \mathrm{wd}(s,t)$, the claim of the lemma follows by applying the union bound to all pairs of nodes $s, t \in \mathcal{S}$. □

[4] Note that trivially $\mathrm{SPD} \leq n-1$, guaranteeing that we take the union over polynomially many events.

C Additional Details Concerning the Main Algorithm

Proof of Corollary 4.5. By Corollary 4.4, the initialization of the lists in Step 6 only adds correct entries w.r.t. G'. Reasoning analogously to Lemma 3.5, this implies that running the algorithm on G' would yield correct lists. Given that nodes in $\mathcal{S}$ get assigned their final lists already at initialization, there is no need for nodes in $\mathcal{S}$ to exchange any information, implying that running the algorithm on G leads to identical results. □

Proof of Lemma 4.6. Fix $v \in V \setminus \mathcal{S}$ and $w \in V$ and a shortest path from v to w in G'. Consider the prefix $p = (v, \dots, s)$ of the path so that $s \in \mathcal{S}$ and all other nodes are in $V \setminus \mathcal{S}$. We claim that, w.h.p., p is also a least-weight path in G. Assuming the contrary, there is a path q from v to s in G so that $W(q) < W(p)$. Observe that $W'(p) = \rho W(p)$ and, for each edge e on q that satisfies that $e \in E$, $W'(e) = \rho W(e)$. Moreover, by Theorem 3.10 and Lemma 3.9, for each edge $\{s,t\}$ on q with $s,t \in \mathcal{S}$, we have that $W'(s,t) = \mathrm{wd}_{\tilde{G}}(s,t) \leq \rho \,\mathrm{wd}_{G_{\mathcal{S},h}}(s,t) = \rho \,\mathrm{wd}(s,t)$ w.h.p. Note that q is also a path in G' and we just showed that $W'(q) \leq \rho W(q) < \rho W(p) = W'(p)$ w.h.p. Thus, unless some event with negligible probality occurs, we arrive at a contradiction. We conclude that p is, indeed, a least-weight path in G w.h.p.

Now fix any pair of nodes $v, w \in V$ and a least-weight path $p \in P_{vw}$. W.h.p., among the first $c\sqrt{n}\log n$ nodes on the path (where c is a sufficiently large constant), there is a node from $\mathcal{S}$: the probability that there is no such node is $(1 - 1/\sqrt{n})^{c\sqrt{n}\log n} \in e^{-\Theta(c)}$. Taking the union bound over all pairs of nodes, we conclude that for any pair of nodes $v, w \in V$, there is a least-weight path from v to w in G so that a node from $\mathcal{S}$ is encountered among the first $O(\sqrt{n}\log n)$ nodes on the path. The claim of the lemma now readily follows from Corollary 4.4, the initialization of the lists in Step 6, and Observation 3.4. □

Proof of Theorem 4.8. The first statement is given by Corollary 4.5. Clearly, the first step of the algorithm can be executed in $O(D)$ rounds. Step 2 is performed locally. Theorem 3.10 shows that Step 3 completes within $\tilde{O}(n^{1/2+1/(2k)} + D)$ rounds w.h.p. The theorem also guarantees that all nodes have the necessary information to perform the local computations required to execute Steps 5 and 6 (without further communication). Finally, Lemma 4.6 and Lemma 4.7 imply that the algorithm can be completed in $O(\sqrt{n}\log^2 n + D)$ additional rounds w.h.p. (cp. Theorem 3.8). Applying the union bound completes the proof. □

Proof of Lemma 4.9. Fix any $\mathcal{S} \subset V$ and denote by $\mathcal{E}_{\mathcal{S}}$ the event that $\mathcal{S}$ is the set of the nodes with random IDs in the first $1/\sqrt{n}$ fraction of the ID range. We will show that

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) \,|\, \mathcal{E}_{\mathcal{S}}] \in O(\log n) \cdot \mathrm{wd}_{G'}(v,u),$$

from which the claim of the lemma immediately follows because $\mathcal{S}$ is arbitrary. Throughout the proof, we will hence condition on $\mathcal{E}_{\mathcal{S}}$; for simplicity, we will largely omit this from the notation. The spanner construction from Theorem 3.10 uses independent randomness. Therefore, after determining the outcome of this construction, G' and thus also $\mathrm{wd}_{G'}$ are now fixed, where the random IDs of the nodes are constrained (exactly) by $\mathcal{E}_{\mathcal{S}}$. Note, however, that the total orders induced on the sets $\mathcal{S}$ and $V \setminus \mathcal{S}$, respectively, are still uniformly random and *independent* of G'. We will exploit this to, essentially, apply the FRT argument to both sets separately.

Fix $u, v \in V$. Suppose for a mapping $\mathfrak{M}$ drawn from the probability distribution of embeddings (conditioning on $\mathcal{E}_{\mathcal{S}}$ and fixed G'), we have that $\mathfrak{M}(v) = (v_0, v_1, \dots, v_L)$ and $\mathfrak{M}(u) = (u_0, u_1, \dots, u_L)$. Since $u_L = v_L$ and $v = v_0 \neq u_0 = v$, there is some i so that $v_i \neq u_i$ and $(v_{i+1}, \dots, v_L) = (u_{i+1}, \dots, u_L)$.

W.l.o.g., in the following we will assume that always $\mathrm{wd}_{G'}(w,v) < \mathrm{wd}_{G'}(w,u)$; the other case is symmetric. We say w *settles* u and v on level i, iff it is the node with the smallest ID so that

$$\mathrm{wd}_{G'}(w,v) \leq \beta 2^{i-1}. \tag{3}$$

Clearly, there is exactly one node settling u and v on each level, which is determined by G' and the random ordering (i.e., random IDs) of the nodes. Moreover, w *cuts* u and v on level i, iff it settles them on level i and also

$$\beta 2^{i-1} \leq \mathrm{wd}_{G'}(w,u). \tag{4}$$

Note that if $v_i \neq u_i$ and $(v_{i+1}, \dots, v_L) = (u_{i+1}, \dots, u_L)$, then $\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) = 2\sum_{j=1}^{i+1} \beta 2^{j-1} < \beta 2^{i+1} < 2^{i+2}$. Moreover, by definition $\mathrm{wd}(v_i, v) \leq \beta 2^{i-1}$, and assuming w.l.o.g. that $v_i < u_i$, we must also have that $\mathrm{wd}(v_i, u) > \beta 2^{i-1}$. Due to these constraints, v_i satisfies (3) and (4) for level i, and by construction $v_i < u_i$ is minimal among all nodes in distance at most $\beta 2^{i-1}$ of v or u; thus, v_i cuts u and v on level i. In summary, if we denote by $\mathcal{E}_{w,i}$ the event that $w \in V$ cuts u and v on level i, we have that

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) \mid \mathcal{E}_{\mathcal{S}}] < \sum_{w \in V} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} = \sum_{w \in \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} + \sum_{w \in V \setminus \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2}.$$

We will handle the two sums separately. Consider $W \subseteq V$, where either $W = \mathcal{S}$ or $W = V \setminus \mathcal{S}$. Sort the nodes $w \in W$ in ascending order w.r.t. $\min\{\mathrm{wd}_{G'}(w, v), \mathrm{wd}_{G'}(w, u)\}$ and let w_k be the k^{th} node in this order. We rewrite $P[\mathcal{E}_{w_k,i}]$ as

$$P[\text{(3) and (4) hold for } w_k \text{ and } i] \cdot P[w_k \text{ settles } u \text{ and } v \text{ on level } i \mid \text{(3) and (4) hold for } w \text{ and } i].$$

Observe that conditional on $\mathcal{E}_{\mathcal{S}}$, the random IDs induce a uniformly random permutation on W that is independent of G'. Therefore, the event that (3) and (4) hold for w and i is independent of the permutation of the nodes restricted to W, while the chosen order ensures that w_k can settle u and v only if $w_k < w_{k'}$ for all $k' \in \{1, \dots, k-1\}$. We conclude that

$$\sum_{i=1}^{L} P[\mathcal{E}_{w_k,i}] \cdot 2^{i+2} \leq \sum_{i=1}^{L} P[\text{(3) and (4) hold for } w_k \text{ and } i] \cdot \frac{1}{k} \cdot 2^{i+2}.$$

Recall that (3) and (4) hold exactly if $\beta 2^{i-1} \in [\mathrm{wd}_{G'}(w, v), \mathrm{wd}_{G'}(w, u)]$. We compute

$$\begin{aligned}
\sum_{i=1}^{L} P[\text{(3) and (4) hold for } w_k \text{ and } i] \cdot 2^{i+2} &= \sum_{i=1}^{L} \int_{1}^{2} P[\text{(3) and (4) hold for } w_k \text{ and } i \mid \beta = x] \cdot 2^{i+2}\, dx \\
&= \sum_{i=1}^{L} \int_{2^{i-1}}^{2^i} P[\text{(3) and (4) hold for } w_k \text{ and } i \mid \beta 2^{i-1} = x] \cdot 2^3\, dx \\
&= \int_{\mathrm{wd}_{G'}(w,v)}^{\mathrm{wd}_{G'}(w,u)} 2^3\, dx \\
&= 8 \cdot (\mathrm{wd}_{G'}(w, u) - \mathrm{wd}_{G'}(w, v)) \leq 8\, \mathrm{wd}_{G'}(v, u),
\end{aligned}$$

where in the final step we applied the triangle inequality. We conclude that

$$\sum_{w \in W} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} \leq \sum_{k=1}^{|W|} \frac{8}{k} \cdot \mathrm{wd}_{G'}(v, u) < 8H_n \cdot \mathrm{wd}_{G'}(v, u) \in O(\log n) \cdot \mathrm{wd}_{G'}(v, u).$$

Overall, we complete the proof as

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u)) \mid \mathcal{E}_{\mathcal{S}}] < \sum_{w \in \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} + \sum_{w \in V \setminus \mathcal{S}} \sum_{i=1}^{L} P[\mathcal{E}_{w,i}] \cdot 2^{i+2} \in O(\log n) \cdot \mathrm{wd}_{G'}(v, u). \quad \square$$

Proof of Corollary 4.10. By Theorem 4.9 and linearity of expectation,

$$\mathbb{E}[\mathrm{wd}_T(\mathfrak{M}(v), \mathfrak{M}(u))] \in O(\log n) \cdot \mathbb{E}[\mathrm{wd}_{G'}(v, u)] \subseteq O(k \log n) \cdot \mathrm{wd}(v, u),$$

where in the last step we exploit that, by Theorem 3.10, $\mathrm{wd}_{G'}(v, u) \leq (2k-1)\, \mathrm{wd}(v, u)$ w.h.p. and trivially $\mathrm{wd}_{G'}(v, u) \in O(\mathrm{WD})$ is polynomially bounded. $\square$

Proof of Theorem 4.11. We run the algorithm by Khan et al. [15] for up to $\sqrt{n}$ iterations on G, which by Theorem 4.7 and Theorem 3.8 takes $\tilde{O}(\min\{n^{0.5}, \mathrm{SPD}\} + D)$ rounds w.h.p. If it terminates, the statement follows, as the LE lists represent an FRT embedding of G and all induced paths have $O(\sqrt{n} \log \mathrm{wd}) \subset \tilde{O}(\sqrt{n})$ hops. Otherwise, it must hold that $\mathrm{SPD} \geq \sqrt{n}$ and we run our algorithm with parameter $k := \lceil 1/\varepsilon \rceil$. By Theorem 4.8, it computes LE lists for G' in $\tilde{O}(\min\{n^{0.5+\varepsilon}, \mathrm{SPD}\} + D)$ w.h.p. By Theorem 4.10, the stretch guarantee is satisfied. Finally, all nodes obtain knowledge of the spanner on $\mathcal{S}$, as stated in Theorem 3.10, showing that claim concerning long paths. □