Near-Optimal Spanners for General Graphs in (Nearly) Linear Time*

Hung Le† Shay Solomon‡

Abstract

Let $G = (V, E, w)$ be a weighted undirected graph on $|V| = n$ vertices and $|E| = m$ edges, let $k \geq 1$ be any integer, and let $\epsilon < 1$ be any parameter. We present the following results on fast constructions of spanners with near-optimal sparsity and lightness,[1] which culminate a long line of work in this area. (By *near-optimal* we mean optimal under Erdos' girth conjecture and disregarding the ϵ-dependencies.)

- There are (deterministic) algorithms for constructing $(2k-1)(1+\epsilon)$-spanners for G with a near-optimal sparsity of $O(n^{1/k} \cdot \log(1/\epsilon)/\epsilon))$. The first algorithm can be implemented in the pointer-machine model within time $O(m\alpha(m,n) \cdot \log(1/\epsilon)/\epsilon) + \mathsf{SORT}(m))$, where $\alpha(\cdot,\cdot)$ is the two-parameter inverse-Ackermann function and $\mathsf{SORT}(m)$ is the time needed to sort m integers. The second algorithm can be implemented in the Word RAM model within time $O(m \log(1/\epsilon)/\epsilon))$.
- There is a (deterministic) algorithm for constructing a $(2k-1)(1+\epsilon)$-spanner for G that achieves a near-optimal bound of $O(n^{1/k} \cdot \mathsf{poly}(1/\epsilon))$ on both sparsity and lightness. This algorithm can be implemented in the pointer-machine model within time $O(m\alpha(m,n) \cdot \mathsf{poly}(1/\epsilon) + \mathsf{SORT}(m))$ and in the Word RAM model within time $O(m\alpha(m,n) \cdot \mathsf{poly}(1/\epsilon))$.

The previous fastest constructions of $(2k-1)(1+\epsilon)$-spanners with near-optimal sparsity incur a runtime of is $O(\min\{m(n^{1+1/k}) + n \log n, k \cdot n^{2+1/k}\})$, even regardless of the lightness. Importantly, the *greedy spanner* for stretch $2k-1$ has sparsity $O(n^{1/k})$ — with no ϵ-dependence whatsoever, but its runtime is $O(m(n^{1+1/k} + n \log n))$. Moreover, the state-of-the-art lightness bound of any $(2k-1)$-spanner (including the greedy spanner) is poor, even regardless of the sparsity and runtime.

1 Introduction

Let $G = (V, E, w)$ be a weighted undirected graph on $|V| = n$ vertices and $|E| = m$ edges. We say that H is a t-spanner for G, for a parameter $t \geq 1$, if H preserves all pairwise distances of G to within a factor of t; the parameter t is called the *stretch* of the spanner. (A more detailed definition appears in Section 2.) The most basic requirement from a low-stretch spanner is to be *sparse*, i.e., of small size; the normalized notion of size, *sparsity*, is the ratio of the spanner size to the size $n-1$ of a spanning tree. A generalized requirement is to have a small *weight*; the *weight* of a spanner is the sum of its edge weights, and the normalized notion of weight, *lightness*, is the ratio of the spanner weight to the weight $w(\mathtt{MST}(G))$ of a minimum spanning tree $\mathtt{MST}(G)$ for G.

Sparse and light spanners have been studied extensively over the years, and have found a wide variety of applications across different areas, from distributed computing and motion planning to computational biology and machine learning. As prime examples, they have been used in achieving efficient broadcast protocols [ABP90, ABP92], for synchronizing networks and computing global functions [Awe85, PU89a, Pel00], in gathering and disseminating data [BKR+02, VWF+03, DK02], and to routing [WCT02, PU89b, ABLP89, TZ01b].

The holy grail is to achieve optimal tradeoffs between stretch and sparsity and between stretch and lightness, within a small running time. For unweighted graphs, this goal has been achieved already in the mid 90s, via a simple yet clever clustering approach due to Halperin and Zwick [HZ96]: A linear-time construction of $(2k-1)$-spanners with the optimal (under Erdos' girth conjecture [Erd64]) sparsity of $O(n^{1/k})$; we note that, for unweighted graphs, the sparsity and lightness parameters coincide.

The fundamental question underlying this work is whether one can achieve this goal in general weighted graphs. Chechik and Wulff-Nilsen [CW16] gave a poly-time construction of $(2k-1)(1+\epsilon)$-spanners with a *near-optimal* bound of $O(n^{1/k} \cdot \mathsf{poly}(1/\epsilon))$ on both sparsity and lightness; by *near-optimal* we mean optimal under

*The full version of the paper can be accessed at `https://arxiv.org/abs/2108.00102v1`

†University of Massachusetts Amherst.

‡Tel Aviv University.

[1]The sparsity (respectively, lightness) is a normalized notion of size (resp., weight), where we divide the size (resp., weight) by the size $n-1$ of a spanning tree (resp., the weight $w(\mathtt{MST})$ of a minimum spanning tree $\mathtt{MST}$).

Erdős' girth conjecture and disregarding the ϵ-dependencies. Although the runtime of the construction of [CW16] is polynomial, it is far from linear. Is it possible to achieve a fast — ideally linear time — spanner construction with the same guarantees? This question is open even disregarding the lightness: All known spanner constructions with near-optimal sparsity incur a rather high runtime.

Next, we survey the main results on spanners for general graphs, starting with sparse spanners and proceeding to light spanners. Subsequently, we present our contribution.

Sparse spanners. Graph spanners were introduced in the late 80s [PS89, PU89a]; initially, the focus was on the stretch-sparsity tradeoff. For unweighted graphs, the aforementioned construction of [HZ96] gives an optimal result. We shall henceforth consider general n-vertex m-edge weighted graphs. The "greedy spanner" is perhaps the most basic spanner construction, introduced in the seminal work of Althöfer et al. [ADD$^+$93]. For any integer parameter $k \geq 1$, it provides a $(2k-1)$-spanner with sparsity $O(n^{1/k})$. On the negative side, the running time of the greedy spanner is rather high, namely $O(m(n^{1+1/k} + n \log n))$.

The celebrated paper of Baswana and Sen [BS03] presents a randomized algorithm for constructing $(2k-1)$-spanners with sparsity $O(n^{1/k} \cdot k)$, within time $O(m \cdot k)$. Roditty, Thorup and Zwick [RTZ05a] derandomized the Baswana-Sen [BS03] algorithm, without any loss in parameters. This result is optimal except for an extra factor of k that appears in both the spanner size and the runtime bound.

Building on Miller et al. [MPVX15], Elkin and Neiman [EN18] gave a randomized algorithm for constructing $(2k-1)(1+\epsilon)$-spanners with sparsity $O(n^{1/k} \cdot \log k \cdot \log(1/\epsilon)/\epsilon)$, within time $O(m)$, for any $\epsilon < 1$; in fact, their runtime analysis overlooks the time consumed by a certain bucketing procedure, which, at least naively, requires $\Omega(\mathsf{SORT}(m))$ time, where $\mathsf{SORT}(m)$ is the time needed to sort m integers. Alstrup et al. [ADF$^+$19] achieved a deterministic algorithm with the same guarantees; we note that time $\Omega(\mathsf{SORT}(m))$ is also needed by the construction of [ADF$^+$19] for the same reason. These results demonstrate that by incurring an arbitrarily small multiplicative error of $1+\epsilon$ to the stretch bound, one can achieve, within linear time (modulo the overlooked time needed for integer sorting), a near-optimal sparsity bound, except for an extra $\log k$ factor. Additional results are summarized in Table 1.

As shown in Table 1, the previous state-of-the-art runtime for constructing $(2k-1)(1+\epsilon)$-spanners with near-optimal sparsity is $O(\min\{m(n^{1+1/k}) + n \log n, k \cdot n^{2+1/k}\})$, even regardless of the lightness.

Question 1. *Can one achieve a (nearly) linear time spanner construction with near-optimal sparsity?*

We answer Question 1 in the affirmative by presenting two algorithms for constructing spanners with near-optimal sparsity in near-linear time. Specifically, we prove the following result. (Refer to Table 1 for a detailed comparison between our and previous results.)

Theorem 1.1. *For any weighted undirected n-vertex m-edge graph G, any integer $k \geq 1$ and any $\epsilon < 1$, one can deterministically construct $(2k-1)(1+\epsilon)$-spanners with a near-optimal sparsity of $O(n^{1/k} \cdot \log(1/\epsilon)/\epsilon)$. This construction can be implemented:*

- *In the pointer-machine model within time $O(m\alpha(m,n) \cdot \log(1/\epsilon)/\epsilon + \mathsf{SORT}(m))$.*[2]
- *In the Word RAM model within time $O(m \log(1/\epsilon)/\epsilon)$.*[3]

We remark that $\alpha(m,n) = O(1)$ when $m = \Omega(n \log^* n)$. In fact, $\alpha(m,n) = O(1)$ even when $m = \Omega(n \log^{*(c)} n)$ for any constant c, where $\log^{*(\ell)}(.)$ denotes the iterated log-star function with ℓ stars; that is, $O(m\alpha(m,n))$ is bounded by $O(m + n \log^{*(c)} n)$ for any constant c. Thus the running time in the first item of Theorem 1.1 is linear in m in almost the entire regime of graph densities, i.e., except for very sparse graphs. Moreover, even

[2] In the *pointer machine model*, one can perform binary comparisons between data, arithmetic operations on data, dereferencing of pointers, and equality tests on pointers. The model does not permit pointer arithmetic or tests other than equality on pointers and thus is less powerful than the RAM model [Tar79].

[3] The Word RAM model is similar to the classic unit-cost RAM model, except that (1) For a word length $w \geq 1$ the contents of all memory cells are integers up to 2^w. (2) Some additional instructions are available; in particular, the available unit-time operations are those from the *restricted instruction set*: addition and subtraction, (noncyclic) bit shifts by an arbitrary number of positions, and bitwise boolean operations, but not multiplication. (3) It is also assumed that $w \geq \log n$. We note that if the running time of the algorithm depends on the input size but not on the word size, then the model is further called Transdichotomous model; the running time of our algorithms do not depend on the word size.

Stretch	Sparsity	Lightness	Construction Time	Ref
$(2k-1)$	$O\left(n^{1/k}\right)$	$O\left(n/k\right)$	$O\left(mn^{1+1/k}+n\log n\right)$	[ADD$^+$93]
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(mn^{1+1/k}+n\log n\right)$	[CDNS92]
$(2k-1)$	$O(n^{1/k})$	—	$O\left(kn^{2+1/k}\right)$	[RZ11]
$(2k-1)$	$O\left(k\cdot n^{1/k}\right)$	—	$O\left(kmn^{1/k}\right)$	[TZ01a]^R
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(kn^{2+1/k}\right)$	[ES16]
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(n^{1/k}\cdot k/\log k\right)$	$O\left(mn^{1+1/k}+n\log n\right)$	[ENS15]
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(n^{1/k}\right)$	$n^{\Theta(1)}$	[CW18]
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(n^{1/k}\right)$	$O\left(mn^{1+1/k}+n\log n\right)$	[FS20]
$(2k-1)(1+\epsilon)$	$O\left(n^{1/k}\right)$	$O\left(n^{1/k}\right)$	$O(n^{2+1/k+\epsilon'})$	[ADF$^+$19]
$(2k-1)$	$O\left(k\cdot n^{1/k}\right)$	—	$O\left(km\right)$	[BS07]^R [RTZ05b]
$(2k-1)(1+\epsilon)$	$O\left(k\cdot n^{1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(\mathsf{SORT}(m)+km+n\log n\right)$	[ES16]
$(2k-1)(1+\epsilon)$	$O(\log k\cdot n^{1/k})$	$O\left(k\cdot n^{1+1/k}\right)$	$O(\mathsf{SORT}(m)+n\cdot\log n)$	[EN18]^R
$(2k-1)(1+\epsilon)$	$O\left(\log k\cdot n^{1/k}\right)$	—	$O(\mathsf{SORT}(m))$	[EN18, ADF$^+$19]
$(2k-1)(1+\epsilon)$	$O\left(\log k\cdot n^{1/k}\right)$	$O\left(\log k\cdot n^{1/k}\right)$	$O(\mathsf{SORT}(m)+n\cdot\log n)$	[ADF$^+$19]
$(2k-1)(1+\epsilon)$	$O(n^{1/k})$	—	$O(m\alpha(m,n)+\mathsf{SORT}(m))$	Theorem 1.1 ^{P}
$(2k-1)(1+\epsilon)$	$O(n^{1/k})$	—	$O(m)$	Theorem 1.1 ^{W}
$(2k-1)(1+\epsilon)$	$O(n^{1/k})$	$O(n^{1/k})$	$O(m\alpha(m,n)+\mathsf{SORT}(m))$	Theorem 1.2 ^{P}
$(2k-1)(1+\epsilon)$	$O(n^{1/k})$	$O(n^{1/k})$	$O(m\alpha(m,n))$	Theorem 1.2 ^{W}

Table 1: Table summarizing known and new spanner constructions for general weighted graphs, for stretch values of $2k-1$ and $(2k-1)(1+\epsilon)$. In the top and middle parts of the table we list rather slow and fast known spanner constructions, respectively. Our new results appear at the bottom. Results marked with ^{R} correspond to randomized constructions. We use the superscript marks ^{P} and ^{W} to distinguish between the new algorithms that apply to the pointer-machine versus the Word RAM models, respectively.

when $\alpha(m,n)$ is super-linear, it can still be viewed as constant for most practical purposes. However, there is a significant qualitative difference between truly linear-time and nearly linear-time algorithms, and shaving this factor for the entire regime of graph densities, as provided by the second item of Theorem 1.1, is of fundamental theoretical importance.

The previous linear-time algorithms for constructing sparse spanners in general weighted graphs [MPVX15, EN18, ADF$^+$19] achieve a sub-optimal sparsity bound of $O(n^{1/k}\cdot\log k\cdot\log(1/\epsilon)/\epsilon)$, and, as mentioned, their runtime is actually $O(\mathsf{SORT}(m))$. Moreover, these constructions, as well as all other spanner constructions with runtime $o(km)$ (including ours), use a hierarchical clustering approach that involves constructing a so-called *cluster graph* in each level of the hierarchy. Importantly, the cluster graph is a simple graph (without self loops and parallel edges), and all the previous works either overlooked the time needed to guarantee that the cluster graph is simple or they included an extra factor of $\alpha(m,n)$ in the runtime bound — due to the usage of the classic Union-Find data structure [Tar75]. We demonstrate that this factor can be shaved via a novel clustering approach, which we name *MST-clustering*; refer to Subsection 1.1 for a discussion on the technical details.

Light spanners. Like sparsity, the lightness of spanners has been extremely well-studied. Althöfer et al. [ADD$^+$93] showed that the lightness of the greedy $(2k-1)$-spanner is $O(n/k)$. Despite extensive research, the state-of-the-art lightness bound of any known $(2k-1)$-spanner construction (including the greedy spanner) remains poor, even regardless of the sparsity and runtime. It is thus only natural to explore the lightness bound for a slightly increased stretch of $(2k-1)(1+\epsilon)$, where $\epsilon<1$ is an arbitrarily small parameter of our choice. Chandra et al. [CDNS92] showed that the greedy $(2k-1)(1+\epsilon)$-spanner has lightness $O(k\cdot n^{1/k}\cdot(1/\epsilon)^2)$. There was a sequence of works from recent years on light spanners [ES16, ENS14, CW16, FS20, EN18, ADF$^+$19, LS21].

In particular, a construction of $(2k-1)(1+\epsilon)$-spanners with a near-optimal lightness of $O(n^{1/k} \cdot \mathsf{poly}(1/\epsilon))$ within a runtime of $O(m\alpha(m,n)$ was presented recently [LS21], where $\alpha(\cdot,\cdot)$ is the inverse-Ackermann function; on the negative side, the sparsity of the construction of [LS21] is unbounded. As mentioned, the construction of [CW16] achieves a near-optimal bound of $O(n^{1/k} \cdot \mathsf{poly}(1/\epsilon))$ on both sparsity and lightness, but its runtime is far from linear. The result of Filtser and Solomon [FS20] implies that the greedy spanner achieves the same bounds as the construction of [CW16], but the runtime $O(m(n^{1+1/k} + n\log n))$ of the greedy spanner is also rather high. The construction of [ADF+19] is nearly linear, but the sparsity bound and lightness bound are suboptimal, as indicated in Table 1.

QUESTION 2. *Can one achieve a (nearly) linear time spanner construction with a near-optimal bound on both the sparsity and lightness?*

We answer Question 2 in the affirmative by presenting an algorithm for constructing $(2k-1)(1+\epsilon)$-spanners with near-optimal sparsity and lightness in near-linear time, which culminates a long line of work in this area. Specifically, we prove the following result.

THEOREM 1.2. *For any weighted undirected n-vertex m-edge graph G, any integer $k \geq 1$ and any $\epsilon < 1$, one can deterministically construct $(2k-1)(1+\epsilon)$-spanners with a near-optimal bound of $O(n^{1/k} \cdot \mathsf{poly}(1/\epsilon))$ on both sparsity and lightness. This construction can be implemented:*

- *In the pointer-machine model within time $O(m\alpha(m,n) \cdot \mathsf{poly}(1/\epsilon) + \mathsf{SORT}(m))$.*
- *In the Word RAM model within time $O(m\alpha(m,n) \cdot \mathsf{poly}(1/\epsilon))$.*

We obtain the result of Theorem 1.2 by strengthening the framework of [LS21] for fast constructions of light spanners to achieve a near-optimal bound on the sparsity as well. To this end, we plug the ideas used in the proof of Theorem 1.1, in conjunction with numerous new insights, on top of the framework of [LS21] in a highly nontrivial way. Our MST-clustering approach plays a key role not just in the proof of Theorem 1.1, but also in the proof of Theorem 1.2; refer to Subsection 1.1 for more details.

1.1 Technical Highlights Our spanner construction is inspired by the constructions of [MPVX15], [EN18] and [ADF+19], which we briefly review next. All these constructions achieve a runtime of $O(m)$, modulo the time needed for sorting the edge weights; we shall elaborate on this point later. The construction of [MPVX15] achieves stretch $O(k)$ with sparsity $O(n^{1/k}\log(k)))$, while the two other constructions achieve the same sparsity but with a stretch of $(2k-1)(1+\epsilon)$. (For clarity, we shall ignore the dependency on ϵ in the sparsity bounds.)

The construction of [MPVX15][4] starts by dividing the edge set into $\mu_k \stackrel{\text{def.}}{=} O(\log k)$ sets $\{E^1, E^2, \ldots, E^{\mu_k}\}$, such that for each set E^σ, $\sigma \in [1,\mu_k]$, any two edge weights are either within a factor of 2 from each other or they are separated by at least a factor of k^c for some constant c. The algorithm then focuses on constructing a spanner H^σ for each edge set E^σ separately; the sparsity of H^σ is $O(n^{1/k})$, which ultimately leads to a sparsity bound of $O(\mu_k \cdot n^{1/k}) = O(\log(k) \cdot n^{1/k})$ of the final spanner H. In the construction of H^σ, the edge set E^σ is further divided into smaller subsets $\{E_1^\sigma, E_2^\sigma, \ldots\}$, where edges in the same set E_i^σ have the same weights up to a factor of 2, and the weights of edges in E_i^σ are at least k^c times greater than the weights of edges in E_{i-1}^σ, for each i. The construction of [MPVX15] uses a *hierarchy of clusters* and an *unweighted cluster graph* R_i for each level i of the hierarchy. The vertex set of R_i corresponds to a *subset* of level-i clusters that are incident to at least one edge in E_i^σ, and the edge set of R_i corresponds to a subset of edges in E_i^σ interconnecting level-i clusters. A preprocessing step is applied to the construction of R_i to remove parallel edges, which are edges in E_i^σ connecting the same two level-i clusters, and self-loops, which are edges in E_i^σ whose both endpoints are in the same level-i cluster. The construction of [MPVX15] then builds an $O(k)$-spanner for the (unweighted) graph R_i to obtain a subset of edges S_i of E_i^σ to add to H^σ. Next, vertices in R_i are grouped into a set $\mathcal{U}$ of subgraphs of (unweighted) diameter $\Theta(k)$; each subgraph in $\mathcal{U}$ is then transformed into a level-$(i+1)$ cluster. The construction then continues to level $i+1$, then to level $i+2$, etc., until all the edges in the graph have been considered. The construction of the (unweighted) $O(k)$-spanner of R_i and the set of subgraphs $\mathcal{U}$ is randomized and based on sampling from an exponential distribution.

[4]The algorithm used in [MPVX15] is parallel, and our interpretation of it is in the standard sequential model.

The construction of [EN18] builds on that of [MPVX15]. First, it partitions the edge set into $\mu_{k,\epsilon} = O(\log(k)/\epsilon)$ sets of edges instead of $O(\log k)$ sets as in [MPVX15]; the idea is that for each set E^σ, $\sigma \in [1, \mu_{k,\epsilon}]$, any two edge weights are either within a factor of $1+\epsilon$ from each other, or are separated by at least a factor of k^c for some constant c. Next, the construction of [EN18] uses the same idea of [MPVX15] to construct the spanner of R_i and the set of subgraphs $\mathcal{U}$. However, the stretch of the spanner is improved to $(2k-1)$, which readily implies a stretch of $(4k-2)(1+\epsilon)$ for the final spanner. We note that the stretch is $(4k-2)(1+\epsilon)$ instead of $(2k-1)(1+\epsilon)$, due to a subtlety involving randomness in [MPVX15]. With a more sophisticated analysis, [EN18] resolves this subtlety and reduces the stretch to $(2k-1)(1+\epsilon)$. The sparsity of the final spanner is $O(\mu_{k,\epsilon} \cdot n^k) = O(\log(k) \cdot n^{1/k})$, ignoring the dependence on ϵ.

Unlike the constructions of [MPVX15, EN18], the construction of [ADF+19] is *deterministic*. A central idea in the construction of [ADF+19], inspired by an earlier work [ES16], is to use a modified version of the Halperin-Zwick algorithm [HZ96] in the construction of the spanner of R_i. The spanner of R_i has stretch $(2k-1)$, which implies the final stretch of $(2k-1)(1+\epsilon)$. The sparsity of the spanner remains $O(\mu_{k,\epsilon} \cdot n^k) = O(\log(k) \cdot n^{1/k})$, as in [MPVX15, EN18].

We note the following points regarding the aforementioned constructions.

1. First, the sparsity incurs an extra factor of $O(\log k)$, i.e., it is $O(\log(k) \cdot n^{1/k})$ rather than $O(n^{1/k})$. This is inevitable, since subgraphs in $\mathcal{U}$ of R_i have a diameter of $\Theta(k)$, hence the weights of edges in E^σ_{i+1} and E^σ_i must be at least a factor of k^c apart from each other, which ultimately leads to a factor $O(\log k)$ in the number of sets that the edge set E is partitioned to.

2. Second, each set E^σ is partitioned into $O(\log U)$ sets $\{E^\sigma_1, E^\sigma_2, \ldots\}$, where U is the maximum edge weight. Thus, at least naively, the partition of E^σ can be constructed in time $O(m + \log U)$ rather than $O(m)$, where U could be unbounded. One way to avoid the dependency on U is to sort all edge weights of E^σ, which requires time $\mathsf{SORT}(m)$. We note that the computation of the partition of E^σ into subsets has been overlooked in the aforementioned constructions [MPVX15, EN18, ADF+19]. In the Word RAM model, we use the simple observation that $O(\log U)$ is roughly the word size to guarantee that such a partition can be computed within $O(m)$ time.

3. Third, the aforementioned constructions involve constructing a cluster graph R_i associated with each level i of the hierarchy. While the details of maintaining R_i are not precisely described in these constructions, we observe that R_i can be efficiently maintained using the Union-Find data structure. However, the total runtime would be $O(m\alpha(m,n))$ rather than $O(m)$. We next show that the non-optimal sparsity bound of $O(n^{1/k} \log k)$ achieved by the previous works can be used to remove the factor $\alpha(m,n)$. Observe that $m\alpha(m,n) = O(m)$ when $m = \Omega(n \log\log(n))$. If $m = O(n^{1+1/k} \log k)$, we can simply return the whole graph as the output spanner. Otherwise, $m = \Omega(n^{1+1/k} \log k) = \Omega(n \log\log n)$ for every $k \geq 2$, in which case the total time to construct a spanner of size $O(n^{1+1/k} \log k)$ is $O(m\alpha(m,n)) = O(m)$. However, the same argument fails when aiming for the near-optimal sparsity bound of $O(n^{1/k})$ that we achieve (e.g., $O(n^{1+1/k}) = O(n)$ when $k = \Omega(\log n)$). To construct a spanner with a sparsity of $O(n^{1/k})$ in $O(m)$ time, one must overcome the "Union-Find barrier". We note that even in the cell-probe model, which is stronger than the Word RAM model, one cannot avoid the factor $\alpha(m,n)$ in the Union-Find data structure [FS89].

Our first construction is in the pointer-machine model; there we overcome the "(unweighted) diameter barrier" of $\Theta(k)$ of subgraphs in $\mathcal{U}$ constructed from R_i: Subgraphs in our construction have (unweighted) diameters of $O(1)$. As a consequence, we demonstrate that it suffices to partition E into $\mu_\epsilon \stackrel{\text{def.}}{=} O(\frac{1}{\epsilon}\log(1/\epsilon))$ sets instead of $O(\log k/\epsilon)$ sets, which ultimately leads to the optimal sparsity of $O(n^{1/k})$, ignoring the dependence on ϵ. The key idea behind our construction is rather simple — we prove that it suffices to construct level-$(i+1)$ clusters from level-i clusters such that the total number of clusters is reduced by $\Omega(|V(R_i)|)$. We then use the Halperin-Zwick algorithm [HZ96] to construct a $(2k-1)$-spanner for R_i. Next, we construct the set of subgraphs $\mathcal{U}$ greedily, with each having diameter $O(1)$. By using the Union-Find data structure in the construction of R_i, the total running time of our algorithm is $O(m\alpha(m,n))$, plus an additive term of $\mathsf{SORT}(m)$ needed for computing the partition of E^σ as discussed above. Note that we cannot use the trick that we provided earlier to remove the $\alpha(m,n)$ factor since our spanner construction does not have any slack on the sparsity. Our construction is deterministic, it improves the aforementioned constructions [MPVX15, EN18, ADF+19] — yet is arguably simpler.

Our linear-time spanner construction in the Word RAM model is based on a novel clustering approach, which we name *MST-clustering*. Specifically, we guarantee that the subgraphs induced by clusters are subtrees of a minimum spanning tree (MST) of the graph, denoted by $\texttt{MST}$, and hence, every UNION operation is performed along the edges of $\texttt{MST}$. That is, each UNION operation is of the form UNION(u, v), where (u, v) is an edge in $\texttt{MST}$. As a result, we are able to determine all the UNION operations even before the cluster construction takes place. This allows us to use a refined UNION-FIND data structure, by Gabow-Tarjan [GT85], which has $O(1)$ amortized cost per UNION/FIND operation. To the best of our knowledge, this is the first time that the MST serves as the union tree in the Gabow-Tarjan UNION-FIND data structure, other than in applications that *directly concern MST*.

The idea of using the MST in the context of clustering in spanner constructions is quite surprising. In many of the known spanner constructions, clusters in the cluster hierarchy need to satisfy a diameter constraint. That is, clusters at level-i should have a diameter of at most $f(L_i)$, for some function f, often a linear function, where L_i is an upper bound on the edge weights in E_i^σ. In particular, the approaches of [MPVX15, EN18, ADF$^+$19, LS21] utilize the fact that some edges (not in $\texttt{MST}$) have been added during the construction of clusters at lower levels, and use these edges to construct clusters that satisfy the diameter constraint. By restricting ourselves to only use $\texttt{MST}$ for clustering, it seems much more challenging (and perhaps impossible at first) to guarantee the diameter constraint for level-i clusters. Our key insight is that it is still possible to do so, and to this end we rely on the cycle property of MST, both for arguing that clusters have small diameters and for constructing clusters efficiently.

Finally, we show how to construct a spanner with near-optimal sparsity *and lightness*. Our construction builds on the fast construction of spanners with near-optimal lightness in [LS21]. The construction of [LS21] has a preprocessing step and a main construction step. In the preprocessing step, every edge of weight at most $\frac{w(\texttt{MST})}{m\epsilon}$ is added to the spanner. Clearly the number of edges added in this step could be as large as $\Omega(n^2)$ (for dense graphs). Our first observation is that, except for $\texttt{MST}$ edges, edges added in the preprocessing step are not involved in the main construction step, and hence we can apply our sparse spanner construction from Theorem 1.1 to reduce the number of edges added in the preprocessing step to $O(n^{1+1/k})$. The main construction step is based on a cluster hierarchy. However, clusters in [LS21] are "equipped" with a potential function, and the challenge of the cluster construction is to guarantee a sufficient reduction in the potential values between two consecutive levels of the hierarchy. A cluster graph is also used to select a subset of edges in E_i^σ to add to the spanner. Again, the number of edges added in this step could be as large as $\Omega(n^2)$. In order to obtain a spanner with near-optimal guarantees on both sparsity and lightness, we employ the insight that we developed in this paper for the construction of sparse spanners, by constructing clusters in such a way that, between two consecutive levels, there is a sufficient reduction not just in the potential values, but also in the *number of clusters*. This, in turn, makes the task of constructing clusters much more challenging; indeed, a-priori, it is unclear that it is possible to achieve both objectives via a single (fast) spanner construction.

The spanner construction of [LS21] constructs level-$(i+1)$ clusters in 5 steps; each level-$(i+1)$ cluster corresponds to a subgraph of a cluster graph R_i. We note that the cluster graph R_i in this construction is different from the cluster graph used in the sparse spanner constructions in that its MST, denoted by $\widetilde{\texttt{MST}}_i$, is derived from the MST of G. We observe that among the 5 steps used in [LS21], there are two steps where the reduction in the number of clusters is not guaranteed. Furthermore, the clusters formed in these two steps are subgraphs of $\widetilde{\texttt{MST}}_i$. Thus, our idea is to apply the insights we developed in the sparse spanner construction in the Word RAM model to this setting. However, there are two subtleties in the construction of [LS21] that we need to address. First, the cluster graph R_i has weights on both edges and vertices. As a result, $\widetilde{\texttt{MST}}_i$ also has weights on both edges on vertices. Second, clusters in the construction of [LS21] contain *virtual vertices*; these vertices are not in the input graph and are introduced to support the design of the potential function for clusters. We show an analogous version of the cycle property for $\widetilde{\texttt{MST}}_i$. We use this property, in addition to several other technical ideas, to transfer insights that we developed in the construction of sparse spanners in the Word RAM model to the cluster construction in this setting. As a result, our spanner construction that achieves near-optimal bounds on both sparsity and lightness is much more involved than our two aforementioned constructions (which prove Theorem 1.1) with near-optimal sparsity but possibly huge lightness.

2 Preliminaries

We denote by $G = (V, E, w)$ a graph G with vertex set V, edge set E, and weight function $w : E(G) \to \mathbb{R}^+$ on its edges. We denote by $\texttt{MST}(G)$ the minimum spanning tree of G; there could be MSTs for G, but we may assume

w.l.o.g. that there is only one (e.g., by using lexicographic rules to break ties for edges of the same weight). When the graph is clear from the context, we abbreviate $\texttt{MST}(G)$ as $\texttt{MST}$. We denote by $w(G) = \sum_{e \in E} w(e)$ the *weight* of G, i.e., the sum of all edge weights in G.

We use $d_G(u, v)$ to denote the distance between two vertices u and v in G. The diameter of G is the maximum pairwise distance in G, and is denoted by $\mathsf{Dm}(G)$.

For a subset of vertices $X \subseteq V$, we denote by $G[X]$ the subgraph of G induced by X. We also define a subgraph of G induced by an *edge set* F by $G[F] = (V, F)$

Let H be a spanning subgraph of G (with edge weights inherited from G). The *stretch* of H is defined as $\max_{u \neq v \in V} \frac{d_H(u,v)}{d_G(u,v)}$; H is called a *t-spanner* of G if its stretch is at most t. The next well-known observation, which states that the stretch of H is realized by an edge of G, follows from the triangle inequality.

OBSERVATION 1. $\max_{u \neq v \in V(G)} \frac{d_H(u,v)}{d_G(u,v)} = \max_{(u,v) \in E(G)} \frac{d_H(u,v)}{d_G(u,v)}$.

We say that H is a spanner for a *subset of edges* $X \subseteq E$ if $\max_{(u,v) \in X} \frac{d_H(u,v)}{d_G(u,v)} \leq t$.

Our constructions use the aforementioned linear-time construction of $(2k-1)$-spanners for unweighted graphs by Halperin-Zwick [HZ96], which we record in the following theorem for further use.

THEOREM 2.1. ([HZ96]) *For any unweighted n-vertex m-edge graph G and any integer $k \geq 1$, a $(2k-1)$-spanner of G with $O(n^{1+\frac{1}{k}})$ edges can be constructed deterministically in $O(m+n)$ time.*

In proving Theorem 1.1 and Theorem 1.2, we assume that ϵ is sufficiently smaller than 1. That is, $\epsilon \leq 1/c$ for some constant $c \geq 1$. We can remove this assumption by setting $\epsilon = \epsilon'/c$ for any $0 < \epsilon' < 1$. This will increase the dependency on ϵ of the sparsity and running time only by a constant factor.

3 An $O(m\alpha(m,n) + \mathsf{SORT}(m))$-time Algorithm

In this section we prove the first item of Theorem 1.1. By scaling, we assume that the minimum edge weight is 1. We partition the edge set E into $\mu_\epsilon = \log_{1+\epsilon}\left(\frac{1}{\epsilon}\right) = \Theta(\frac{\log(1/\epsilon)}{\epsilon})$ sets $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$ such that each E^σ can be written as $E^\sigma = \cup_{i \in \mathbb{N}} E_i^\sigma$ with:

$$E_i^\sigma = \{e \in E : \frac{L_i}{(1+\epsilon)} \leq w(e) \leq L_i, i \in \mathbb{N}\}, \text{ where } L_i = L_0/\epsilon^i, L_0 = (1+\epsilon)^\sigma. \tag{3.1}$$

Thus, for any edge set E^σ, any two edge weights are either roughly the same (up to a factor of $1+\epsilon$) or far apart (separated by at least a factor of $1/\epsilon$). For technical convenience, we shall define $L_{-1} = 0$.

For notional convenience, we define E_i^σ for every $i \in \mathbb{N}$. However, we are only interested in i such that $E_i^\sigma = \emptyset$, and there are only a finite number of such indices.

We note that the time needed to compute the partition of E into the sets $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$ is upper bounded by $O(m + \mathsf{SORT}(m')) = O(\mathsf{SORT}(m))$, where m' is the number of non-empty sets. Indeed, this computation can be carried out naively in linear time, except for the time needed to sort the *indices* of the non-empty sets in $\{E_i^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon, i \in \mathbb{N}}$. In the runtime analysis that follows we shall disregard this initial time investment, under the understanding that we include it in the final runtime bound.

We now construct a $(2k-1)(1+O(\epsilon))$-spanner H^σ for each set E^σ with sparsity $O(n^{1/k})$ in $O(m \cdot \alpha(m,n))$ time. A $(2k-1)(1+O(\epsilon))$-spanner H for G with sparsity $O(n^{1/k} \cdot \frac{\log(1/\epsilon)}{\epsilon})$ is then obtained as the union of all H^σ's: $H = \cup_{1 \leq \sigma \leq \mu_\epsilon} H^\sigma$, within time $O(m \cdot \alpha(m,n) \cdot \frac{\log(1/\epsilon)}{\epsilon})$.

We focus on the construction of H^σ, for a fixed $\sigma \in [1, \mu_\epsilon]$. Initially $H_0^\sigma = (V, \emptyset)$. The construction is carried out in what we call *levels*: at level i, we shall construct a subgraph H_i^σ such that $H_{\leq i}^\sigma$ is a $(2k-1)(1+O(\epsilon))$-spanner for the edge set $E_{\leq i}^\sigma$. Here $H_{\leq i}^\sigma = \cup_{0 \leq j \leq i} H_j^\sigma$ and $E_{\leq i}^\sigma = E_{0 \leq j \leq i}^\sigma$. By induction, $H^\sigma \stackrel{\text{def.}}{=} \cup_{i \geq 0} H_i^\sigma$ would provide a $(2k-1)(1+O(\epsilon))$-spanner for the edge set E^σ. Consequently, $H = \cup_{1 \leq \sigma \leq \mu_\epsilon} H^\sigma$ will provide a $(2k-1)(1+O(\epsilon))$-spanner for $E = \bigcup_{1 \leq \sigma \leq \mu_\epsilon} E^\sigma$, and, by Observation 1, also for G . All graphs H_i^σ share the same vertex set V and hence are distinguished by the edge set.

A *cluster* is a set of vertices. Our construction uses a *hierarchical clustering*, where for each $i \geq 0$, the construction at level i is associated with a set of clusters $\mathcal{C}_i$ such that:

- **(P1)** Each cluster $C \in \mathcal{C}_i$ is a subset of V. Furthermore, clusters in $\mathcal{C}_i$ induce a partition of V.

- **(P2)** Each cluster $C \in \mathcal{C}_i$ induces a subgraph $H^{\sigma}_{\leq i}[C]$ of diameter $\leq gL_{i-1}$ for some constant g.

$\mathcal{C}_0$ is the set of n singletons of V and hence trivially satisfies both Properties (P1) and (P2) (recall that $L_{-1} = 0$). The cluster sets $\{\mathcal{C}_0, \mathcal{C}_1, \ldots\}$ provide a *hierarchy of clusters* $\mathcal{H}$. In particular, for any $i \geq 1$, $\mathcal{C}_{i-1}$ is a *refinement* of $\mathcal{C}_i$: any cluster $C \in \mathcal{C}_i$ is the union of a subset of clusters in $\mathcal{C}_{i-1}$.

Representing $\mathcal{C}_i$ by Disjoint Sets. We shall use the classic Union-Find data structure [Tar75] in our clustering procedure, for representing clusters in $\mathcal{C}_i$, grouping subsets of clusters to larger clusters (via the Union operation), and checking whether a pair of vertices belongs to the same cluster (via the Find operation). In particular, each cluster $C \in \mathcal{C}_i$ will have a *representative* vertex, denoted by $r(C)$, that can be accessed from any vertex $v \in C$ by calling Find(v); we define $r(v) :=$ Find(v). The *amortized* time per each Union or Find operation is $O(\alpha(a, b))$, where a is the total number of Union and Find operations and b is the number of vertices in the data structure.

Constructing H^{σ}_i. We assume that $|E^{\sigma}_i| > 0$; otherwise, we will skip the construction at level i and set $\mathcal{C}_{i+1} = \mathcal{C}_i$. We say that a cluster at level i is *isolated* if none of its vertices is incident on any edge of E^{σ}_i; otherwise it is *non-isolated*. Let $\mathcal{X}$ be the set of all non-isolated level-i clusters. We say that two edges (u, v) and (u', v') in E^{σ}_i are *parallel* if $r(u) = r(u')$ (i.e., u and u' are in the same level-i cluster) and $r(v) = r(v')$ (i.e., v and v' are in the same level-i cluster). We say that (u, v) is a self-loop if $r(u) = r(v)$ (i.e., u and v are in the same level-i cluster). Let S_i be obtained from E^{σ}_i by removing from it all self-loops and keeping only the lightest edge in every maximal set of parallel edges of E^{σ}_i; we refer to the edges of S_i as the *source edges*.

We then construct an unweighted graph R_i, called the *representative graph*, as follows: $V(R_i) = \{r(C) : C \in \mathcal{X}\}$ and $E(R_i) = \{(r(u), r(v)) : (u, v) \in S_i\}$. The vertices and edges of R_i are referred to as the *representative vertices* and *representative edges*, respectively; note that each representative edge corresponds to a unique source edge. Let $E'_i \leftarrow$ HalperinZwick$(R_i, 2k-1)$ be the edge set obtained by applying the spanner algorithm of Theorem 2.1 to R_i. Let S'_i be the subset of source edges in S_i corresponding to the representative edges in E'_i. Our graph H^{σ}_i has S'_i as its edge set.

Lemma 3.1. *$d_{H^{\sigma}_{\leq i}}(u, v) \leq (2k-1)(1+O(\epsilon))w(u, v)$ for every edge $(u, v) \in E^{\sigma}_i$, assuming $\epsilon \leq 1/(2g)$. Furthermore, S'_i can be constructed in $O(|E^{\sigma}_i|\alpha(m, n))$ time.*

Proof. Let (u, v) be an arbitrary edge in E^{σ}_i. We first consider the case where $(u, v) \in S_i$. Then, there is an edge $(r(u), r(v)) \in R_i$. By Theorem 2.1, there is a path P between $r(u)$ and $r(v)$ in $(V(R_i), E'_i)$ that contains at most $2k-1$ edges. We write $P = (r(x_0) = r(u), (r(x_0), r(x_1)), r(x_1), (r(x_1), r(x_2)), \ldots, r(x_p) = r(v))$ as an alternating sequence of representative vertices and edges, where $x_0 = u, x_p = v$ and $p \leq 2k-1$. Let $(y^2_\ell, y^1_{\ell+1})$ be the source edge in S'_i that corresponds to the representative edge $(r(x_\ell), r(x_{\ell+1}))$, for each $\ell \in [0, p-1]$. Denote by C_ℓ the level-i cluster with $r(C_\ell) = r(x_\ell)$. Let $y^1_0 = u$ and $y^2_p = v$. Let

$$Q = Q_0(y^1_0, y^2_0) \circ (y^2_0, y^1_1) \circ Q_1(y^1_1, y^2_1) \ldots \circ (y^2_{p-1}, y^1_p) \circ Q_p(y^1_p, y^2_p) \tag{3.2}$$

be a path from u to v, where $Q_\ell(y^1_\ell, y^2_\ell)$ is a shortest path between y^1_ℓ and y^2_ℓ in $H^{\sigma}_{\leq i-1}[C_\ell]$, for each $0 \leq \ell \leq p$, and $\circ$ is the path concatenation operator. By property (P2), $w(Q_\ell(y^1_\ell, y^2_\ell)) \leq gL_{i-1} = g\epsilon L_i$. It follows that

$$\begin{aligned} w(Q) &\leq (2k-1)L_i + (2k)g\epsilon L_i \leq (2k-1)(1+2g\epsilon)L_i \\ &\leq (2k-1)(1+2g\epsilon)(1+\epsilon)w(u, v) \qquad (\text{since } w(u, v) \geq L_i/(1+\epsilon)) \\ &\leq (2k-1)(1+(4g+1)\epsilon)w(u, v) \qquad (\text{since } \epsilon \leq 1) \end{aligned} \tag{3.3}$$

Thus, the stretch of (u, v) is at most $(2k-1)(1+(4g+1)\epsilon)$.

Next, we consider the complementary case that $(u, v) \notin S_i$. By definition, the edge (u, v) is not in S_i either because it is a self-loop or it is parallel to another edge (u', v') that belongs to S_i, with $w(u', v') \leq w(u, v)$. In the former case, property (P2) implies the existence of a path from u to v in $H^{\sigma}_{\leq i-1}$ of weight at most $gL_{i-1} = g\epsilon L_i \leq \frac{L_i}{1+\epsilon} \leq w(u, v)$ when $\epsilon < \frac{1}{2g}$. Thus, in this case the stretch of edge (u, v) is 1. For the latter case, let C_u and C_v be the level-i clusters containing u and v, respectively, and without loss of generality assume that $u' \in C_u$ and $v' \in C_v$. By property (P2), $\mathsf{Dm}(H^{\sigma}_{\leq i-1}[C_u]), \mathsf{Dm}(H^{\sigma}_{\leq i-1}[C_v])) \leq gL_{i-1} = g\epsilon L_i$. The same argument used for deriving Equation (3.3), when applied to the edge (u', v') rather than (u, v), yields:

$$d_{H_{\leq i}}(u', v') \leq (2k-1)(1+(4g+1)\epsilon)w(u', v') \leq (2k-1)(1+(4g+1)\epsilon)w(u, v). \tag{3.4}$$

By the triangle inequality,

$$\begin{aligned} d_{H_{\leq i}}(u,v) \leq\ & d_{H_{\leq i}}(u',v') + \mathsf{Dm}(H_{\leq i-1}[C_u]) + \mathsf{Dm}(H_{\leq i-1}[C_v]) \\ \leq\ & (2k-1)(1+(4g+1)\epsilon)w(u,v) + 2g\epsilon L_i \qquad \text{(by Equation (3.4))} \\ \leq\ & (2k-1)(1+(4g+1)\epsilon)w(u,v) + 4g\epsilon w(u,v) \qquad \text{(since } w(u,v) \geq L_i/(1+\epsilon) \geq L_i/2\text{)} \\ =\ & (2k-1)(1+(8g+1)\epsilon)w(u,v) \qquad \text{(since } k \geq 1\text{)} \end{aligned}$$

Thus, the stretch of edge (u,v) is $(2k-1)(1+(8g+1)\epsilon) = (2k-1)(1+O(\epsilon))$. Summarizing, we have shown that in all cases the stretch of edge (u,v) is at most $(2k-1)(1+O(\epsilon))$, as required.

By construction of the representative graph R_i, all clusters corresponding to vertices of R_i are non-isolated, hence no vertex of R_i is isolated, yielding $|V(R_i)| = O(|E(R_i)|) = O(|E_i^\sigma|)$. Thus, the construction of the edge set S_i and the representative graph R_i, via the usage of the Union-Find data structure, takes total time of $O(|E_i^\sigma|\alpha(m,n))$. The set of edges S_i' by Theorem 2.1 can be constructed in time $O(|E(R_i)| + |V(R_i)|) = O(|E_i^\sigma|)$ time. Thus, the total running time to construct S_i' is $O(|E_i^\sigma|\alpha(m,n))$.

□

Constructing $\mathcal{C}_{i+1}$. Every cluster $C \in \mathcal{C}_i \setminus \mathcal{X}$ becomes a level-$(i+1)$ cluster. We next focus on the level-i clusters of $\mathcal{X}$. Recall that $V(R_i)$ is the set of all representatives of clusters in $\mathcal{X}$. We construct a collection $\mathcal{U}$ of vertex-disjoint subgraphs of R_i in the following two steps:

(1) Initially, we greedily construct a maximal set of vertex-disjoint stars of R_i, and initialize $\mathcal{U}$ as this edge set; thus, each subgraph $U \in \mathcal{U}$ contains a vertex and all of its neighbors in R_i.

(2) We scan the remaining vertices in $V(R_i)$ that haven't been grouped to any subgraph in $\mathcal{U}$. For every such remaining vertex $v \in V(R_i)$, it must have at least one neighbor that is contained in a subgraph $U \in \mathcal{U}$ (by the maximality of $\mathcal{U}$); we add to U the vertex v and an edge (v,u) leading to such a neighbor u of v (chosen arbitrarily if there are multiple such neighbors).

For each subgraph U in the resulting edge set $\mathcal{U}$, we form a level-$(i+1)$ cluster $C_U \in \mathcal{C}_i$ by taking the union of all the clusters whose representatives are $V(U)$ as C_U.

Lemma 3.2. *All clusters in $\mathcal{C}_{i+1}$ satisfy Properties (P1) and (P2) when $\epsilon \leq 1/(2g)$ and $g \geq 9$, and they can be constructed in time $O(|E_i^\sigma| \cdot \alpha(m,n))$. Furthermore, every cluster $C_U \in \mathcal{C}_{i+1}$ that is formed from a subgraph $U \in \mathcal{U}$, as described above, is the union of at least 2 level-i clusters.*

Proof. Property (P1) holds trivially. To prove that Property (P2) holds, we first note that each subgraph $U \in \mathcal{U}$ (with vertices in R_i) has hop diameter at most 4, which follows directly from the above two-step construction of $\mathcal{U}$. Any edge connecting two vertices in U corresponds to a source edge in S_i, and thus also in E_i^σ, and as such has length at most L_i, which implies that C_U induces a subgraph of diameter at most $5(gL_{i-1}) + 4L_i = 5g\epsilon L_i + 4L_i \leq 9L_i = gL_i$, since $\epsilon \leq 1/g$ and $g \geq 9$. Thus, Property (P2) holds.

The construction of the edge set S_i and the representative graph R_i takes total time of $O(|E_i^\sigma|\alpha(m,n))$ using the Union-Find data structure. As for the construction of the collection $\mathcal{U}$ of vertex-disjoint subgraphs of R_i, Step (1) of this construction, i.e., which constructs a maximal set of vertex-disjoint stars, involves a greedy linear-time algorithm, whereas Step (2) naively takes linear time, so together they are implemented within time $O(|E(R_i)|) = O(|E_i^\sigma|)$. Constructing the corresponding clusters $\{C_U : U \in \mathcal{U}\}$ can be implemented within the same amount of time in the obvious way. The construction of clusters in $\mathcal{C}_{i+1}$ that are clusters in $\mathcal{C}_i \setminus \mathcal{X}$ requires no extra time.

Finally, we argue that any cluster $C_U \in \mathcal{C}_{i+1}$ that is formed from a subgraph $U \in \mathcal{U}$ contains at least 2 level-i clusters. Indeed, any cluster formed in Step (1) of the construction of $\mathcal{U}$ contains at least 2 level-i clusters, by the maximality of $\mathcal{U}$ and since no vertex in R_i is isolated. Any remaining level-i cluster must be grouped in Step (2) of the construction of $\mathcal{U}$ to clusters formed in Step (1), and this too holds by the maximality in Step 1 of the construction of $\mathcal{U}$ and since no vertex in R_i is isolated. □

We are now ready to prove the first item of Theorem 1.1.

Proof. [Proof of the first item of Theorem 1.1] Recall that $H = \cup_{1 \leq \sigma \leq \mu_\epsilon} H^\sigma$. Let $\Delta_{i+1} = |\mathcal{C}_i| - |\mathcal{C}_{i+1}|$. Recall that $\mathcal{C}_0$ is the set of n singletons, i.e., $|\mathcal{C}_0| = n$. Thus, $\sum_{i \geq 0} \Delta_{i+1} \leq |\mathcal{C}_0| = n$.

By Lemma 3.2, $\Delta_{i+1} \geq \frac{|V(R_i)|}{2}$. Furthermore, Theorem 2.1 yields $|S'_i| = O(|V(R_i)|^{1+1/k})$, hence $|S'_i| = O(n^{1/k}) \cdot \Delta_{i+1}$. Thus, we have:

$$|E(H^\sigma)| \;=\; |\cup_{i \geq 0} E(H_i^\sigma)| \;=\; \sum_{i \geq 0} |S'_i| \;=\; \sum_{i \geq 0} O(n^{1/k}) \cdot \Delta_{i+1} \;=\; O(n^{1+1/k}). \tag{3.5}$$

The sparsity of H is $O(n^{1/k} \cdot \frac{\log(1/\epsilon)}{\epsilon})$ by Equation (3.5). The stretch of H is at most $(2k-1)(1+O(\epsilon))$ by Lemma 3.1; we can reduce the stretch down to $(2k-1)(1+\epsilon)$ by scaling $\epsilon \leftarrow \epsilon/c$, for a sufficiently large constant c, which will affect the sparsity and runtime bounds by constant factors. The time needed to construct H^σ is $O(\sum_{i \geq 0} |E_i^\sigma| \cdot \alpha(m,n)) = O(m \cdot \alpha(m,n))$ by Lemma 3.1 and Lemma 3.2. Thus, the overall time needed to construct H, when also considering the runtime $O(\mathsf{SORT}(m))$ for computing the partition of E into the sets $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$, is $O(m \cdot \alpha(m,n) \cdot \frac{\log(1/\epsilon)}{\epsilon} + \mathsf{SORT}(m))$. ∎

4 A Linear Time Algorithm in the Transdichotomous Model

In this section, we prove the second item of Theorem 1.1. We follow the same framework as in Section 3; our focus, as before, is on constructing a $(2k-1)(1+O(\epsilon))$-spanner H^σ for E^σ, for a fixed $\sigma \in [1, \mu_\epsilon]$. The construction is carried out in levels, where H_i^σ is constructed at level i, and uses a hierarchy of clusters such that each cluster $C \in \mathcal{C}_i$ satisfies two properties that are similar to those used in Section 3, namely Properties (P1) and (P2).

We also use a Union-Find data structure to represent clusters in $\mathcal{C}_i$. However, our construction relies on a special case of Union-Find , where the set of Union operations are pre-specified at the outset of the construction. Gabow and Tarjan [GT85] designed a data structure for this special case of Union-Find in the Transdichotomous model; this result is summarized in the following theorem.

Theorem 4.1. (Gabow and Tarjan [GT85]) *Let T be a rooted tree with n vertices. One can design a* Union-Find *data structure in the Transdichotomous model that maintains disjoint sets of $V(T)$ and supports m* Union *and* Find *operations in $O(m+n)$ total time, in which each* Union *operation is of the form* Union$(v, p_T(v))$ *for some non-root vertex $v \in V(T)$. Here $p_T(v)$ denotes the parent of v in T.*

We emphasize that the Union-Find data structure of Gabow and Tarjan in Theorem 4.1 only works in the Transdichotomous model. The tree T in Theorem 4.1 is called a *union tree* of the Union-Find data structure. We use Link(v) to specifically denote the Union operation of the form Union$(v, p_T(v))$.

The construction of Section 3 achieves a super-linear running time. To improve this runtime to linear in m, we plug the following new ideas on top of the construction of Section 3.

The second term in the super-linear runtime $O(m \cdot \alpha(m,n) \cdot \frac{\log(1/\epsilon)}{\epsilon} + \mathsf{SORT}(m))$, namely $\mathsf{SORT}(m)$, stems from the time needed to compute the partition of E into the sets $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$, which boils down to sorting the indices of the non-empty sets in $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$. In the Word RAM model, we employ a rather simple trick to carry out such an index sorting in time $O(m \cdot \frac{\log(1/\epsilon)}{\epsilon})$; the details of this optimization appear in Subsection 4.1.

The main obstacle lies in shaving the factor $\alpha(m,n)$ from the first term $O(m \cdot \alpha(m,n) \cdot \frac{\log(1/\epsilon)}{\epsilon})$. For this optimization, the two key ideas are the following:

- **Idea 1.** We use an MST for G as the union tree for the Union-Find data structure. In the Transdichotomous model, Fredman and Willard [FW94] designed an algorithm to construct a minimum spanning tree in $O(m)$ time. Let `MST` be an arbitrary minimum spanning tree for G; we root `MST` at an arbitrary vertex r.
- **Idea 2.** We guarantee that every level-i cluster $C \in \mathcal{C}_i$ *induces* a subtree of `MST` of diameter at most gL_{i-1}, for some constant g. As we will show in the sequel, by forcing clusters to induce subtrees of `MST`, we are able to use Link operations to form level-$(i+1)$ clusters from level-i clusters, which is the source of our speed-up. The crux of our construction is in realizing idea 2.

Theorem 4.1 guarantees that each of the Union and Find operations takes $O(1)$ amortized time. As a result, we shave the $\alpha(m,n)$ factor in the running time of the algorithm from Section 3.

Next we proceed to the details of the linear-time construction. The construction will satisfy the following two properties of clusters in $\mathcal{C}_i$, the first of which is identical to Property (P1) of Section 3 whereas the second is an adaptation of Property (P2).

- **(P1')** Each cluster $C \in \mathcal{C}_i$ is a subset of V. Furthermore, clusters in $\mathcal{C}_i$ induce a partition of V.
- **(P2')** Each cluster $C \in \mathcal{C}_i$ induces a (connected) subtree $\texttt{MST}[C]$ of $\texttt{MST}$ with diameter at most gL_{i-1}, for some constant g (the same constant used in Idea 2 above which is different than the one used in (P2)).

We will add all edges of $\texttt{MST}$ to the spanner, by setting H_0^σ as $\texttt{MST}$, which adds one unit to the sparsity and lightness. Property (P2') is inherently more restrictive than Property (P2), as it aims at guaranteeing the same (perhaps up to a constant factor) diameter bound, but when restricted to subtrees of $\texttt{MST}$.

Representing $\mathcal{C}_i$. As in Section 3, we use the Union-Find data structure to represent clusters in $\mathcal{C}_i$, but we use the data structure provided by Theorem 4.1, which guarantees constant amortized cost. As a result, we will maintain the property that the representative $r(C)$ of any cluster $C \in \mathcal{C}_i$ is always set to be the *root of the subtree* $\texttt{MST}[C]$. By setting the representative of a cluster C to be its root, C can be united with other clusters via $\textsc{Link}(r(C))$, which is crucial for applying the result of Theorem 4.1. The children of C can be united to C by the same way.

Constructing H_i^σ. The construction is the same as the construction of H_i^σ in Section 3. Specifically, we construct a set of level-i clusters $\mathcal{X}$, the representative graph R_i, and the edge set S_i', which is obtained by running the spanner algorithm of Theorem 2.1 to R_i. Since the Union and Find operations now admit $O(1)$ (amortized) time, we derive the following lemma, whose proof follows along similar lines as those in the proof of Lemma 3.1.

Lemma 4.1. *$d_{H_{\le i}}(u,v) \le (2k-1)(1+O(\epsilon))w(u,v)$ for every edge $(u,v) \in E_i^\sigma$, assuming $\epsilon < 1/(2g)$. Furthermore, S_i' can be constructed in $O(|E_i^\sigma|)$ time.*

Constructing $\mathcal{C}_{i+1}$. Our construction of $\mathcal{C}_{i+1}$ relies on the notion of *cluster forest* defined below; see Figure 1 for an illustration.

Definition 1. (Cluster Forest) *Let $\mathcal{Y} \subseteq \mathcal{C}_i$ be a set of level-i clusters. A* cluster forest *for $\mathcal{Y}$, denoted by $\mathcal{F}_\mathcal{Y}$, is a directed forest with a weight function ω on the edges such that:*

(1) Each node $\varphi_C \in \mathcal{F}_\mathcal{Y}$ corresponds to a cluster $C \in \mathcal{Y}$,

(2) There is a directed edge $(\varphi_{C_1} \to \varphi_{C_2})$ in the forest $\mathcal{F}_\mathcal{Y}$ if C_2 contains the parent, say $p_{\texttt{MST}}(v)$, of the representative, say v, of C_1. Furthermore, $\omega(\varphi_{C_1} \to \varphi_{C_2}) = w(v, p_{\texttt{MST}}(v))$.

By definition, every edge of a cluster forest $\mathcal{F}_\mathcal{Y}$ corresponds to an $\texttt{MST}$ edge. Let $\mathcal{MST}_i = \mathcal{F}_{\mathcal{C}_i}$ be the cluster forest defined for the entire set $\mathcal{C}_i$ of level-i clusters; by Property (P2'), it holds that $\mathcal{MST}_i$ is a tree. We stress that $\mathcal{MST}_i$ is only used in the analysis of our algorithm; indeed, computing $\mathcal{MST}_i$, at least naively, would require $\Omega(|\mathcal{C}_i|)$ time, which is too costly.

For a set $\mathcal{Y}$ of level-i clusters, we say that the cluster forest $\mathcal{F}_\mathcal{Y}$ is *L_i-bounded* if every edge in it has weight at most L_i. The following lemma is the crux of our construction.

Recall that $\mathcal{X}$ denotes the set of all non-isolated level-i clusters in the representative graph R_i.

Lemma 4.2. *Let $\mathcal{A}_i$ be the set of $\mathcal{MST}_i$ edges of weight at most L_i, and let $\mathcal{Y}$ be the set of nodes that are incident on at least one edge in $\mathcal{A}_i$. Let $\mathcal{F}_\mathcal{Y}$ be the forest with node set $\mathcal{Y}$ and edge set $\mathcal{A}_i$. Then the following two conditions hold:*

(1) $\mathcal{X} \subseteq \mathcal{Y}$.

(2) Every tree in $\mathcal{F}_\mathcal{Y}^{\mathsf{pruned}}$ has at least 2 nodes.

Proof. Condition (2) follows directly from the construction. We next prove that Condition (1) holds.

Let φ_{C_u} be the node corresponding to a level-i cluster C_u in $\mathcal{X}$. By the definition of $\mathcal{X}$, there is an edge $(u,v) \in E_i^\sigma$ such that $u \in \varphi_{C_u}$. Let C_v be the level-i cluster containing v and φ_{C_v} be the node corresponding to C_v. If $(u,v) \in \texttt{MST}$, then $\varphi_{C_u} \in \mathcal{Y}$, and we're done.

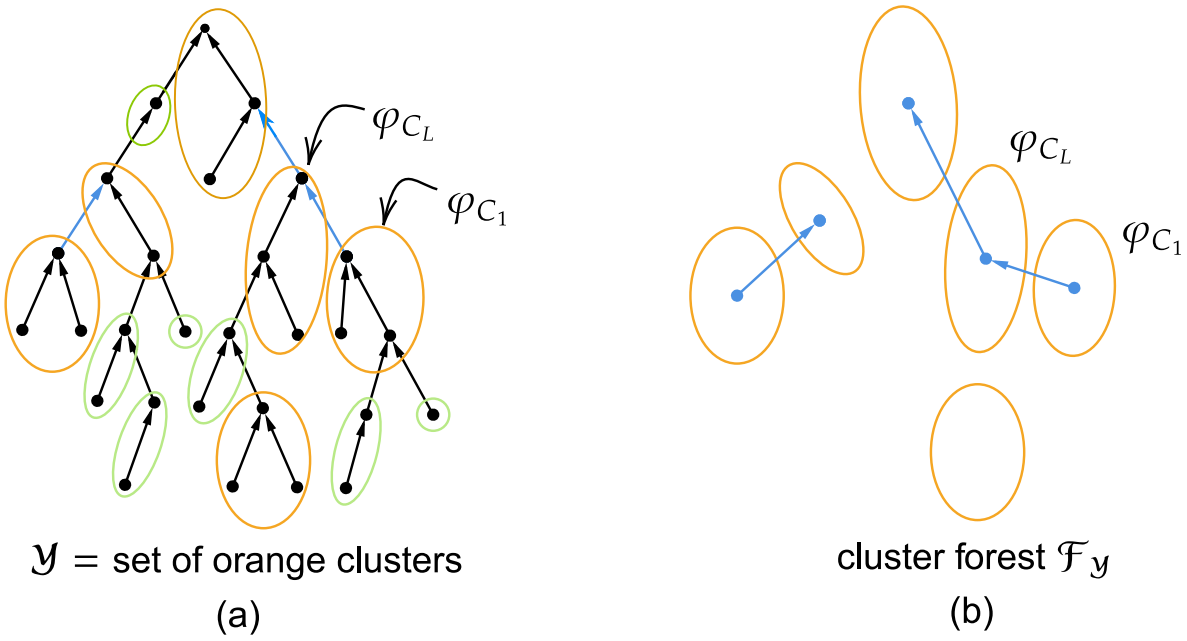

Figure 1: (a) Level-i clusters induce subtrees of MST enclosed by oval curve, and (b) a cluster forest $\mathcal{F}_{\mathcal{Y}}$.

We henceforth assume that $(u, v) \notin \texttt{MST}$. Consider the fundamental cycle C_{uv} of MST formed by $\texttt{MST}[u, v]$ and edge (u, v). By the cycle property of MST, every edge $e \in \texttt{MST}[u, v]$ satisfies $w(e) \leq w(u, v)$. Recall that $\mathcal{MST}_i$ is a tree by Property (P2'). Moreover, by the definition of $\mathcal{MST}_i$, each edge in $\mathcal{MST}_i[\varphi_{C_u}, \varphi_{C_v}]$ corresponds to an edge in $\texttt{MST}[u, v]$, and so has weight at most $w(u, v) \leq L_i$. Hence φ_{C_u} is incident to an edge of $\mathcal{A}_i$ by the definition of $\mathcal{A}_i$, which yields $\varphi_{C_u} \in \mathcal{Y}$. ∎

We now construct the set of level-$(i+1)$ clusters $\mathcal{C}_{i+1}$ as follows. Let $\mathcal{F}_{\mathcal{Y}}$ be the cluster forest for $\mathcal{Y}$ provided by Lemma 4.2. We construct $\mathcal{C}_{i+1}$ as follows. Every level-i cluster $C \in \mathcal{C}_i \setminus \mathcal{Y}$ becomes a level-$(i+1)$ cluster. Then, we construct a collection $\mathbb{U}$ of subtrees of $\mathcal{F}_{\mathcal{Y}}$, such that each subtree $\mathcal{U} \in \mathbb{U}$ contains at least two nodes and has hop-diameter at most 4. For each subtree $\mathcal{U}$, we form a level $(i+1)$ cluster $C_{\mathcal{U}} = \cup_{\varphi_C \in \mathcal{V}(\mathcal{U})} C$. We note that $\mathbb{U}$ can be constructed greedily via the same algorithm used in Section 3, within time $O(|\mathcal{Y}|)$.

In the following lemma we assume that the set of clusters $\mathcal{Y}$ is given to us. In the proof of Theorem 1.1 where we use Lemma 4.3, we will specify the construction of $\mathcal{Y}$.

LEMMA 4.3. *All clusters in $\mathcal{C}_{i+1}$ satisfy Properties (P1') and (P2') when $\epsilon \leq 1/(2g)$ and $g \geq 9$, and they can be constructed in time $O(|\mathcal{Y}|)$. Furthermore, every cluster $C_{\mathcal{U}} \in \mathcal{C}_{i+1}$ that is formed from a subgraph $\mathcal{U} \in \mathbb{U}$, as described above, is the union of at least 2 level-i clusters.*

Proof. The proof of this lemma follows similar lines to those in the proof of Lemma 3.2 from Section Section 3, hence we aim for conciseness. As mentioned, $\mathbb{U}$ can be constructed within time $O(|\mathcal{Y}|)$.

Recall that every edge in $\mathcal{F}_{\mathcal{Y}}$ corresponds to an edge of the form $v \to p_{\texttt{MST}}(v)$ for some vertex $v \in V$. Thus, for each subgraph $\mathcal{U} \in \mathbb{U}$, the level-$(i+1)$ cluster $C_{\mathcal{U}}$ can be constructed by calling $|\mathcal{V}(\mathcal{U})| - 1$ LINK operations. Therefore, $\{C_{\mathcal{U}} : \mathcal{U} \in \mathbb{U}\}$ can be constructed in time $O(\sum_{\mathcal{U} \in \mathbb{U}} |\mathcal{U}|) = O(|\mathcal{Y}|)$. Note that we do not pay any running time for constructing clusters in $\mathcal{C}_{i+1}$ that are clusters in $\mathcal{C}_i \setminus \mathcal{Y}$. Therefore $\mathcal{C}_{i+1}$ can be constructed in $O(|\mathcal{Y}|)$ time.

Property (P1') holds trivially. Property (P2') follows from the fact that each subgraph $\mathcal{U} \in \mathbb{U}$ has hop diameter at most 4 and that each edge between two nodes in $\mathcal{U}$ corresponds to an edge in MST of length at most L_i since every edge of $\mathcal{F}_{\mathcal{Y}}$ has a weight at most L_i by construction.

Note that $\mathbb{U}$ is constructed using same two-step algorithm used in Section 3. Thus, the same argument in Lemma 3.2 applies to this case. Specifically, any cluster formed in Step (1) of the construction of $\mathbb{U}$ contains

at least 2 nodes, since no vertex in $\mathcal{F}_{\mathcal{Y}}$ is isolated, and any remaining node must be grouped in Step (2) of the construction of $\mathbb{U}$ to sugraphs formed in Step (1). ∎

We are now ready to prove the second item of Theorem 1.1.

Proof. [Proof of the second item of Theorem 1.1] Recall that $H = \cup_{1\leq\sigma\leq\mu_\sigma} H^\sigma$. We employ a similar charging argument to the one used in Section 3 to bound $|E(H^\sigma)|$. Let $\Delta_{i+1} = |\mathcal{C}_i| - |\mathcal{C}_{i+1}|$. Note that $|\mathcal{C}_0| = n$, hence $\sum_{i\geq 0}\Delta_{i+1} \leq n$. By Lemma 4.3 and Lemma 4.2, we have $\Delta_{i+1} \geq \frac{|\mathcal{Y}|}{2} = \Omega(|\mathcal{X}|) = \Omega(|V(R_i)|)$. (Note that $|\mathcal{X}| = |V(R_i)|$.) Thus, Equation 3.5 of Section 3 holds in this case as well. It follows that the sparsity of H is $O(n^{1/k}\cdot\frac{\log(1/\epsilon)}{\epsilon})$. The stretch is $(2k-1)(1+O(\epsilon))$ by Lemma 4.1; we can reduce the stretch down to $(2k-1)(1+\epsilon)$ by scaling $\epsilon \leftarrow \epsilon/c$, for a sufficiently large constant c, which will affect the sparsity and runtime bounds by constant factors. The runtime to construct H^σ is $O(\sum_{i\geq 0}|E_i^\sigma|) = O(m)$ by Lemma 4.1.

We now bound the time to construct the clusters in $\mathcal{C}_{i+1}$. The main difficulty is that the size of $\mathcal{Y}$ constructed in Lemma 4.2 could be much larger than $|E_i^\sigma|$, hence we cannot bound the runtime by $|E_i^\sigma|$ as we did in Section 3. Here we employ a more delicate argument. At the outset of the construction, we divide the edges of MST into levels as we did for E_i^σ. The level-i edges of MST, denoted by B_i, include every edge of length larger than L_{i-1} and at most L_i. The time to construct B_i is $O(n\log(1/\epsilon))$, following the same index-sorting argument used for constructing E_i^σ efficiently in Subsection 4.1.

At the outset of the construction of $\mathcal{C}_{i+1}$, we assume that we are given the set of edges $\mathcal{D}_{i-1}$ that contains every edge of weight at most L_{i-1} of $\mathcal{MST}_i$. For level $i = 0$, we set $\mathcal{D}_{i-1} = \emptyset$. Let $\mathcal{B}_i$ be the set of edges of $\mathcal{MST}_i$ corresponding to edges in B_i. The edge set $\mathcal{B}_i$ can be constructed in $O(|B_i|)$ time as follows. For each edge $(u,v) \in B_i$, we add an edge $(\varphi_{C_u}, \varphi_{C_v})$ to $\mathcal{B}_i$, where C_u and C_v are the two level-i clusters containing u and v, respectively, which can be found via FIND(u) and FIND(v).

The set of edges $\mathcal{A}_i$ defined in Lemma 4.2 is $\mathcal{D}_{i-1}\cup\mathcal{B}_i$. Note that $|\mathcal{A}_i| \leq |\mathcal{Y}|$ since $\mathcal{F}_{\mathcal{Y}}$ is acyclic. Thus, the running time to construct $\mathcal{A}_i$ is $O(|\mathcal{Y}|)$, as we have both $\mathcal{D}_{i-1}$ and $\mathcal{B}_i$ stored in a list data structure. To construct the set of $\mathcal{D}_i$ for the construction at the next level, we simply identify edges in $\mathcal{F}_{\mathcal{Y}}$ that are between two different subgraphs $\mathbb{U}$ in the construction of $\mathcal{C}_{i+1}$. Thus, the running time to construct $\mathcal{D}_i$ is also $O(|\mathcal{Y}|)$. The running time to construct $\mathcal{C}_{i+1}$ is $O(|\mathcal{Y}|)$ by Lemma 4.3. It follows that the total running time of the construction of clusters at level i is $O(|\mathcal{Y}|)$. Since $\Delta_{i+1} \geq \frac{|\mathcal{Y}|}{2}$, the time to construct $\mathcal{C}_{i+1}$ is bounded by $O(\Delta_{i+1})$, where $\Delta_{i+1} = |\mathcal{C}_i| - |\mathcal{C}_{i+1}|$. It follows that the total running time to construct clusters over all levels is $\sum_{i\geq 0} O(\Delta_{i+1}) = O(n)$.

In summary, the running time to construct H is $O((m+n)\cdot\frac{\log(1/\epsilon)}{\epsilon}) = O(m\cdot\frac{\log(1/\epsilon)}{\epsilon})$. ∎

4.1 Index sorting in linear time First, we assume that the word size is $\bar{w} \geq \log(n)$ and all edge weights are bounded above by $2^{\bar{w}}$, as per the Word RAM model. The total number of different indices is given by $\log_{1+\epsilon} 2^{\bar{w}} = \Theta(\bar{w}/\epsilon)$. It follows that the number of integers is $n' \leq \bar{w}/\epsilon$. In this range of values, predecessor search can be done in $O(\log(n')/\log\bar{w}) = O(\log(1/\epsilon))$ time using the fusion tree data structure [FW90] (see also [PT06]). Consequently, the time needed to compute the partition of E into the sets $\{E^\sigma\}_{1\leq\sigma\leq\mu_\epsilon}$, which involves index sorting via predecssor search, is bounded by $O(m\log(1/\epsilon))$. Partitioning the set of edges of MST into levels can be done in the same way; the running time is $O(n\log(1/\epsilon))$ as there are $n-1$ edges in MST. Summarizing, the running time of these partitioning steps is bounded by $O((m+n)\log(1/\epsilon)) = O(m\log(1/\epsilon))$.

5 Optimally Sparse and Light Spanners in $O(m\alpha(m,n))$ Time

Le and Solomon [LS21] recently show that a $(2k-1)(1+\epsilon)$-spanner with lightness $O_\epsilon(n^{1/k})$ can be constructed in $O_\epsilon(m\alpha(m,n))$ time; the notation $O_\epsilon(.)$ hides a polynomial factor of $1/\epsilon$. However, their spanner is not sparse, i.e., in the worst case, the number of edges of the spanner is $\Omega(m)$, which could be $\Omega(n^2)$ for dense graphs. Here we use the insights we develop in Section 3 and Section 4 on top of the construction of [LS21] to obtain a $(2k-1)(1+\epsilon)$-spanner that is both sparse and light as claimed in Theorem 1.2.

First, we briefly recap the algorithm of Le and Solomon [LS21], called *LS algorithm*. LS algorithm first divides E into two sets of edges: $E_{\text{light}} = \{e \in: w(e) \leq \frac{w(\text{MST})}{m\epsilon}\}$ and $E_{\text{heavy}} = E \setminus E_{\text{light}}$. Every edge in E_{light} shall be added to the final spanner, and this only incurs an additive $+\frac{1}{\epsilon}$ in the lightness since:

$$w(E_{\text{light}}) \leq m\cdot\frac{w(\text{MST})}{m\epsilon} \leq \frac{w(\text{MST})}{\epsilon}. \tag{5.6}$$

For edges in E_{heavy}, LS algorithm constructs a $(2k-1)(1+\epsilon)$-spanner H_{heavy} that has two properties:

$$\begin{aligned} &(1) \quad w(H_{\text{heavy}}) \leq O_\epsilon(n^{1+1/k}) w(\texttt{MST}) \\ &(2) \quad d_G(u,v) \leq d_{H_{\text{heavy}}}(u,v) \leq (2k-1)(1+\epsilon) d_G(u,v) \quad \forall (u,v) \in E_{\text{heavy}} \end{aligned} \tag{5.7}$$

The final spanner of the graph is $H = H_{\text{heavy}} \cup E_{\text{light}}$. By Equation (5.6) and Equation (5.7), it follows that $w(H) = (O_\epsilon(n^{1/k}) + \frac{1}{\epsilon}) w(\texttt{MST}) = O_\epsilon(n^{1/k}) w(\texttt{MST})$, and hence the lightness of H is $O_\epsilon(n^{1/k})$.

Our first observation is that in LS algorithm, H_{heavy} does not contain any other edge of E_{light}, except for MST edges. It follows that if we construct a $(2k-1)(1+\epsilon)$-spanner H_{light} for the subgraph of G induced by $E_{\text{light}} \cup \texttt{MST}$ by applying the construction in Theorem 1.1, and set $H = H_{\text{light}} \cup H_{\text{heavy}} \cup \texttt{MST}$, then H is still a $(2k-1)(1+\epsilon)$-spanner of G. Furthermore, $w(H) \leq w(E_{\text{light}}) + w(H_{\text{heavy}}) + w(\texttt{MST}) = O_\epsilon(n^{1/k}) w(\texttt{MST})$. Thus, the lightness of H is $O_\epsilon(n^{1/k})$. Observe that $H_{\text{light}} \cup \texttt{MST}$ has sparsity $O_\epsilon(n^{1/k})$. It follows that, to guarantee that H has sparsity $O_\epsilon(n^{1/k})$, we need to construct H_{heavy} such that its sparsity and lightness are both $O_\epsilon(n^{1/k})$. Our construction crucially makes use of the cycle property of MST following the same spirit of the construction in Section 4.

5.1 The construction of H_{heavy} We assume that E_{heavy} has no edges of weight at least $w(\texttt{MST})$ since we could safely remove them from E_{heavy} without affecting the stretch of the construction. The spanner H_{heavy} constructed by LS algorithm is a subgraph of G_{heavy}, which is a subgraph G induced by $E_{\text{heavy}} \cup E(\texttt{MST})$. However, the construction operates on a graph $\tilde{G}$ obtained from G_{heavy} by subdividing edges of MST using *virtual vertices*. Specifically, we define $\bar{w} = w(\texttt{MST})/\epsilon$, and for each edge of $e \in \texttt{MST}$, if $w(e) \geq \bar{w}$, we subdivide e into $\lceil \frac{w(e)}{\bar{w}} \rceil$ edges, each of weight at most $\bar{w}$, whose total weight is $w(e)$. Let $\widetilde{\texttt{MST}}$ be the subdivided MST and $\tilde{G} = (\tilde{V}, E(\widetilde{\texttt{MST}}) \cup E_{\text{heavy}})$. That is, $\tilde{G}$ and G share the same set of edges E_{heavy}. Vertices in $\tilde{V} \setminus V$ are called *virtual vertices*. Le and Solomon [LS21] observed that:

OBSERVATION 2. (OBSERVATION 3.4 IN [LS21]) $|\tilde{V}| = O(m)$.

We now divide E_{heavy} further into subsets $\{E^\sigma\}_{1 \leq \sigma \leq \mu_\epsilon}$ with $\mu_\epsilon = O(\frac{1}{\epsilon}\log(1/\epsilon))$, as we did in Section 3 (Equation (3.1)):

$$E_i^\sigma = \left\{ e : \frac{L_i}{1+\epsilon} \leq w(e) \leq L_i \right\} \text{ with } L_i = L_0/\epsilon^i, L_0 = (1+\epsilon)^\sigma \bar{w} \ . \tag{5.8}$$

We note that constructing every E_i^σ can be done in $O(m)$ by simply sorting all indices i such that $E_i^\sigma \neq \emptyset$. This is because the maximum index $i_{\max}$ is $O(\log(n))$ (Claim 3.5 in [LS21]) and hence, the sorting step takes only $O(\log(n)\log\log(n)) = O(n)$ time.

The construction then focuses on each set E^σ separately. That is, we construct a $(2k-1)(1+O(\epsilon))$-spanner H^σ for each set E^σ in $O(m\alpha(m,n))$ time, and set $H_{\text{heavy}} = \cup_{1 \leq \sigma \leq \mu_\epsilon} H^\sigma$. It follows that the running time to construct H_{heavy} is $O(m\alpha(m,n)/\epsilon)$. Here we slightly abuse notation as H^σ is a subgraph of $\tilde{G}$ instead of being a subgraph of G_{heavy}. However, the difference between $\tilde{G}$ and G_{heavy} lies only in MST vs $\widetilde{\texttt{MST}}$, and by assuming that H^σ contains $\widetilde{\texttt{MST}}$, we can transform H^σ to a subgraph of G_{heavy} by replacing each path of subdividing virtual vertices with the corresponding original edge of MST.

For notational convenience, we set $H_0^\sigma = (\tilde{V}, E(\widetilde{\texttt{MST}}))$. The construction of H^σ happens in *levels*: at level i, we construct a subgraph H_i^σ such that $H_{\leq i}^\sigma$ is a $(2k-1)(1+O(\epsilon))$-spanner for edges in $E_{\leq i}^\sigma$. Here $H_{\leq i}^\sigma = \cup_{0 \leq j \leq i} H_j^\sigma$ and $E_{\leq i}^\sigma = \cup_{0 \leq j \leq i} E_j^\sigma$. Recall that $E_0^\sigma = \emptyset$ since every edge in E_{heavy} has a weight of at least $\bar{w}/\epsilon$. By induction, $H^\sigma \stackrel{\text{def.}}{=} \cup_{i \geq 0} H_i^\sigma$ is a $(2k-1)(1+O(\epsilon))$-spanner for $\tilde{G}$.

Similar to the construction of a sparse spanner in Section 3, we construct a hierarchy of clusters, and each level $i \geq 1$ of the construction is associated with a set of clusters $\mathcal{C}_i$ satisfying the following properties:

(1) Each cluster $C \in \mathcal{C}_i$ is a subset of V. Furthermore, clusters in $\mathcal{C}_i$ induce a partition of $\tilde{V}$.
(2) Each cluster $C \in \mathcal{C}_i$ is the union of $\Omega(1/\epsilon)$ clusters in $\mathcal{C}_{i-1}$ for any $i \geq 2$.
(3) Each cluster $C \in \mathcal{C}_i$ induces a subgraph $H_{\leq i-1}^\sigma[C]$ of diameter at most gL_{i-1} for some constant g.

By property (3), clusters at level 1 are subgraphs of H_0, which is $\widetilde{\texttt{MST}}$. The construction is described in the following lemma.

LEMMA 5.1. (LEMMA 3.8 IN [LS21]) *In time $O(m)$, we can construct a set of level-1 clusters $\mathcal{C}_1$ such that, for each cluster $C \in \mathcal{C}_1$, the subtree $\widetilde{MST}[C]$ of $\widetilde{MST}$ induced by C satisfies $L_0 \leq \mathsf{Dm}(\widetilde{MST}[C]) \leq 14L_0$.*

Thus, by choosing $g \geq 14$, property (3) is satisfied for clusters in $\mathcal{C}_1$. Property (1) follows directly from Lemma 5.1, and property (2) is not applicable.

A crucial component in LS algorithm is a potential function $\Phi : 2^{\tilde{V}} \to \mathbb{R}^+$ that associates each cluster $C \in \mathcal{C}_i$ with a potential value $\Phi(C)$. Let $\Phi_i = \sum_{C \in \mathcal{C}_i} \Phi(C)$ be the total potential value at level i. The potential values of level 1 clusters are defined as follows:

$$\Phi(C) = \mathsf{Dm}(\widetilde{\texttt{MST}}[C]) \quad \forall C \in \mathcal{C}_1 \tag{5.9}$$

By Lemma 5.1, we have that:

$$\Phi_1 = \sum_{C \in \mathcal{C}_1} \mathsf{Dm}(\widetilde{\texttt{MST}}[C]) \leq w(\texttt{MST}) \tag{5.10}$$

Next, Le and Solomon [LS21] define a potential change $\Delta_{i+1} = \Phi_i - \Phi_{i+1}$. Let $i_{\max}$ be the maximum level, and define $\Phi_{i_{\max}+1} = 0$. The idea is to bound the weight of the to-be-constructed spanner H_i^σ by the potential change $O_\epsilon(n^{1/k})\Delta_{i+1}$ (modulo a small additive term that we will describe later). It follows that we can bound the weight of H^σ, again modulo a small additive term, as follows.

$$w(H^\sigma) \leq O_\epsilon(n^{1/k}) \sum_{i=0}^{i_{\max}} \Delta_{i+1} = O_\epsilon(n^{1/k})(\Phi_1 - \Phi_{i_{\max}+1}) = O_\epsilon(n^{1/k})\Phi_1 = O_\epsilon(n^{1/k})w(\texttt{MST}) \tag{5.11}$$

In [LS21], Le and Solomon showed the following lemma, which is the key to their construction.

LEMMA 5.2. (LEMMA 2.6 AND THEOREM 1.9 [LS21]) *For each level $i \geq 1$, there is an algorithm that can compute a subgraph H_i^σ* induced by a subset of *E_i^σ, as well as the set of level-$(i+1)$ clusters $\mathcal{C}_{i+1}$ satisfying properties (1)-(3) given a set of clusters $\mathcal{C}_i$ at level i, such that:*

(1) $w(H_i^\sigma) = O_\epsilon(n^{1/k})\Delta_{i+1} + a_i$ for some $a_i \geq 0$ such that $\sum_{1 \leq i \leq i_{\max}} a_i = O_\epsilon(n^{1/k})w(MST)$.
(2) for every $(u,v) \in E_i^\sigma$, $d_{H_{\leq i}^\sigma}(u,v) \leq (2k-1)(1+(10g+1)\epsilon)w(u,v)$.

Furthermore, the total running time of the construction of all levels *is $O(m\alpha(m,n))$ in the pointer-machine model.*

In Lemma 5.2, a_i is a corrective term added to handle some edge cases where $\Delta_{i+1} = 0$ or even negative. The stretch is $(2k-1)(1+(10g+1)\epsilon)$ instead of $(2k-1)(1+\epsilon)$, but we can obtain the stretch $(2k-1)(1+\epsilon)$ by scaling $\epsilon \leftarrow \frac{\epsilon}{10g+1}$. Note that Lemma 5.2 does not provide any bound on the number of edges of H_i^σ.

To bound the sparsity of H^σ in our construction, we distinguish between *isolated clusters* and *non-isolated clusters*. A cluster $C \in \mathcal{C}_i$ is non-isolated if it contains at least one endpoint of an edge in $E(H_i^\sigma)$, and otherwise, is isolated. By examining the construction of Le and Solomon carefully, we have that:

LEMMA 5.3. (LE AND SOLOMON [LS21]) *Let $\mathcal{Y}_i \subseteq \mathcal{C}_i$ be the set of all non-isolated clusters. Then $|E(H_i^\sigma)| = O_\epsilon(n^{1/k})|\mathcal{Y}_i|$.*

By property (2), the number of clusters is geometrically decreasing when ϵ is sufficiently smaller than 1, and hence, the total number of clusters at all levels is $O(|\mathcal{C}|_1)$. This implies that:

$$|E(H^\sigma)| = \sum_{i \geq 1} |E(H_i^\sigma)| = \sum_{i \geq 1} O_\epsilon(n^{1/k})|\mathcal{C}_i| = O_\epsilon(n^{1/k})|\mathcal{C}_1| = O_\epsilon(n^{1/k})|\tilde{V}| \tag{5.12}$$

However, $|\tilde{V}|$ could be up to $\Omega(m)$ as it contains virtual vertices (Observation 2). Thus, Equation (5.12) does not provide any meaningful bound on the number of edges of H^σ.

We now describe our idea to modify LS algorithm and to bound the number of edges in H_i^σ. For each cluster $C \in \mathcal{C}_i$, we introduce two new types of clusters: *non-virtual clusters*, denoted by $\mathcal{N}_i$, and *virtual clusters*, denoted by $\mathcal{M}_i$. A cluster $C \in \mathcal{C}_i$ is virtual if C only contains virtual vertices, i.e., $C \subseteq \tilde{V} \setminus V$; otherwise C is non-virtual. Since a non-isolated cluster contains at least one non-virtual vertex, which is the endpoint of an edge in $E(H_i^\sigma)$, we have:

OBSERVATION 3. $\mathcal{Y}_i \subseteq \mathcal{N}_i$.

Following the same idea of the construction in Section 4, our goal is to construct a set of cluster $\mathcal{C}_{i+1}$ such that (in addition to properties in Lemma 5.2) $|\mathcal{N}_i| - |\mathcal{N}_{i+1}| = \Omega(|\mathcal{Y}_i|)$. For notational convenience, we define $\mathcal{N}_{i_{\max}+1} = \emptyset$. By the same argument in Section 4 and using Lemma 5.3, we could show that $|E(H^\sigma)| = O_\epsilon(n^{1/k})|\mathcal{N}_1|$. Recall by the definition of non-virtual clusters that $|\mathcal{N}_1| \leq n$. It follows that $|E(H^\sigma)| = O_\epsilon(n^{1+1/k})$, which implies the desired sparsity bound. These ideas are formalized in the following lemma, whose proof is provided in Subsection 5.3.

LEMMA 5.4. *For each level $i \geq 1$, there is an algorithm that can compute a subgraph H_i^σ induced by a subset of E_i^σ, as well as the set of level-$(i+1)$ clusters $\mathcal{C}_{i+1}$ satisfying properties (1)-(3) given a set of clusters $\mathcal{C}_i$ at level i, such that:*

(1) $w(H_i^\sigma) = O_\epsilon(n^{1/k})\Delta_{i+1} + a_i$ for some $a_i \geq 0$ such that $\sum_{1 \leq i \leq i_{\max}} a_i = O_\epsilon(n^{1/k})w(\texttt{MST})$.
(2) for every $(u,v) \in E_i^\sigma$, $d_{H^\sigma_{\leq i}}(u,v) \leq (2k-1)(1+(10g+1)\epsilon)w(u,v)$.
(3) $|E(H_i^\sigma)| = O_\epsilon(n^{1/k})|\mathcal{Y}_i|$.
(4) $|\mathcal{N}_i| - |\mathcal{N}_{i+1}| \geq |\mathcal{Y}_i|/2$.

Furthermore, the total running time of the construction of all levels is $O_\epsilon(m\alpha(m,n))$ in the pointer-machine model.

In the next section, we prove Theorem 1.2, assuming that Lemma 5.4 holds.

5.2 Proof of Theorem 1.2 Recall that $H = H_{\text{light}} \cup H_{\text{heavy}} \cup \texttt{MST}$, where H_{light} is a $(2k-1)(1+\epsilon)$-spanner of E_{light}. By Theorem 1.1, H_{light} can be constructed in $O(m\alpha(m,n)\mathsf{poly}(1/\epsilon) + \mathsf{SORT}(m))$ in the pointer-machine model, and in $O(m\mathsf{poly}(1/\epsilon))$ time in the Transdichotomous model. MST can be constructed in $O(m\alpha(m,n))$ by the pointer-machine model by Chazelle's algorithm [Cha00]. By Lemma 5.4, the running time to construct H^σ is $O(m\alpha(m,n)\mathsf{poly}(1/\epsilon))$, which implies the running time to construct H is $O(m\alpha(m,n)\mathsf{poly}(1/\epsilon))\mu_\epsilon = O(m\alpha(m,n)\mathsf{poly}(1/\epsilon))$. Thus, the running time to construct H is $O(m\alpha(m,n)\mathsf{poly}(1/\epsilon) + \mathsf{SORT}(m))$ in the pointer-machine model and is $O(m\alpha(m,n)\mathsf{poly}(1/\epsilon))$ in the Transdichotomous model.

We now focus on bounding the sparsity and lightness of H. By Item (1) in Lemma 5.4, we have that:

$$\begin{aligned} w(H^\sigma) &= \sum_{i=0}^{i_{\max}} w(H_i^\sigma) = \sum_{i=0}^{i_{\max}} O_\epsilon(n^{1/k})\Delta_{i+1} + a_i \\ &= O_\epsilon(n^{1/k})(\Phi_1) + \sum_{i=0}^{i_{\max}} a_i = O_\epsilon(n^{1/k})w(\texttt{MST}), \end{aligned} \tag{5.13}$$

by Equation (5.10) and Item (2) of Lemma 5.4. Furthermore, by Item (4) of Lemma 5.4, it follows that:

$$\begin{aligned} |E(H^\sigma)| &= \sum_{i=0}^{i_{\max}} |E(H_i^\sigma)| = \sum_{i=0}^{i_{\max}} O_\epsilon(n^{1/k})|\mathcal{Y}_i| \quad \text{(by Item (3) of Lemma 5.4)} \\ &= \sum_{i=0}^{i_{\max}} O_\epsilon(n^{1/k})(|\mathcal{N}_i| - |\mathcal{N}_{i+1}|) \quad \text{(by Item (4) of Lemma 5.4)} \\ &= O_\epsilon(n^{1/k})|\mathcal{N}_1| = O_\epsilon(n^{1+1/k}). \end{aligned} \tag{5.14}$$

It follows that $w(H_{\text{heavy}}) = \sum_{\sigma \in [1,\mu_\epsilon]} w(H^\sigma) = O_\epsilon(n^{1/k})w(\texttt{MST})$ and $|E(H_{\text{heavy}})| = \sum_{\sigma \in [1,\mu_\epsilon]} |E(H^\sigma)| = O_\epsilon(n^{1+1/k})$ since $\mu_\epsilon = O(\log(1/\epsilon)1/\epsilon)$.

Observe that $w(H_{\text{light}}) \leq w(E_{\text{light}}) \leq w(\texttt{MST})/\epsilon$ by Equation (5.6). Furthermore, $|E(H_{\text{light}})| = O_\epsilon(n^{1+1/k})$ by Theorem 1.2. We then conclude that:

$$w(H) \leq w(H_{\text{light}}) + w(H_{\text{heavy}}) = O_\epsilon(n^{1/k})w(\texttt{MST})$$
$$|E(H)| = |E(H_{\text{light}})| + |E(H_{\text{heavy}})| = O_\epsilon(n^{1+1/k}).$$

That is, the sparsity and lightness of H are both $O_\epsilon(n^{1/k})$.

We now bound the stretch of H. Let (u, v) be any edge in G. If (u, v) is in E_{light}, then the stretch of (u, v) is $(2k-1)(1+\epsilon)$ in H_{light}. If $(u, v) \in \texttt{MST}$, then the stretch is 1 since H contains MST as a subgraph. Otherwise, $(u, v) \in E_{\text{heavy}}$, and this means there exist $\sigma \in [1, \mu_\epsilon]$ and $i \in [1, i_{\max}]$ such that $(u, v) \in E_i^\sigma$. By Item (1) in Lemma 5.4, the stretch of (u, v) in $H^\sigma_{\leq i}$, and hence in H_{heavy}, is $(2k-1)(1+(10g+1)\epsilon)$. In summary, the stretch in H of any edge $(u, v) \in E(G)$ is at most $(2k-1)(1+(10g+1)\epsilon)$. By scaling $\epsilon \leftarrow \epsilon/(10g+1)$, we obtain a spanner of stretch $(2k-1)(1+\epsilon)$, and with the same lightness and sparsity bounds.

5.3 Construction of H_i^σ and $\mathcal{C}_{i+1}$ In this section, we construct H_i^σ and $\mathcal{C}_{i+1}$ with properties claimed in Lemma 5.4. Without loss of generality, we assume that ϵ is sufficiently small, and in particular, ϵ is smaller than $1/(c \cdot g)$ for any constant c. We now introduce new notation used in this section.

Notation. We consider graphs with weights on both *edges and vertices* in this section. We define the *augmented weight* of a path to be the total weight of all edges and vertices along the path. The *augmented distance* between two vertices in G is defined as the minimum augmented weight of a path between them in G. The augmented diameter of G is denoted by $\mathsf{Adm}(G)$, which is the maximum pairwise augmented distance in G.

Cluster graphs. The construction of $\mathcal{C}_{i+1}$ is done via a *cluster graph* $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i', \omega)$ that has weights on both edges and nodes (we use nodes to refer to vertices of $\mathcal{G}_i$). Each node $\varphi_C \in \mathcal{V}_i$ corresponds to a level-i cluster C and has weight:

$$\omega(\varphi_C) = \Phi(C) \tag{5.15}$$

That is, the weight of each node is the *potential value* of its corresponding cluster. The edge set $\mathcal{E}_i'$ is the union of two edge sets $\mathcal{E}_i \cup \widetilde{\texttt{MST}}_i$:

- Each edge $\mathbf{e} = (\varphi_{C_u}, \varphi_{C_v}) \in \mathcal{E}_i$ corresponds to an edge $(u, v) \in E_i^\sigma \cup E(\widetilde{\texttt{MST}})$ where C_u and C_v are level-i clusters containing u and v, respectively. Furthermore, $\omega(\mathbf{e}) = w(u, v)$.
- $\mathcal{E}_i$ corresponds to a subset of edges of E_i^σ and $\widetilde{\texttt{MST}}_i$ corresponds to a subset of edges of $\widetilde{\texttt{MST}}$, the subdivided MST. $\widetilde{\texttt{MST}}_i$ induces a minimum spanning tree of $\mathcal{G}_i$, and we abuse notation by denoting $\widetilde{\texttt{MST}}_i$ the spanning tree of $\mathcal{G}_i$ by edges in $\widetilde{\texttt{MST}}_i$.

We refer readers to Lemma 3.16 in [LS21] for the construction of $\mathcal{G}_i$. At a high level, the construction removes edges that are self-loops, parallel edges, and those that have stretch at most $(2k-1)(1+6g\epsilon)$ in $\widetilde{\texttt{MST}}_i$ as these edges already have a good stretch.

LEMMA 5.5. (LEMMA 3.16 AND LEMMA 3.22 [LS21]) *$\mathcal{G}_i$ can be constructed in $O((|\mathcal{V}_i| + |E_i^\sigma|)\alpha(m, n))$ time. Furthermore, if the subset of edges of E_i^σ corresponding to $\mathcal{E}_i$ has stretch $(2k-1)(1+s\epsilon)$ in $H^\sigma_{\leq i}$ for some constant s that only depends on g, then every edge in E_i^σ has stretch $(2k-1)(1+\max\{s+4g, 10g\}\epsilon)$ in $H^\sigma_{\leq i}$ when $\epsilon \leq \frac{1}{\max\{s+4g,10g\}}$.*

Lemma 5.5 implies that it suffices for the construction to focus on constructing a spanner for the subset of edges of E_i^σ that correspond to edges in $\mathcal{E}_i$.

Level-$(i+1)$ clusters. Instead of constructing level-$(i+1)$ directly from level-i clusters, we construct a collection $\mathbb{X}$ of vertex-disjoint subgraphs of $\mathcal{G}_i$. Each subgraph $\mathcal{X} \in \mathbb{X}$ has the vertex set denoted by $\mathcal{V}(\mathcal{X})$ and the edge set denoted by $\mathcal{E}(\mathcal{X})$, and is mapped to a level-$(i+1)$ cluster, denoted by $C_\mathcal{X}$, as follows:

$$C_\mathcal{X} = \cup_{\varphi_C \in \mathcal{V}(\mathcal{X})} C \tag{5.16}$$

That is, $C_\mathcal{X}$ is the union of all level-i clusters corresponding to the nodes of $\mathcal{X}$. Note that $\mathcal{X}$ has weights on both edges and nodes. We then define the potential value of $C_\mathcal{X}$ as follows.

$$\Phi(C_\mathcal{X}) = \mathsf{Adm}(\mathcal{X}) \tag{5.17}$$

That is, the potential value of $C_{\mathcal{X}}$ is the augmented diameter of the corresponding subgraph. Recall that the potential value will then be the weight of the node corresponding to $C_{\mathcal{X}}$ in the cluster graph $\mathcal{G}_{i+1}$ in the construction of the next level, see Equation (5.15). Furthermore, inductively, we can show that, if $\omega(\varphi_C)$ is an upper bound on $\mathsf{Dm}(H^{\sigma}_{\leq i-1}[C])$, then $\mathsf{Adm}(\mathcal{X})$ is an upper bound on $\mathsf{Dm}(H^{\sigma}_{\leq i}[C_{\mathcal{X}}])$. As a result, guaranteeing properties (1)-(3) for level-$(i+1)$ clusters can be translated into guaranteeing the following properties for subgraphs in $\mathbb{X}$:

- **(P1').** $\{\mathcal{V}(\mathcal{X})\}_{\mathcal{X}\in\mathbb{X}}$ is a partition of $\mathcal{V}_i$.
- **(P2').** $|\mathcal{V}(\mathcal{X})| = \Omega(\frac{1}{\epsilon})$.
- **(P3').** $L_i \leq \mathsf{Adm}(\mathcal{X}) \leq gL_i$.

LEMMA 5.6. (LEMMA 3.14 [LS21]) *Let $\mathcal{X}$ be any subgraph in $\mathbb{X}$ satisfying properties (P1')-(P3'). Suppose that every edge $(\varphi_{C_u}, \varphi_{C_v}) \in \mathcal{E}(\mathcal{X})$ corresponds to an edge (u, v) that is added to H^{σ}_i. Then, $C_{\mathcal{X}}$ satisfies all properties (1)-(3).*

We remark that Lemma 5.6 is based on the assumption that (u, v) is added to H^{σ}_i, which we have not constructed yet.

Constructing level$(i+1)$ clusters. Lemma 5.6 translates the construction of clusters in $\mathcal{C}_{i+1}$ to the construction of the set of subgraphs $\mathbb{X}$ satisfying (P1')-(P3'). The main difficulty is not only to satisfy these properties; but also to guarantee that the weight of H^{σ}_i is bounded by the potential change Δ_{i+1} (and a small additive term) as claimed in Item (1) of Lemma 5.4. Recall by Equation (5.15) and Equation (5.17) that:

$$\begin{aligned} \Phi_i &= \sum_{C\in\mathcal{C}_i} \Phi(C) = \sum_{\varphi_C\in\mathcal{V}_i} \omega(\varphi_C) \\ \Phi_{i+1} &= \sum_{C_{\mathcal{X}}\in\mathcal{C}_{i+1}} \Phi(C_{\mathcal{X}}) = \sum_{\mathcal{X}\in\mathbb{X}} \mathsf{Adm}(\mathcal{X}) \end{aligned} \tag{5.18}$$

Thus, if we define the *local potential change* of $\mathcal{X}$ as follows:

$$\Delta_{i+1}(\mathcal{X}) = \sum_{\varphi_C\in\mathcal{V}(\mathcal{X})} \omega(\varphi_{\mathcal{X}}) - \mathsf{Adm}(\mathcal{X}), \tag{5.19}$$

then it follows that:

CLAIM 1. (CLAIM 3.15 [LS21]) $\Delta_{i+1} = \sum_{\mathcal{X}\in\mathbb{X}} \Delta_{i+1}(\mathcal{X})$.

That is, the potential change Δ_{i+1} can be decomposed into local potential changes of subgraphs in $\mathbb{X}$. This meanss we could bound the weight of H^{σ}_i *locally* via bounding the total weight of edges incident to nodes in $\mathcal{X}$ by the local potential change of $\mathcal{X}$.

Partitioning $\mathcal{V}_i$ and $\mathbb{X}$. We say that a partition $\mathbb{V} = \{\mathcal{V}_i^{\mathsf{high}}, \mathcal{V}_i^{\mathsf{low}^+}, \mathcal{V}_i^{\mathsf{low}^-}\}$ of $\mathcal{V}_i$ is a *degree-specific* partition if every node $\varphi_C \in \mathcal{V}_i^{\mathsf{high}}$ is incident to $\Omega(1/\epsilon)$ edges in $\mathcal{E}_i$ and every node $\varphi_C \in \mathcal{V}_i^{\mathsf{low}^+} \cup \mathcal{V}_i^{\mathsf{low}^-}$ is incident to $O(1/\epsilon)$ edges in $\mathcal{E}_i$. That is, $\mathcal{V}_i^{\mathsf{high}}$ is the set of high-degree nodes of $\mathcal{V}_i$ and $\mathcal{V}_i^{\mathsf{low}^+} \cup \mathcal{V}_i^{\mathsf{low}^-}$ is the set of low-degree nodes of $\mathcal{V}_i$. The difference between $\mathcal{V}_i^{\mathsf{low}^+}$ and $\mathcal{V}_i^{\mathsf{low}^-}$ will be made clear later.

We say that a partition $\{\mathbb{X}^{\mathsf{high}}, \mathbb{X}^{\mathsf{low}^+}, \mathbb{X}^{\mathsf{low}^-}\}$ of a collection $\mathbb{X}$ of subgraphs of $\mathcal{G}_i$ *conforms* with a degree-specific partition $\mathbb{V}$ if

(i) Every subgraph $\mathcal{X} \in \mathbb{X}^{\mathsf{low}^-}$ has $\mathcal{V}(\mathcal{X}) \subseteq \mathcal{V}_i^{\mathsf{low}^-}$, and $\cup_{\mathcal{X}\in\mathbb{X}^{\mathsf{low}^-}} \mathcal{V}(\mathcal{X}) = \mathcal{V}_i^{\mathsf{low}^-}$.
(ii) For every node $\varphi_C \in \mathcal{V}_i^{\mathsf{high}}$, there exists a subgraph $\mathcal{X} \in \mathbb{X}^{\mathsf{high}}$ such that $\varphi_C \in \mathcal{V}(\mathcal{X})$, and that every subgraph $\mathcal{X} \in \mathbb{X}^{\mathsf{high}}$ contains at least one node in $\mathcal{V}_i^{\mathsf{high}}$.

Observe that property (ii) implies that $\mathcal{V}(\mathcal{X}) \subseteq \mathcal{V}_i^{\mathsf{low}^+}$ for every $\mathcal{X} \in \mathbb{X}^{\mathsf{low}^+}$. Also, it is possible that a subgraph $\mathcal{X} \in \mathbb{X}^{\mathsf{high}}$ contains a node in $\mathcal{V}_i^{\mathsf{low}^+}$.

The construction of $\mathbb{X}$ in [LS21] is described by the following lemma.

LEMMA 5.7. (LEMMA 3.17 [LS21]) *Given $\mathcal{G}_i$, we can construct in time $O((|\mathcal{V}_i| + |\mathcal{E}_i|)\epsilon^{-1})$ (i) a degree-specific partition $\mathbb{V} = \{\mathcal{V}_i^{\mathsf{high}}, \mathcal{V}_i^{\mathsf{low}^+}, \mathcal{V}_i^{\mathsf{low}^-}\}$ of $\mathcal{V}_i$ and (ii) a collection $\mathbb{X}$ of subgraphs of $\mathcal{G}_i$ along with a partition $\{\mathbb{X}^{\mathsf{high}}, \mathbb{X}^{\mathsf{low}^+}, \mathbb{X}^{\mathsf{low}^-}\}$ conforming $\mathbb{V}$ such that:*

(1) Let $\Delta_{i+1}^+(\mathcal{X}) = \Delta(\mathcal{X}) + \sum_{\mathbf{e} \in \widetilde{MST}_i \cap \mathcal{E}(\mathcal{X})} w(\mathbf{e})$. Then, $\Delta_{i+1}^+(\mathcal{X}) \geq 0$ for every $\mathcal{X} \in \mathbb{X}$, and

$$\sum_{\mathcal{X} \in \mathbb{X}^{\mathsf{high}} \cup \mathbb{X}^{\mathsf{low}^+}} \Delta_{i+1}^+(\mathcal{X}) = \sum_{\mathcal{X} \in \mathbb{X}^{\mathsf{high}} \cup \mathbb{X}^{\mathsf{low}^+}} \Omega(|\mathcal{V}(\mathcal{X})|\epsilon^2 L_i). \tag{5.20}$$

(2) There is no edge in $\mathcal{E}_i$ between a node in $\mathcal{V}_i^{\mathsf{high}}$ and a node in $\mathcal{V}_i^{\mathsf{low}^-}$. Furthermore, if there exists an edge $(\varphi_{C_u}, \varphi_{C_v}) \in \mathcal{E}_i$ such that both φ_{C_u} and φ_{C_v} are in $\mathcal{V}_i^{\mathsf{low}^-}$, then $\mathcal{V}_i^{\mathsf{low}^-} = \mathcal{V}_i$ and $|\mathcal{E}_i| = O(\frac{1}{\epsilon^2})$; this case is called the degenerate case.

(3) For every subgraph $\mathcal{X} \in \mathbb{X}$, $\mathcal{X}$ satisfies the three properties (P1')-(P3') with constant $g = 31$, and $|\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i| = O(|\mathcal{A}_\mathcal{X}|)$ where $\mathcal{A}_\mathcal{X}$ is the set of nodes in $\mathcal{X}$ incident to an edge in $\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i$.

We call $\Delta_{i+1}(\mathcal{X})$ the *corrected potential change* of $\mathcal{X}$. We remark that $\Delta_{i+1}(\mathcal{X})$ could be negative but $\Delta_{i+1}(\mathcal{X})$ is always positive by Item (1) of Lemma 5.7. Furthermore, Item (1) in Lemma 5.7 only tells us about the corrected potential changes of subgraphs in $\mathbb{X}^{\mathsf{high}} \cup \mathbb{X}^{\mathsf{low}^+}$; there is no guarantee on the corrected potential changes of subgraphs in $\mathbb{X}^{\mathsf{low}^-}$ other than non-negativity, and as a result, we could not bound the total weight of edges incident to a subgraph $\mathcal{X} \in \mathbb{X}^{\mathsf{low}^-}$ by the local potential change of $\mathcal{X}$. However, Item (2) means that subgraphs in $\mathbb{X}^{\mathsf{low}^-}$ do not need to "pay for" their incident edges (by their corrected potential changes)—these edges can be paid for by subgraphs in $\mathbb{X}^{\mathsf{high}} \cup \mathbb{X}^{\mathsf{low}^+}$—unless the degenerate case happens, which only incurs a small weight (of $O(1/\epsilon^2)$ edges). Furthermore, subgraphs in $\mathbb{X}^{\mathsf{low}^-}$ do not contain any edge in $\mathcal{E}_i$ by Item (2) of Lemma 5.7 unless the degenerate case happens.

OBSERVATION 4. (OBSERVATION 3.20 IN [LS21]) *If the degenerate case does not happen, for every edge (φ_1, φ_2) with one endpoint in $\mathcal{V}_i^{\mathsf{low}^-}$, the other endpoint must be in $\mathcal{V}_i^{\mathsf{low}^+}$, and hence, $\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i = \emptyset$ if $\mathcal{X} \in \mathbb{X}^{\mathsf{low}^-}$.*

We remark that Item (3) in Lemma 5.7 is slightly different from the corresponding item in Lemma 3.17 [LS21], which is Item (5), in that $|\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i|$ is bounded by $O(|\mathcal{V}(\mathcal{X})|)$. Here we need a slightly stronger bound, and Item (3) can be seen directly from the construction of [LS21]. For completeness, we will show this item in the construction in Subsection 5.3.2.

While the construction in Lemma 5.7 provides a mean to construct H_i^σ and bounding its weight by (corrected) potential changes via Item (1), it does not give us a sufficient reduction in the number of non-virtual clusters as claimed by Items (3) and (4) in Lemma 5.4. The reduction in the number of non-virtual clusters was used to bound the total number of edges of H^σ in Subsection 5.2. Our main contribution is a modification of the construction by Le and Solomon [LS21] using the cycle property of `MST` to achieve the reduction in the number of non-virtual clusters.

We call a node φ_C *virtual* if it corresponds to a virtual cluster C; otherwise, we call φ_C *non-virtual*. We say that φ_C is isolated if C is isolated, and otherwise, is non-isolated. By definition, a non-isolated node is a non-virtual node.

We abuse notation by denoting $\mathcal{N}_i$ and $\mathcal{M}_i$ the sets of non-virtual nodes and virtual nodes of $\mathcal{V}_i$, respectively. We denote by $\mathcal{Y}_i$ the set of non-isolated nodes in $\mathcal{V}_i$. We will show later that $\mathcal{Y}_i$ is exactly the set of nodes defined in Lemma 5.3. That is, every node in $\mathcal{Y}_i$ corresponds to a level-i cluster that contains at least one endpoint of an edge in H_i^σ.

We say that a subgraph $\mathcal{X} \in \mathbb{X}$ is non-virtual if it contains at least one non-virtual node, and otherwise, is virtual. A non-virtual subgraph corresponds to a non-virtual level-$(i+1)$ cluster. Our main contribution is the construction of $\mathbb{X}$ described by the following lemma.

LEMMA 5.8. *We can construct in $O((|\mathcal{V}_i| + |\mathcal{E}_i|)\epsilon^{-1})$ a degree-specific partition $\mathcal{V}$ of $\mathcal{V}_i$ and a collection $\mathbb{X}$ of subgraphs of $\mathcal{G}_i$ that satisfy all properties in Lemma 5.7 with $g = 42$. Furthermore, if we denote by $\mathcal{N}_{i+1}$ the set of non-virtual subgraphs in $\mathbb{X}$, then $|\mathcal{N}_i| - |\mathcal{N}_{i+1}| \geq |\mathcal{Y}_i|/2$.*

In the following section, we prove Lemma 5.4 assuming that Lemma 5.8 holds. The proof of Lemma 5.8 is deferred to Subsection 5.3.2.

5.3.1 Proof of Lemma 5.4 We use the same algorithm in [LS21] to construct H_i^σ. The algorithm has three steps. Initially H_i^σ has no edge.

- **Step 1.** For every subgraph $\mathcal{X} \in \mathbb{X}$, we add to H_i^σ every edge in E_i^σ that corresponds to an edge in $\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i$. The purpose of this step is to guarantee the assumption of Lemma 5.6.
- **Step 2.** Wee use Halperin-Zwick algorithm (Theorem 2.1) to construct a $(2k-1)$-spanner for edges between $\mathcal{V}_i^{\mathsf{high}}$ only. Specifically, we create an *unweighted* graph $\mathcal{K}_i$ that has $\mathcal{V}_i^{\mathsf{high}}$ as the vertex set and the subset of edges of $\mathcal{E}_i$ between $\mathcal{V}_i^{\mathsf{high}}$ as the edge set. Then, we run Halperin-Zwick algorithm [HZ96] on $\mathcal{K}_i$ to obtain an edge set $\mathcal{E}_i^{\mathsf{pruned}}$. We then add every edge in E_i^σ corresponding to an edge in $\mathcal{E}_i^{\mathsf{pruned}}$ to H_i^σ.
- **Step 3.** We add to H_i^σ every edge that corresponding to an edge of $\mathcal{E}_i$ incident to a node in $\mathcal{V}_i^{\mathsf{low}^+} \cup \mathcal{V}_i^{\mathsf{low}^-}$.

Le and Solomon (Lemma 3.22 and Lemma 4.5 in [LS21]) showed that $w(H_i^\sigma) = O_\epsilon(n^{1/k})\Delta_{i+1} + a_i$ for a_i satisfying Lemma 5.4, and that the stretch of every edge $(u,v) \in E_i^\sigma$ is at most $(2k-1)(1+(10g+1)\epsilon)$ in H_i^σ. Since their proof only uses properties stated in Lemma 5.7, and that our construction in Lemma 5.8 also satisfies Lemma 5.7, Items (1) and (2) in Lemma 5.4 hold in our construction as well. We remark that the additive term a_i is used to handle the degenerate case in Item (2) of Lemma 5.7, since in that case, $\Delta_{i+1} \le 0$.

We now focus on proving Items (3) and (4) of Lemma 5.4. First, we observe that for every node φ_C that is incident to an edge $\mathbf{e} \in \mathcal{E}_i$, the corresponding edge of $\mathbf{e}$ in E_i^σ is added to H_i^σ, unless $\varphi_C \in \mathcal{V}_i^{\mathsf{high}}$ and Halperin-Zwick algorithm does not pick $\mathbf{e}$ to $\mathcal{E}_i^{\mathsf{pruned}}$. In this exceptional case, another edge incident to φ_C must be picked to $\mathcal{E}_i^{\mathsf{pruned}}$; otherwise, φ_C is not connected to any node in the graph induced by $\mathcal{E}_i^{\mathsf{pruned}}$, contradicting that the output is a spanner. It follows that $\mathcal{Y}_i$ corresponds to level-i clusters that have at least one incident edge in H_i^σ. Thus, Item (4) of Lemma 5.4 follows from Lemma 5.8.

By Item (3) in Lemma 5.7, the total number of edges added in Step 1 is $O(1)\sum_{\mathcal{X}\in\mathbb{X}} \mathcal{A}_{\mathcal{X}} = O(1)|\mathcal{Y}_i|$. The number of edges added in Step 2 is $|\mathcal{E}_i^{\mathsf{pruned}}| = O(|\mathcal{V}_i^{\mathsf{high}}|^{1+1/k}) = O(n^{1/k})|\mathcal{V}_i^{\mathsf{high}}| = O(n^{1/k})|\mathcal{Y}_i|$ since $\mathcal{V}_i^{\mathsf{high}} \subseteq \mathcal{Y}_i$ by the definition of non-isolated nodes. In Step 3, for each node $\varphi_C \in \mathcal{V}_i^{\mathsf{low}^+} \cup \mathcal{V}_i^{\mathsf{low}^-}$, we add at most $O(1/\epsilon)$ incident edges to H_i^σ since nodes in $\mathcal{V}_i^{\mathsf{low}^+} \cup \mathcal{V}_i^{\mathsf{low}^-}$ have degree $O(1/\epsilon)$. Thus, the total number of edges added in Step 3 is $O(1/\epsilon)|\mathcal{Y}_i|$. Item (3) of Lemma 5.4 now follows.

For the running time, we first note that constructing $\mathcal{G}_i$ takes $O((|\mathcal{V}_i|+|\mathcal{E}_i|)\alpha(m,n))$ time by Lemma 5.5. The set of subgraphs $\mathbb{X}$ is constructed in $O_\epsilon(|\mathcal{V}_i|+|\mathcal{E}_i|)$ time by Lemma 5.8. In the construction of H_σ^i, Steps 1 and 3 take $O(|\mathcal{V}_i|+|\mathcal{E}_i|)$ by a straightforward implementation. Step 2 takes $O(|\mathcal{V}_i|+|\mathcal{E}_i|)$ time by Theorem 2.1. Thus, the total running time of the construction at level i is $O((|\mathcal{V}_i|+|\mathcal{E}_i|)\alpha(m,n))$. It follows that the total running time over all levels is:

$$\begin{aligned}
\sum_{i\ge 1} O_\epsilon\left((|\mathcal{V}_i|+|\mathcal{E}_i|)\alpha(m,n)\right) &= O_\epsilon\left(\left(\sum_{i\ge 1}|\mathcal{V}_i|+m\right)\alpha(m,n)\right) \\
&= O_\epsilon\left((|\tilde{V}|+m)\alpha(m,n)\right) \qquad \text{(by property (P2))} \\
&= O_\epsilon\left(m\alpha(m,n)\right) \qquad \text{(by Observation 2)}
\end{aligned}$$

Lemma 5.4 now follows.

5.3.2 The construction of $\mathbb{X}$ Recall that $\widetilde{\texttt{MST}}$ is the tree obtained by subdividing MST edges by virtual vertices. For each edge $e \in \texttt{MST}$, we denote by P_e the corresponding path of MST subdivided from e. We call P_e the *subdivided path* of e. Since each virtual cluster $C \in \mathcal{C}_i$ only contains virtual vertices, C induces a subpath of the subdivided path P_e of some edge $e \in \texttt{MST}$. We call P_e the *parent path* of C, and e the *parent edge* of C; see Figure 2(a). We also refer to P_e as the parent path and to e as the parent edge of the virtual node φ_C corresponding to C.

Our goal is to construct $\mathbb{X}$ satisfying all properties in Lemma 5.7, and such that there is a significant reduction in the number of non-isolated clusters as claimed in Lemma 5.4. To guarantee this additional constraint, we rely on a specific structure of $\mathcal{G}_i$ described in the following lemma, which is an analogous version of the cycle property of the minimum spanning tree.

LEMMA 5.9. *Let $\mathbf{e} = (\varphi_1, \varphi_2)$ be any edge in $\mathcal{E}_i$, and $\mathcal{Z}$ the fundamental cycle of $\widetilde{\texttt{MST}}_i$ formed by $\mathbf{e}$. For any virtual node $\varphi \in \mathcal{Z}$, $w(e_\varphi) \le \omega(\mathbf{e})$ where e_φ is the parent edge of φ.*

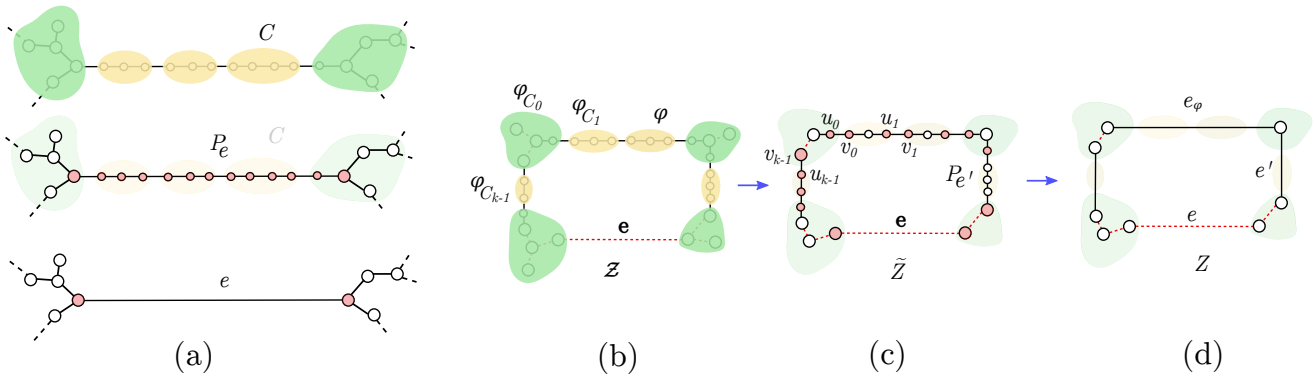

Figure 2: Virtual clusters are yellow shaded and non-virtual clusters are green shaded. (a) A virtual cluster C, its parent path P_e, and the edge e that corresponds to P_e. (b) The fundamental cycle $\mathcal{Z}$ of $\widetilde{\texttt{MST}}_i$ formed by an edge $\mathbf{e}$. (c) The corresponding cycle $\tilde{Z}$ in $\tilde{G}_{\text{heavy}}$ corresponding to $\mathcal{Z}$. Shaded nodes are $u_0, v_0, u_1, v_1, \ldots, u_{k-1}, v_{k-1}$. (d) The cycle Z of G_{heavy} corresponding to $\tilde{Z}$ obtained by replacing each subdivided path $P_{e'}$ with the corresponding edge e' in MST. Solid (black) edges are MST edges, and red (dashed) edges are non-MST edges.

Proof. Recall that $\tilde{G}_{\text{heavy}}$ is obtained from G_{heavy} by subdividing MST edges, and that $G_{\text{heavy}} = (V, E_{\text{heavy}} \cup E(\texttt{MST}))$. Let e be the edge in G_{heavy} corresponding to $\mathbf{e}$. We construct a cycle $\tilde{Z}$ of $\tilde{G}_{\text{heavy}}$ from $\mathcal{Z}$ as follows. Write

$$\mathcal{Z} = (\varphi_{C_0}, \mathbf{e}_0, \varphi_{C_1}, \mathbf{e}_1, \ldots, \varphi_{C_{k-1}}, \mathbf{e}_{k-1}, \varphi_{C_0})$$

as an alternating sequence of nodes and edges that starts from and ends at the same node φ_{C_0}. (See Figure 2(a) and (b) for an illustration.) For notational convenience, we regard the last node φ_{C_0} as φ_{C_k} with the subscript modulo k. Let (u_i, v_i) be the edge in $\tilde{G}_{\text{heavy}}$ corresponding to $\mathbf{e}_i$, and Q_i be the shortest path from $v_{(i-1) \mod k}$ to u_i in $H^{\sigma}_{\leq i}[C_i]$ for any $0 \leq i \leq k-1$. Then $\tilde{Z} = (u_0, v_0) \circ Q_1 \circ (u_1, v_1) \circ Q_2 \circ \ldots \circ (u_{k-1}, v_{k-1}) \circ Q_{k-1}$ is a cycle of $\tilde{G}$; here $\circ$ is the path concatenation operator. Observe that $\tilde{Z}$ contains the parent path, say P_e, of φ. Let Z be the cycle of G_{heavy} obtained from $\tilde{Z}$ by replacing each subdivided path say $P_{e'}$ in $\tilde{Z}$ with the corresponding MST edge e'; see Figure 2(d). Note that both e and e_φ belong to Z.

Observe by property (P3) that $\mathsf{Dm}(H^{\sigma}_{\leq i}[C_i]) \leq g\epsilon L_i < L_i/(1+\epsilon) \leq \omega(\mathbf{e}) = w(e)$ when $\epsilon \leq 1/(2g)$. Thus, the weight of any non-MST edge in Z is at most $w(e)$. That is, any edge of weight larger than $w(e)$ in Z must be an MST edge. If there exists such an edge, then the edge of maximum weight in Z is an MST edge, contradicting the cycle property of MST. Thus, e is an edge of maximum weight in Z, which gives $w(e_\varphi) \leq w(e) = \omega(\mathbf{e})$ as claimed. ∎

Note by definition that a non-isolated node is a non-virtual node. We say that a subgraph $\mathcal{X}$ is *good* if it either contains no non-isolated node or if it contains one non-isolated node, it has at least two non-virtual nodes (one of which is the non-isolated node). If every subgraph in $\mathbb{X}$ is good, then we could show that the number of non-virtual clusters is reduced by at least $|\mathcal{Y}_i|/2$. In LS construction, which has five steps, only subgraphs formed in Steps 2 and 5 (more precisely, Step 5B) may not be good. For Step 5B, only need to make a minor modification and argue that the resulting subgraph is good using Lemma 5.9. For Step 2, we need an entirely different construction. As a result, our construction also has five steps. Steps 1,3, 4 and 5A are the same as the LS construction, and are taken verbatim from [LS21] for completeness. Notation introduced in this section is summarized in the following table.

Notation	**Meaning**
E_{light}	$\{e \in E(G) : w(e) \leq w/\varepsilon\}$
E_{heavy}	$E \setminus E^{light}$
E^{σ}	$\bigcup_{i \in \mathbb{N}^+} E^{\sigma}_i$

E_i^σ	$\{e \in E(G) : \frac{L_i}{1+\epsilon} < w(e) \leq L_i\}$
H_i^σ	A spanner constructed for edges in E_i^σ
$H_{\leq i}^\sigma$	$H_{\leq i}^\sigma = \cup_{j \leq i} H_i^\sigma$
g	constant in property (P3), $g = 42$
Non-virtual cluster	A cluster containing at least one non-virtual vertex
Non-virtual node	A node in $\mathcal{V}_i$ corresponding to a non-virtual cluster
$\mathcal{N}_i$	the set of non-virtual clusters (nodes) at level i
$\mathcal{M}_i$	the set of virtual clusters (nodes) at level i
Non-isolated cluster	A cluster containing an endpoint of an edge added to H_i^σ
Non-isolated node	A node in $\mathcal{V}_i$ corresponding to a non-isolated cluster
$\mathcal{Y}_i$	the set of non-isolated clusters (nodes) at level i; $\mathcal{Y}_i \subseteq \mathcal{N}_i$
$\mathcal{G}_i = (\mathcal{V}_i, \widetilde{\mathtt{MST}}_i \cup \mathcal{E}_i, \omega)$	cluster graph
$\mathcal{E}_i$	corresponds to a subset of edges of E_i^σ
$\mathbb{X}$	a collection of subgraphs of $\mathcal{G}_i$
$\mathcal{X}, \mathcal{V}(\mathcal{X}), \mathcal{E}(\mathcal{X})$	a subgraph in $\mathbb{X}$, its vertex set, and its edge set
Good subgraph $\mathcal{X}$	$\mathcal{X}$ contains no non-isolated node or at least two non-virtual nodes
Φ_i	$\sum_{c \in C_i} \Phi(c)$
Δ_{i+1}	$\Phi_i - \Phi_{i+1}$
$\Delta_{i+1}(\mathcal{X})$	$(\sum_{\phi_C \in \mathcal{X}} \Phi(C)) - \Phi(C_\mathcal{X})$
$\Delta^+_{i+1}(\mathcal{X})$	$\Delta_{i+1}(\mathcal{X}) + \bigcup_{e \in \mathcal{E}(\mathcal{X}) \cap \widetilde{\mathtt{MST}}_i} w(e)$
$C_\mathcal{X}$	$\bigcup_{\phi_C \in \mathcal{X}} C$
$\{\mathcal{V}_i^{\mathsf{high}}, \mathcal{V}^{\mathsf{low}^+}, \mathcal{V}_i^{\mathsf{low}^-}\}$	a degree-specific partition of $\mathcal{V}_i$
$\{\mathbb{X}^{\mathsf{high}}, \mathbb{X}^{\mathsf{low}^+}, \mathbb{X}^{\mathsf{low}^-}\}$	A partition of $\mathbb{X}$ conforming a degree-specific partition.

Table 2: Notation introduced in this section

LEMMA 5.10. (STEP 1, LEMMA 5.1 [LS21]) *Let $\mathcal{V}_i^{\mathsf{high}}$ be the set of nodes incident to at least $2g/\epsilon$ edges in $\mathcal{E}_i$, and $\mathcal{V}_i^{\mathsf{high}+}$ be the set of all nodes in $\mathcal{V}_i^{\mathsf{high}}$ and their neighbors that are connected via edges in $\mathcal{E}_i$. We can construct in $O(|\mathcal{V}_i| + |\mathcal{E}_i|)$ time a collection of node-disjoint subgraphs $\mathbb{X}_1$ of $\mathcal{G}_i$ such that:*

(1) Each subgraph $\mathcal{X} \in \mathbb{X}_1$ is a tree.
(2) $\cup_{\mathcal{X} \in \mathbb{X}_1} \mathcal{V}(\mathcal{X}) = \mathcal{V}_i^{\mathsf{high}+}$.
(3) $L_i \leq \mathsf{Adm}(\mathcal{X}) \leq 13 L_i$, assuming that $\epsilon \leq 1/g$.
(4) $|\mathcal{V}(\mathcal{X})| \geq \frac{2g}{\epsilon}$.

Let $\widetilde{F}_i^{(2)}$ be the forest obtained from $\widetilde{\mathtt{MST}}_i$ by removing every node in $\mathcal{V}_i^{\mathsf{high}+}$ (defined in Lemma 5.10). LS algorithm deals with branching nodes of $\widetilde{F}^{(2)}$ in Step 2. We say that a node in a tree $\widetilde{T}$ is $\widetilde{T}$-*branching* if it has degree at least 3 in $\widetilde{T}$. A node in a forest $\widetilde{F}$ is $\widetilde{F}$-branching if it is $\widetilde{T}$-branching in some tree $\widetilde{T}$ of $\widetilde{F}$. We will omit the prefixes $\widetilde{T}$ and $\widetilde{F}$ in the branching notation whenever the tree and the forest are clear from the context.

Similar to LS algorithm, our goal is to group all branching nodes of $\widetilde{F}_i^{(2)}$ into subgraphs. However, we need to guarantee that subgraphs formed in this step are good, which a priori, are not guaranteed to be good in LS construction.

LEMMA 5.11. *We can construct in $O(|\mathcal{V}_i|)$ time a collection $\mathbb{X}_2$ of subtrees of $\widetilde{F}_i^{(2)}$ and a subset of nodes $\mathcal{Z}$ of $\widetilde{F}_i^{(2)}$ such that, for every $\mathcal{X} \in \mathbb{X}_2$:*

(1) $\mathcal{X}$ *is a tree, has an* $\mathcal{X}$*-branching node, and is good.*
(2) $L_i \le \mathsf{Adm}(\mathcal{X}) \le 20L_i$.
(3) $|\mathcal{V}(\mathcal{X})| = \Omega(\frac{1}{\epsilon})$ *when* $\epsilon \le 2/g$.
(4) Let $\widetilde{F}_i^{(3)}$ *be obtained from* $\widetilde{F}_i^{(2)}$ *by removing every node contained in subgraphs of* $\mathbb{X}_2$ *and in* $\mathcal{Z}$*. Then, for every tree* $\widetilde{T} \subseteq \widetilde{F}_i^{(3)}$*, either (4a)* $\mathsf{Adm}(\widetilde{T}) \le 6L_i$ *or (4b)* $\widetilde{T}$ *is a path.*
(5) Nodes in $\mathcal{Z}$ *are augmented to subgraphs in* $\mathbb{X}_1$ *such that for every subgraph* $\mathcal{Y} \in \mathbb{X}_1$ *that are augmented,* $\mathcal{Y}^{\mathsf{aug}}$ *remains a tree and* $\mathsf{Adm}(\mathcal{Y}^{\mathsf{aug}}) \le 24L_i$ *where* $\mathcal{Y}^{\mathsf{aug}}$ *is* $\mathcal{Y}$ *after the augmentation.*

There are two differences in the construction of Step 2 in our construction compared to the construction in LS algorithm. First, the graphs constructed are good. Second, for some edges cases where we could not group branching nodes into subgraphs satisfying Item (1), we show that they could be augmented to subgraphs in $\mathbb{X}_1$. These nodes are in $\mathcal{Z}$ in Item (5), and our construction guarantees that the augmentation does not change the structure of subgraphs in $\mathbb{X}_1$. That is, subgraphs in $\mathbb{X}$ remain trees, and their diameters are not increased by much. The increase in the diameter from $13L_i$ in Lemma 5.10 to $24L_i$ in Item (5) in Lemma 5.11 does not affect the overall argument of Le and Solomon [LS21]; this only affects the choice of g, which we have the freedom to choose as large as we want. The augmented diameter of $\mathcal{X}$ in Item (2) in Lemma 5.11 is also slightly larger than the diameter of subgraphs in [LS21], which is at most $2L_i$. This change also only affects the choice of g. The proof of Lemma 5.11 will be delayed to Subsection 5.3.3.

Step 3: Augmenting $\mathbb{X}_1 \cup \mathbb{X}_2$. We say that a path of augmented diameter at least $6L$ in the forest $\widetilde{F}_i^{(3)}$ in Item (4) of Lemma 5.11 a *long path*. In this step, we further augment graphs formed in Steps 1 and 2. The purpose is to guarantee that for any long path after this step, at least one endpoint of the path is connected to a node in a subgraph of $\mathbb{X}_1 \cup \mathbb{X}_2$ via an $\widetilde{\mathtt{MST}}_i$ edge.

> **The construction.** Let $\mathcal{A}$ be the set of all nodes in a long path of $\widetilde{F}_i^{(3)}$ that is $\widetilde{\mathtt{MST}}_i$-branching. For each node $\varphi \in \mathcal{A}$, let $\mathcal{X} \in \mathbb{X}_1 \cup \mathbb{X}_2$ be (any) subgraph such that φ is connected to a node in $\mathcal{X}$ via an $\widetilde{\mathtt{MST}}_i$ edge $\mathbf{e}$. We then add φ and $\mathbf{e}$ to $\mathcal{X}$.

LEMMA 5.12. (LEMMA 5.3. [LS21]) *The augmentation in Step 3 can be implemented in* $O(|\mathcal{V}_i|)$ *time, and increases the augmented diameter of each subgraph in* $\mathbb{X}_1 \cup \mathbb{X}_2$ *by at most* $4L_i$ *when* $\epsilon \le 1/g$.
Furthermore, let $\widetilde{F}_i^{(4)}$ *be the forest obtained from* $\widetilde{F}_i^{(3)}$ *by removing every node in* $\mathcal{A}$*. Then, for every tree* $\widetilde{T} \subseteq \widetilde{F}_i^{(4)}$*, either:*

(1) $\mathsf{Adm}(\widetilde{T}) \le 6L_i$ *or*
(2) $\widetilde{T}$ *is a path such that (2a) every node in* $\widetilde{T}$ *has* degree at most 2 *in* $\widetilde{\mathtt{MST}}_i$ *and (2b) at least one endpoint* φ *of* $\widetilde{T}$ *is connected via an* $\widetilde{\mathtt{MST}}_i$ *edge to a node* φ' *in a subgraph of* $\mathbb{X}_1 \cup \mathbb{X}_2$*, unless* $\mathbb{X}_1 \cup \mathbb{X}_2 = \emptyset$.

We emphasize that in Item (2a) of Lemma 5.12, the degree bound is in $\widetilde{\mathtt{MST}}_i$. This is important for the construction in Step 5. Step 4 deals with long paths of $\widetilde{F}_i^{(4)}$, the forest in Lemma 5.12. The construction uses Red/Blue Coloring. The coloring guarantees that for any long path in $\widetilde{F}_i^{(4)}$, the nodes in the prefix/suffix of augmented length at most L_i get red color, while other nodes get blue color.

> **Red/Blue Coloring.** The coloring applies to each long path $\widetilde{P} \in \widetilde{F}_i^{(4)}$. Specifically, a node gets red color if its augmented distance to at least one of the two endpoints of $\widetilde{P}$ is at most L_i; otherwise, it gets blue color.

LEMMA 5.13. (STEP 4, LEMMA 5.4 [LS21]) *We can construct in* $O((|\mathcal{V}_i| + |\mathcal{E}_i|)\epsilon^{-1})$ *time a collection* $\mathbb{X}_4$ *of subgraphs of* $\mathcal{G}_i$ *such that every* $\mathcal{X} \in \mathbb{X}_4$*:*

(1) $\mathcal{X}$ *contains a single edge in* $\mathcal{E}_i$.
(2) $L_i \le \mathsf{Adm}(\mathcal{X}) \le 5L_i$.
(3) $|\mathcal{V}(\mathcal{X})| = \Theta(\frac{1}{\epsilon})$ *when* $\epsilon \ll \frac{1}{g}$.
(4) $\Delta^+_{i+1}(\mathcal{X}) = \Omega(\epsilon^2 |\mathcal{V}(\mathcal{X})| L_i)$.
(5) Let $\widetilde{F}_i^{(5)}$ *be obtained from* $\widetilde{F}_i^{(4)}$ *by removing every node contained in subgraphs of* $\mathbb{X}_4$*. If we apply Red/Blue Coloring to each path of augmented diameter at least* $6L_i$ *in* $\widetilde{F}_i^{(5)}$*, then there is no edge in* $\mathcal{E}_i$ *that connects two blue nodes in* $\widetilde{F}_i^{(5)}$.

Item (5) of Lemma 5.13 guarantees that for any edge with one endpoint in a long path of $\widetilde{F}_i^{(5)}$, at least one of the endpoints must have red color. $\widetilde{F}_i^{(5)}$ has the following structure.

OBSERVATION 5. (OBSERVATION 5.7 [LS21]) *Every tree $\widetilde{T} \subseteq \widetilde{F}_i^{(5)}$ of augmented diameter at least $6L_i$ is connected via $\widetilde{\texttt{MST}}_i$ edge to a node in some subgraph $\mathcal{X} \in \mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4$, unless there is no subgraph formed in Steps 1-4, i.e., $\mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4 = \emptyset$.*

We observe that any tree $\widetilde{T} \subseteq \widetilde{F}_i^{(5)}$ of diameter at least $6L_i$ must be a path, and that, by Item (2a) in Lemma 5.12, only endpoints of $\widetilde{T}$ could have an edge in $\widetilde{\texttt{MST}}_i$ to a node outside $\widetilde{T}$. We call such an endpoint a *connecting endpoint* of $\widetilde{T}$. Note that $\widetilde{T}$ could have up to two connecting endpoints.

Step 5 has two smaller steps. In Step 5A, we augment trees of $\widetilde{F}_i^{(5)}$ of low augmented diameter to existing subgraphs. In Step 5B, we form new subgraphs from long paths, and augment the prefix/suffix to an existing subgraph in previous steps.

Step 5. Let $\widetilde{T}$ be a path in $\widetilde{F}_i^{(5)}$ obtained by Item (5) of Lemma 5.13. We construct two sets of subgraphs, denoted by $\mathbb{X}_5^{\mathsf{intrnl}}$ and $\mathbb{X}_5^{\mathsf{pref}}$.

- (Step 5A) If $\widetilde{T}$ has augmented diameter at most $6L_i$, let $\mathbf{e}$ be an $\widetilde{\texttt{MST}}_i$ edge connecting $\widetilde{T}$ and a node in some subgraph $\mathcal{X} \in \mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4$, assuming that $\mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4 \neq \emptyset$. We add both $\mathbf{e}$ and $\widetilde{T}$ to $\mathcal{X}$.
- (Step 5B) Otherwise, $\widetilde{T}$ is a path. We break $\widetilde{T}$ into subpaths of augmented diameter at least L_i and at most $7L_i$ by applying the construction in Lemma 5.14 below. For any subpath $\widetilde{P}$ broken from $\widetilde{T}$, if $\widetilde{P}$ is connected to a node in a subgraph $\mathcal{X}$ via an edge $\mathbf{e} \in \widetilde{\texttt{MST}}_i$, we add $\widetilde{P}$ and $\mathbf{e}$ to $\mathcal{X}$; otherwise, $\widetilde{P}$ becomes a new subgraph. We add $\widetilde{P}$ to $\mathbb{X}_5^{\mathsf{pref}}$ if it is a prefix/suffix of $\widetilde{T}$; otherwise, we add $\widetilde{P}$ to $\mathbb{X}_5^{\mathsf{intrnl}}$.

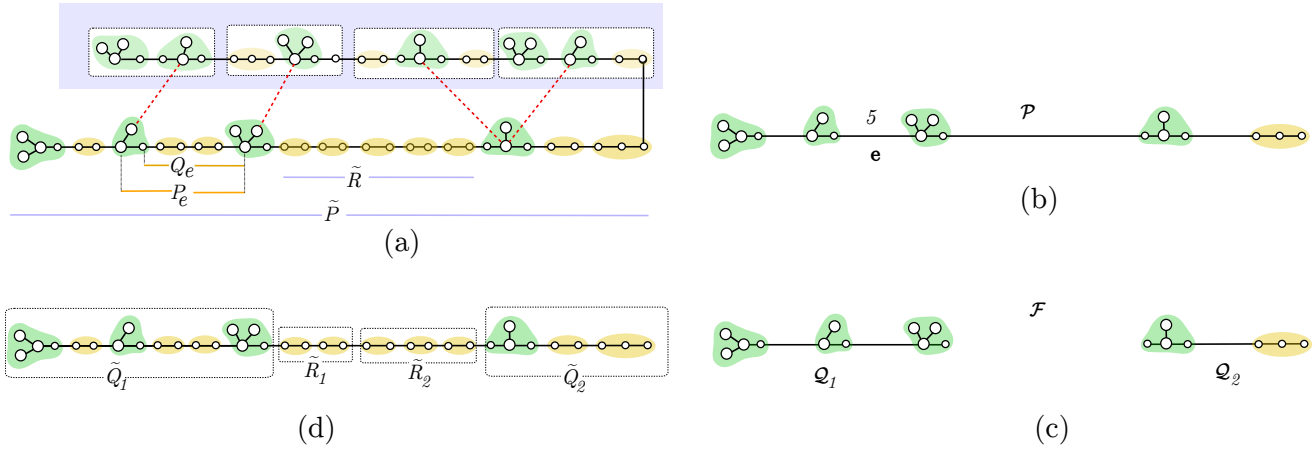

Figure 3: An illustration for the proof of Lemma 5.14. Small circles are virtual vertices; black (solid) edges are $\widetilde{\texttt{MST}}$ edges and red (dashed) edges are edges in $\mathcal{E}_i$. (a) Non-isolated nodes in $\widetilde{P}$ are those incident to red edges. Nodes grouped in previous steps are in the blue-shaded region. The path Q_e corresponds to an $\mathbf{e}$ in $\mathcal{P}$ is highlighted. Q_e is a subpath of the parent path P_e of the virtual clusters in the construction of $\mathbf{e}$. (b) The path $\mathcal{P}$ obtained from $\widetilde{P}$ in figure (a) by the construction in the proof of Lemma 5.14; the only virtual node in $\mathcal{P}$ is the (connecting) endpoint of $\mathcal{P}$. Suppose that every edge in Q_e in figure (a) has weight 1, then $\mathbf{e}$ has weight 5 since Q_e has 5 edges. In general, $\omega(\mathbf{e}) = w(Q_e)$. (c) Forest $\mathcal{F}$ obtained from $\mathcal{P}$ by removing every edge of weight at least $2L_i$. $\mathbb{A}$ in this case includes two paths $\mathcal{Q}_1$ and $\mathcal{Q}_2$. (d) Two paths $\tilde{Q}_1$ and $\tilde{Q}_2$ in $\mathbb{P}$ constructed from $\mathcal{Q}_1$ and $\mathcal{Q}_2$, respectively. Two other paths $\tilde{R}_1$ and $\tilde{R}_2$ are broken from the path $\tilde{R}$ in (a).

LEMMA 5.14. *Let $\widetilde{P}$ be a path of augmented diameter at least $6L_i$ in $\widetilde{F}_i^{(5)}$. We can break $\widetilde{P}$ into a collection of paths $\mathbb{P}$ such that each path $\widetilde{P}' \in \mathbb{P}$ has two properties:*

(1) $L_i \leq \mathsf{Adm}(\widetilde{P}') \leq 7L_i$.
(2) If $\widetilde{P}'$ *contains a non-isolated node, then it contains at least two non-virtual nodes, or a connecting endpoint of* $\widetilde{P}$.

The running time of the construction is $O(|\mathcal{V}(\widetilde{P})|)$.

Proof. Recall that by Item (2a) in Lemma 5.12, every node in $\widetilde{P}$ has degree 2 in $\widetilde{\texttt{MST}}_i$. This means, if an endpoint of $\widetilde{P}$ is non-connecting, then it is a non-virtual node. Recall by the definition of a virtual node φ_C, its corresponding cluster C is virtual, and hence, structurally, C induced a subpath of the parent path P_e.

We construct path graph $\mathcal{P}$ from $\widetilde{P}$ that contains non-virtual nodes and the endpoints of $\widetilde{P}$ as follows. Each edge $\mathbf{e} = (\varphi, \varphi') \in \mathcal{P}$ corresponds to a path between φ and φ' in $\widetilde{T}$ whose internal nodes are virtual. Note that all virtual nodes on the path between φ and φ' in $\widetilde{T}$ share the same parent path P_e. Let Q_e be the minimal subpath of P_e whose endpoints are in the clusters corresponding to φ and φ'. We then assign a weight $\omega(\mathbf{e}) = w(Q_e)$. Observe that $\omega(\mathbf{e}) \leq w(P_e) = w(e)$ where e is the $\texttt{MST}$ edge from which P_e is subdivided. See Figure 3(a) and (b) for an illustration.

Note by Item (2a) of Lemma 5.11, every node in $\widetilde{P}$ has degree at most 2 in $\widetilde{\texttt{MST}}_i$. If φ is a non-isolated node in $\widetilde{P}$, then it is incident to an edge, say $\mathbf{e}'$, in $\mathcal{E}_i$ by definition. One of the incident edges of φ is part of the fundamental cycle of $\widetilde{\texttt{MST}}_i$ formed by $\mathbf{e}'$. It follows from Lemma 5.9 that at least one edge in $\mathcal{P}$ of φ must have a weight at most L_i.

Let $\mathcal{F}$ be the forest induced by edges of weight at most $2L_i$ in $\mathcal{P}$. We further remove singletons from $\mathcal{F}$. Observe that a singleton in $\mathcal{F}$ is either a connecting endpoint of $\mathcal{P}$, or an isolated node. We then greedily break each path in $\mathcal{F}$ that contains at least three edges into subpaths of at least two edges and at most three edges each. As a result, we obtain a collection $\mathbb{A}$ of subpaths of $\mathcal{P}$ that contain at least two nodes each. See Figure 3(c).

We now construct $\mathbb{P}$ as follows. (Step 1) For each path $\mathcal{Q} \in \mathbb{A}$, we construct the corresponding subpath $\widetilde{Q}$ of $\widetilde{P}$ by replacing each edge in $\mathcal{Q}$ by the corresponding subpath in $\widetilde{P}$. We then add $\widetilde{Q}$ to $\mathbb{P}$. (Step 2) After Step 1, remaining nodes in $\widetilde{P}$ that are not grouped to a path in $\mathbb{P}$ induces a collection of subpaths, say $\mathbb{Q}$, of $\widetilde{P}$. Observe by the construction of $\mathcal{F}$ that, each subpath in the collection $\mathbb{Q}$ corresponds to a subpath of $\mathcal{P}$, which only contains virtual nodes and isolated nodes, that has at least one edge of weight at least $2L_i$. Now for each path $\tilde{R} \in \mathbb{Q}$, observe that $\mathsf{Adm}(\tilde{R}) \geq 2L_i - 2\bar{w} - 2g\epsilon L_i \geq 2L_i - 4g\epsilon L_i \geq L_i$ when $\epsilon \leq 1/2g$. The negative term $-2\bar{w} - 2g\epsilon L_i$ is due to that the two nodes neighboring the endpoints of $\tilde{R}$ are grouped to subpaths in $\mathbb{P}$. We then break $\tilde{R}$ into subpaths of augmented diameter at least L_i and at most $2L_i$ and add them to $\mathbb{P}$. This completes the construction of $\mathbb{P}$. See Figure 3(d) for an illustration.

The running time follows directly from the construction. To bound the augmented diameter of paths in $\mathbb{P}$, we observe that path $\widetilde{Q}$ in Step 1 has augmented diameter at most $3(2L_i) + 4\epsilon g L_i \leq 7L_i$ when $\epsilon \leq 1/4g$. The additive term $4\epsilon g L_i$ is due to (at most) four endpoints of (at most) three edges in $\widetilde{Q}$. Thus, every path in $\mathbb{P}$ has an augmented diameter of at most $\max\{7L_i, 2L_i\} = 7L_i$. The lower bound L_i follows directly from the construction; this implies Item (1). Item (2) follows from the construction of $\mathbb{A}$. ∎

We note that in Step 5B in LS algorithm, $\widetilde{T}$ is broken into subpaths of augmented length at least L_i and at most $2Li$ instead of at least L_i and at most $7L_i$ as in our construction. The increase in the augmented diameter ultimately affects the choice of g. Other properties of subgraphs in $\mathbb{X}_5^{\mathsf{intrnl}}$ and $\mathbb{X}_5^{\mathsf{pref}}$ remains the same.

LEMMA 5.15. (LEMMA 5.8 [LS21]) *We can implement the construction of* $\mathbb{X}_5^{\mathsf{intrnl}}$ *and* $\mathbb{X}_5^{\mathsf{pref}}$ *in* $O(|\mathcal{V}_i|)$ *time. Furthermore, every subgraph* $\mathcal{X} \in \mathbb{X}_5^{\mathsf{intrnl}} \cup \mathbb{X}_5^{\mathsf{pref}}$ *satisfies:*

(1) $\mathcal{X}$ *is a subpath of* $\widetilde{MST}_i$.
(2) $L_i \leq \mathsf{Adm}(\mathcal{X}) \leq 7L_i$.
(3) $|\mathcal{V}(\mathcal{X})| = \Theta(\frac{1}{\epsilon})$.

We note that the degenerate case in the above construction happens when $\mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4 = \emptyset$. When the degenerate case happens, $\widetilde{F}_i^{(5)}$ has the following structure.

LEMMA 5.16. (LEMMA 5.10 [LS21]) *If* $\mathbb{X}_1 \cup \mathbb{X}_2 \cup \mathbb{X}_4 = \emptyset$, *then* $\widetilde{F}_i^{(5)} = \widetilde{MST}_i$, *and* $\widetilde{MST}_i$ *is a single (long) path. Moreover, every edge* $\mathbf{e} \in \mathcal{E}_i$ *must be incident to a node in* $\widetilde{P}_1 \cup \widetilde{P}_2$, *where* $\widetilde{P}_1$ *and* $\widetilde{P}_2$ *are the prefix and suffix subpaths of* $\widetilde{MST}_i$ *of augmented diameter at most* L_i. *Furthermore,* $|\mathcal{E}_i| = O(1/\epsilon^2)$.

We are now ready to prove Lemma 5.8.

Proof of Lemma 5.8. The degree-specific partition $\mathbb{V}$ of $\mathcal{V}_i$ and the partition of $\mathbb{X}$ conforming $\mathbb{V}$ are constructed as follows. If the degenerate case happens, then $\mathcal{V}_i^{\mathsf{low}^-} = \mathcal{V}_i$ (and hence $\mathcal{V}_i^{\mathsf{high}} = \mathcal{V}_i^{\mathsf{low}^+} = \emptyset$). In this case, $\mathbb{X}^{\mathsf{low}^-} = \mathbb{X}_5^{\mathsf{intrnl}} \cup \mathbb{X}_5^{\mathsf{pref}}$, while $\mathbb{X}^{\mathsf{high}} = \mathbb{X}^{\mathsf{low}^+} = \emptyset$. Otherwise, $\mathcal{V}_i^{\mathsf{high}}$ to be the set of all nodes that are incident to at least $2g/\epsilon$ edges in $\mathcal{E}_i$ in Lemma 5.10, $\mathcal{V}_i^{\mathsf{low}^-} = \cup_{\mathcal{X} \in \mathbb{X}_5^{\mathsf{intrnl}}} \mathcal{V}(\mathcal{X})$ and $\mathcal{V}_i^{\mathsf{low}^+} = \mathcal{V}_i \setminus (\mathcal{V}_i^{\mathsf{high}} \cup \mathcal{V}_i^{\mathsf{low}^-})$. The partition of $\mathbb{X}$ is $\{\mathbb{X}^{\mathsf{high}} = \mathbb{X}_1, \mathbb{X}^{\mathsf{low}^+} = \mathbb{X}_2 \cup \mathbb{X}_4 \cup \mathbb{X}_5^{\mathsf{pref}}, \mathbb{X}^{\mathsf{low}^-} = \mathbb{X}_5^{\mathsf{intrnl}}\}$.

We note that Items (1) and (2) in Lemma 5.7 hold by the same proof in [LS21]. For Item (3), subgraphs in $\mathbb{X}$ satisfy all properties (P1')-(P3') with constant $g = 42$ instead of 31 since the construction of Step 2 in Lemma 5.11 increases the augmented diameter of subgraphs in $\mathbb{X}_1$ by $11L_i$ (on top of the upper bound $31L_i$). We remark that the augmented diameter of other subgraphs is smaller than the augmented diameters of subgraphs in $\mathbb{X}_1$, and hence, the increased diameter due to our construction does not affect g. The fact that $|\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i| = O(|\mathcal{A}_\mathcal{X}|)$ where $\mathcal{A}_\mathcal{X}$ is the set of nodes in $\mathcal{X}$ incident to an edge in $\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i$ follows from that $\mathcal{X}$ is a tree for all cases, except in Step 4 (Lemma 5.13). However, in this case, $\mathcal{X}$ has a single edge in $\mathcal{E}_i$, and hence $|\mathcal{E}(\mathcal{X}) \cap \mathcal{E}_i| \leq 1 = O(|\mathcal{A}_\mathcal{X}|)$.

It remains to show the reduction in the number of non-virtual clusters as claimed in Lemma 5.8. All we need to show is that for every subgraph $\mathcal{X}$ that contains a non-isolated node, it contains at least two non-virtual nodes. That is, $\mathcal{X}$ is good. This holds for subgraphs in $\mathbb{X}_1 \cup \mathbb{X}_4$, since every subgraph in this set contains at least one edge in $\mathcal{E}_i$, whose endpoints are non-isolated by the definition of a non-isolated node. Every subgraph in $\mathbb{X}_2$ is good by Item (1) in Lemma 5.11. Observe that each subgraph $\mathcal{X} \in \mathbb{X}_5^{\mathsf{intrnl}} \cup \mathbb{X}_5^{\mathsf{pref}}$ corresponds to a subpath of $\widetilde{T}$ in Step 5B that does not contain the connecting endpoint. By Item (2) in Lemma 5.14, $\mathcal{X}$ contains at least two non-virtual nodes, if it contains at least one non-isolated node, and hence $\mathcal{X}$ is good. Lemma 5.8 now follows.

5.3.3 Proof of Lemma 5.11

In this section, we provide the proofs of Lemma 5.11, which we restate below.

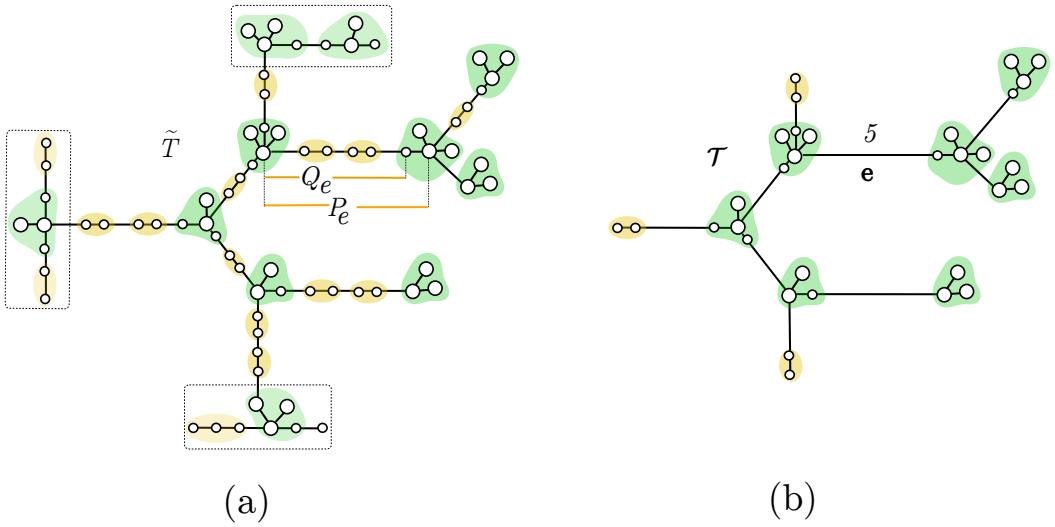

Figure 4: Virtual clusters are yellow shaded and non-virtual clusters are green shaded. Virtual vertices are small circles. (a) A tree $\widetilde{T}$ considered in the construction of Step 2 (Lemma 5.11). (b) The tree $\mathcal{T}$ constructed from non-virtual nodes and connecting nodes of $\widetilde{T}$ in the proof of Lemma 5.11. If every edge in the path Q_e has weight 1 as in figure (a), then the weight of $\mathbf{e}$ in figure (b) is 5. In general, $\omega(\mathbf{e}) = w(Q_e)$. Every virtual node in $\mathcal{T}$ is a connecting node. Subgraphs in the rectangular dashed curves are subgraphs formed in previous steps.

LEMMA 5.11. *We can construct in $O(|\mathcal{V}_i|)$ time a collection $\mathbb{X}_2$ of subtrees of $\widetilde{F}_i^{(2)}$ and a subset of nodes $\mathcal{Z}$ of $\widetilde{F}_i^{(2)}$ such that, for every $\mathcal{X} \in \mathbb{X}_2$:*

(1) $\mathcal{X}$ is a tree, has an $\mathcal{X}$-branching node, and is good.
(2) $L_i \leq \mathsf{Adm}(\mathcal{X}) \leq 20L_i$.

(3) $|\mathcal{V}(\mathcal{X})| = \Omega(\frac{1}{\epsilon})$ *when* $\epsilon \leq 2/g$.

(4) Let $\widetilde{F}_i^{(3)}$ *be obtained from* $\widetilde{F}_i^{(2)}$ *by removing every node contained in subgraphs of* $\mathbb{X}_2$ *and in* $\mathcal{Z}$*. Then, for every tree* $\widetilde{T} \subseteq \widetilde{F}_i^{(3)}$*, either (4a)* $\mathsf{Adm}(\widetilde{T}) \leq 6L_i$ *or (4b)* $\widetilde{T}$ *is a path.*

(5) Nodes in $\mathcal{Z}$ *are augmented to subgraphs in* $\mathbb{X}_1$ *such that for every subgraph* $\mathcal{Y} \in \mathbb{X}_1$ *that are augmented,* $\mathcal{Y}^{\mathsf{aug}}$ *remains a tree and* $\mathsf{Adm}(\mathcal{Y}^{\mathsf{aug}}) \leq 24L_i$ *where* $\mathcal{Y}^{\mathsf{aug}}$ *is* $\mathcal{Y}$ *after the augmentation.*

Proof. Let $\widetilde{T}$ be a tree of augmented diameter at least $6L_i$ in $\widetilde{F}^{(2)}$. We say that a node $\varphi \in \widetilde{T}$ is a connecting node if it has an $\mathtt{MST}$ edge to a subgraph $\mathcal{X} \in \mathbb{X}_1$.

We now construct a tree $\mathcal{T}$ in the same way we construct a path $\mathcal{P}$ in Lemma 5.14. $\mathcal{T}$ is a tree that contains non-virtual nodes and connecting nodes of $\widetilde{T}$, which may or may not be virtual. Note that branching nodes of $\widetilde{T}$ are non-virtual. Each edge $\mathbf{e} = (\varphi, \varphi') \in \mathcal{T}$ corresponds to a path between φ and φ' in $\widetilde{T}$ whose internal nodes are virtual. Note that all virtual nodes on the path between φ and φ' in $\widetilde{T}$ share the same parent path P_e. Let Q_e be the minimal subpath of P_e whose endpoints are in the clusters corresponding to φ and φ'. We then assign a weight $\omega(\mathbf{e}) = w(Q_e)$. Observe that $\omega(\mathbf{e}) \leq w(P_e) = w(e)$ where e is the $\mathtt{MST}$ edge from which P_e is subdivided. See Figure 4 for an illustration.

Claim 2. *If a node* φ *in* $\widetilde{T}$ *is non-isolated and non-connecting, then* φ *is incident to an edge of weight at most* L_i *in* $\mathcal{T}$*.*

Proof. By definition of a non-isolated node, φ is incident to an edge, say $\mathbf{e}'$, in $\mathcal{E}_i$ by definition. One of the incident edges of φ belongs the fundamental cycle of $\widetilde{\mathtt{MST}}_i$ formed by $\mathbf{e}'$. It follows from Lemma 5.9 that at least one edge in $\mathcal{T}$ of φ must have a weight at most $\omega(\mathbf{e}') \leq L_i$. ∎

We first apply the following construction to obtain a collection of trees, say $\mathbb{A}$, and then we will post-process the trees to obtain $\mathbb{X}_2$ as claimed in Lemma 5.11. We say that a tree $\widetilde{T}$ in $\widetilde{F}^{(2)}$ a long tree if its augmented diameter is at least $6L_i$. The construction of $\mathbb{A}$ is similar to Step 2 in LS algorithm, except that the radius of the BFS step in our construction is slightly larger.

- (Step i) Pick a long tree $\widetilde{T}$ of $\widetilde{F}_i^{(2)}$ with at least one $\widetilde{T}$-branching node, say φ. If $\widetilde{T}$ has a $\widetilde{T}$-branching node that is non-isolated, we then choose φ to be a non-isolated node. We traverse $\widetilde{T}$ by BFS starting from φ and *truncate* the traversal at nodes whose augmented distance from φ is at least $2L_i$. The augmented radius (with respect to the center φ) of the subtree induced by the visited nodes is at least L_i and at most $2L_i + \bar{w} + g\epsilon L_i \leq 2L_i + 2g\epsilon L_i$. We then create a new tree $\widetilde{T}'$ induced by the visited nodes.

After the construction in Step i, every tree in $\widetilde{T}$ either has augmented diameter at most $6L_i$ or is a path.

An important property that we would like to have is that every tree in $\mathbb{A}$ either contains no non-isolated node or at least two non-virtual nodes. To this end, we need to post-process $\mathbb{A}$. Our postprocessing relies on the following structure of trees in $\mathbb{A}$.

Claim 3. *Let* $\widetilde{T}' \in \mathbb{A}$ *be a tree that contains exactly one non-isolated node, no connecting node, and no other non-virtual node. Then* $\widetilde{T}'$ *is adjacent to a tree* $\mathcal{T}'' \in \mathbb{A}$ *that has at least two non-virtual nodes.*

Proof. Let φ be the non-isolated node in $\widetilde{T}'$. Observe that the center of $\widetilde{T}'$ is a branching node, and hence, is non-virtual. It follows that φ must be the center of $\widetilde{T}'$ since otherwise, $\widetilde{T}'$ contains two non-virtual nodes, contradicting the assumption of the claim. Let φ' be the neighbor in $\mathcal{T}$ of φ whose edge (φ', φ) has weight at most L_i by Claim 2. By construction, the radius of the traversal is at least $2L_i > L_i + 2\epsilon g L_i$ when $\epsilon \leq 1/4g$. If φ' is a virtual node (see Figure 5(a)), then it must be connecting, and hence φ' belong to $\widetilde{T}'$, contradicting that $\widetilde{T}'$ has no connecting node. Otherwise, φ' is a non-virtual node and is grouped into another tree, say $\widetilde{T}'' \in \mathbb{A}$ (see Figure 5(b)). Observe that $\widetilde{T}'$ and $\widetilde{T}''$ are adjacent, i.e., connected by an edge in $\widetilde{\mathtt{MST}}_i$, since all nodes between φ and φ' have degree 2 as they are virtual nodes. We claim that $\widetilde{T}''$ must have at least two non-virtual nodes. If φ' is not a center of $\widetilde{T}''$, then $\widetilde{T}''$ contains at least two non-virtual nodes since its center is a non-virtual node. Otherwise, φ' is the center of $\widetilde{T}''$, and hence, φ would have been merged to $\widetilde{T}''$ during the construction of $\widetilde{T}''$, a contradiction. ∎

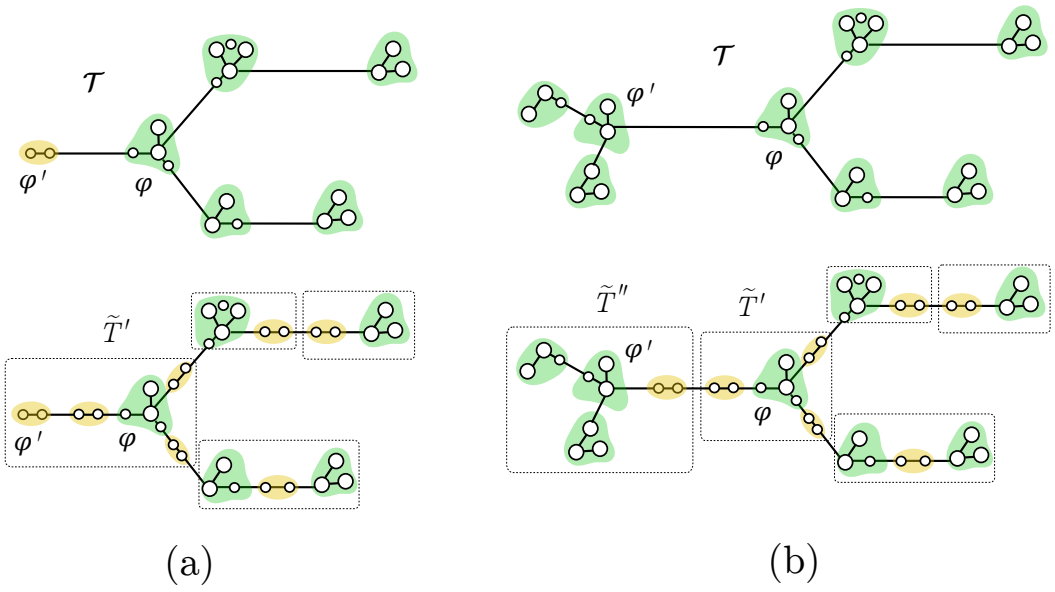

Figure 5: Two cases in the proof of Claim 3. (a) φ' is a virtual node in $\mathcal{T}$. Then it is a connecting node, and is grouped to $\widetilde{T}'$. (b) φ' is a non-virtual node. Then it is grouped in $\widetilde{T}''$ that is adjacent to $\widetilde{T}'$. $\widetilde{T}''$ contains at least two non-virtual nodes (3 non-virtual nodes in this figure).

Our construction in the next step is as follows.

- (Step ii) Pick a tree $\widetilde{T}'$ in $\mathbb{A}$ that has one non-isolated node and no other non-virtual node. If $\widetilde{T}'$ contains a connecting node, say φ. Let $\mathcal{Y} \in \mathbb{X}_1$ be a subgraph such that φ has an $\widetilde{\texttt{MST}}_i$ edge $\mathbf{e}$ to a node in $\mathcal{Y}$. We then add $\widetilde{T}'$ and $\mathbf{e}$ to $\mathcal{Y}$, and add the set of nodes of $\widetilde{T}'$ to $\mathcal{Z}$. Otherwise, $\widetilde{T}'$ is adjacent to another tree $\widetilde{T}'' \in \mathbb{A}$ that has at least two non-virtual nodes by Claim 3. We then add $\widetilde{T}'$ and the $\widetilde{\texttt{MST}}_i$ edge connecting $\widetilde{T}'$ and $\widetilde{T}''$ to $\widetilde{T}''$. We then repeat this step until it no longer applies. The set $\mathbb{X}_2$ is the set of trees in $\mathbb{A}$ after this step completed.

We now prove all properties in Lemma 5.11. Step i is the same as Step 2 in LS algorithm and hence can be implemented in $O(\mathcal{V}_i)$ following [LS21] (Lemma 5.2). Step ii can be implemented in $O(|\mathcal{V}(\widetilde{F}^{(2)})|) = O(\mathcal{V}_i)$ by following each step of the construction. Thus, the total running time is $O(|\mathcal{V}_i|)$.

Item (1) of Lemma 5.11 and Item (4) follows directly from the construction. By the construction in Step i, every tree has an augmented diameter at least $2L_i$ and at most $(2L_i + 2\epsilon L_i)$. The augmentation in Step ii is done via a star-like way, and hence, increases the diameter of each tree in $\mathbb{A}$ by at most $2(2L_i + 2\epsilon g L_i) + 2\bar{w} \leq 2(2L_i + 2\epsilon g L_i) + 2g\epsilon L_i = 4L_i + 6\epsilon L_i$. (Here we use the fact that $\bar{w} \leq L_{i-1} = \epsilon L_i \leq g\epsilon L_i$.) Thus, the final diameter is at most $2L_i + 2\epsilon g L_i + 4L_i + 6\epsilon L_i \leq 6L_i + 8\epsilon L_i \leq 14L_i < 20L_i$ when $\epsilon \leq 1/g$; this implies Item (2) of Lemma 5.11.

For Item (3), note that each tree $\mathcal{X} \in \mathbb{X}_2$ has augmented diameter at least $2L_i$, and that every edge/node has a weight at most $\max\{\bar{w}, g\epsilon L_i\} = g\epsilon L_i$. It follows that $|\mathcal{V}(\mathcal{X})| \geq \frac{2L_i}{g\epsilon L_i} = \Omega(1/\epsilon)$, as claimed.

For Item (5), we observe that each subgraph $\mathcal{Y} \in \mathbb{X}_1$ is augmented in Step ii via an $\widetilde{\texttt{MST}}_i$ edges an in a star-like way. Thus, $\mathsf{Adm}(\mathcal{Y}^{\mathsf{aug}}) \leq \mathsf{Adm}(\mathcal{Y}) + 2(2L_i + 2\epsilon g L_i) + 2\bar{w} \leq \mathsf{Adm}(\mathcal{Y}) + 4L_i + 6\epsilon g L_i \leq 13L_i + 4L_i + 6g\epsilon L_i \leq 23L_i < 24L_i$ when $\epsilon \leq 1/g$. This complete the proof of Lemma 5.11. ∎

Acknowledgments. Hung Le is supported by a start up funding of University of Massachusetts at Amherst and the National Science Foundation under Grant No. CCF-2121952. Shay Solomon is partially supported by the Israel Science Foundation grant No.1991/19 and by Len Blavatnik and the Blavatnik Family foundation.

References

[ABLP89] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Compact distributed data structures for adaptive routing (extended abstract). In *STOC*, pages 479–489. ACM, 1989. 1

[ABP90] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. In *Proc. of 9th PODC*, pages 177–187, 1990. 1

[ABP92] B. Awerbuch, A. Baratz, and D. Peleg. Efficient broadcast and light-weight spanners. *Technical Report CS92-22, Weizmann Institute*, October, 1992. 1

[ADD+93] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Computational Geometry*, 9(1):81–100, 1993. 2, 3

[ADF+19] S. Alstrup, S. Dahlgaard, A. Filtser, M. Stöckel, and C. Wulff-Nilsen. Constructing light spanners deterministically in near-linear time. In *27th Annual European Symposium on Algorithms (ESA 2019)*, pages 4:1–4:15, 2019. 2, 3, 4, 5, 6

[Awe85] Baruch Awerbuch. Communication-time trade-offs in network synchronization. In *Proc. of 4th PODC*, pages 272–276, 1985. 1

[BKR+02] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat. Opus: an overlay peer utility service. In *Prof. of 5th OPENARCH*, 2002. 1

[BS03] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a (2k-1)-spanner of $o(n^{1+1/k})$ size in weighted graphs. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 384–296. Springer, 2003. 2

[BS07] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007. 3

[CDNS92] B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, 1992. 3

[Cha00] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000. 16

[CW16] S. Chechik and C. Wulff-Nilsen. Near-optimal light spanners. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'16, pages 883–892, 2016. 1, 2, 3, 4

[CW18] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Trans. Algorithms*, 14(3):33:1–33:15, 2018. preliminary version published in SODA 2016. 3

[DK02] Amin Vahdat Dejan Kostic. Latency versus cost optimizations in hierarchical overlay networks. *Technical report, Duke University*, (CS-2001-04), 2002. 1

[EN18] M. Elkin and O. Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Transactions on Algorithms*, 15(1), 2018. Announced at SODA '17. 2, 3, 4, 5, 6

[ENS14] Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. In *Proc. of 41th ICALP*, pages 442–452, 2014. 3

[ENS15] Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. *SIAM J. Discret. Math.*, 29(3):1312–1321, 2015. preliminary version published in ICALP 2014. 3

[Erd64] P. Erdős. Extremal problems in graph theory. *Theory of Graphs and Its Applications (Proc. Sympos. Smolenice)*, pages 29–36, 1964. 1

[ES16] M. Elkin and S. Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Transactions on Algorithms*, 12(3), 2016. 3, 5

[FS89] M. Fredman and M. Saks. The cell probe complexity of dynamic data structures. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, STOC'89. ACM Press, 1989. 5

[FS20] Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. *SIAM J. Comput.*, 49(2):429–447, 2020. preliminary version published in PODC 2016. 3, 4

[FW90] M. L. Fredman and D. E. Willard. BLASTING through the information theoretic barrier with FUSION TREES. In *Proceedings of the 22nd Annual ACM Symposium on Theory of computing*, STOC' 90, 1990. 13

[FW94] M. L. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences*, 48(3):533–551, 1994. Announced at FOCS'90. 10

[GT85] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2):209–221, 1985. 6, 10

[HZ96] S. Halperin and U. Zwick. Linear time deterministic algorithm for computing spanners for unweighted graphs, 1996. Manuscript. 1, 2, 5, 7, 20

[LS21] Hung Le and Shay Solomon. Towards a unified theory of light spanners I: Fast (yet optimal) constructions. *arXiv preprint arXiv:2106.15596*, 2021. `https://arxiv.org/abs/2106.15596`. 3, 4, 6, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28

[MPVX15] Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In Guy E. Blelloch and Kunal Agrawal, editors, *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 192–201. ACM, 2015. 2, 3, 4, 5, 6

[Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, PA, 2000. 1

[PS89] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. 2

[PT06] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proceedings of the 38th annual ACM Symposium on Theory of Computing*, STOC' 06. ACM Press, 2006. 13

[PU89a] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989. 1, 2

[PU89b] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989. 1

[RTZ05a] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *International Colloquium on Automata, Languages, and Programming*, pages 261–272. Springer, 2005. 2

[RTZ05b] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 261–272, 2005. 3

[RZ11] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. 3

[Tar75] Robert E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975. 3, 8

[Tar79] Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J. Comput. Syst. Sci.*, 18(2):110–127, 1979. 2

[TZ01a] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *Proc. of 33rd STOC*, pages 183–192, 2001. 3

[TZ01b] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. of 13th SPAA*, pages 1–10, 2001. 1

[VWF+03] Jürgen Vogel, Jörg Widmer, Dirk Farin, Martin Mauve, and Wolfgang Effelsberg. Priority-based distribution trees for application-level multicast. In *Proceedings of the 2nd Workshop on Network and System Support for Games, NETGAMES 2003, Redwood City, California, USA, May 22-23, 2003*, pages 148–157, 2003. 1

[WCT02] Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. Light graphs with small routing cost. *Networks*, 39(3):130–138, 2002. 1