Fast Constructions of Lightweight Spanners for General Graphs

MICHAEL ELKIN, Ben-Gurion University of the Negev
SHAY SOLOMON, Tel Aviv University

It is long known that for every *weighted* undirected n-vertex m-edge graph $G = (V, E, \omega)$, and every integer $k \geq 1$, there exists a $((2k-1)\cdot(1+\epsilon))$-spanner with $O(n^{1+1/k})$ edges and weight $O(k \cdot n^{1/k} \cdot \omega(MST(G))$, for an arbitrarily small constant $\epsilon > 0$. (Here $\omega(MST(G))$ stands for the weight of the minimum spanning tree of G.) To our knowledge, the only algorithms for constructing sparse and lightweight spanners for general graphs admit high running times. Most notable in this context is the greedy algorithm of Althöfer et al. [1993], analyzed by Chandra et al. [1992], which requires $O(m \cdot (n^{1+1/k} + n \cdot \log n))$ time.

In this article, we devise an efficient algorithm for constructing sparse and lightweight spanners. Specifically, our algorithm constructs $((2k-1)\cdot(1+\epsilon))$-spanners with $O(k \cdot n^{1+1/k})$ edges and weight $O(k \cdot n^{1/k}) \cdot \omega(MST(G))$, where $\epsilon > 0$ is an arbitrarily small constant. The running time of our algorithm is $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$. Moreover, by slightly increasing the running time we can reduce the other parameters. These results address an open problem by Roditty and Zwick [2004].

CCS Concepts: • **Theory of computation → Sparsification and spanners; Data structures design and analysis;**

Additional Key Words and Phrases: Graph spanners, light spanners, sparse graphs

ACM Reference Format:
Michael Elkin and Shay Solomon. 2016. Fast constructions of lightweight spanners for general graphs. ACM Trans. Algorithms 12, 3, Article 29 (April 2016), 21 pages.
DOI: http://dx.doi.org/10.1145/2836167

1. INTRODUCTION

1.1. Centralized Algorithms

Given an undirected weighted graph $G = (V, E, \omega)$ with a positive weight function ω over the edges and a parameter $t \geq 1$, a subgraph $H = (V, E')$ of G ($E' \subseteq E$) is called a *t-spanner* if for every edge $e = (u, v) \in E$, $dist_H(u, v) \leq t \cdot dist_G(u, v)$. (Here $dist_G(u, v)$ and $dist_H(u, v)$ stand for the distance between u and v in the graphs G and H, respectively.) Graph spanners were introduced in 1989 by Peleg and Schäffer [1989] and Peleg and Ullman [1989], who showed that for every unweighted n-vertex graph $G = (V, E)$ and an integer parameter $k \geq 1$, there exists an $O(k)$-spanner with $O(n^{1+1/k})$ edges. Althöfer et al. [1993] improved and generalized these results. They analyzed the natural greedy algorithm for constructing graph spanners, and showed that for every n-vertex *weighted* graph $G = (V, E, \omega)$ and an integer parameter $k \geq 1$, this algorithm constructs a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges. (This is near optimal [Peleg and Schäffer 1989].)

The work of the first-named author is supported by the BSF grant No. 2008430, by the ISF grant No. 87209011, and by the Lynn and William Frankel Center for Computer Sciences.
Authors' addresses: M. Elkin, Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel; email: elkinm@cs.bgu.ac.il; S. Solomon, School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; email: solo.shay@gmail.com.

DOI: http://dx.doi.org/10.1145/2836167

They also showed that the weight of the resulting spanner is $O(n/k) \cdot \omega(MST(G))$, where $\omega(MST(G))$ stands for the weight of the minimum spanning tree of G. (We will use the normalized notion of weight, called *lightness*, which is the ratio between the weight of the spanner and $\omega(MST(G))$.) In 1992, Chandra et al. [1992] improved the lightness bound, and showed that spanners obtained by the greedy algorithm have lightness $O(k \cdot n^{(1+\epsilon)/(k-1)} \cdot (1/\epsilon)^2)$, for an arbitrarily small $\epsilon > 0$. A somewhat stronger form of the trade-off of Chandra et al. [1992] is stretch $(2k-1)\cdot(1+\epsilon)$, $O(n^{1+1/k})$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$; we will henceforth refer to this form of the trade-off of Chandra et al. [1992] in our discussion. However, the running time of their algorithm is $O(m\cdot(n^{1+1/k}+n\cdot\log n))$, where m stands for $|E|$. Around the same time Awerbuch et al. [1992] devised an algorithm that constructs $O(k)$-spanners with $O(k \cdot n^{1+1/k} \cdot \Lambda)$ edges and lightness $O(k^2 \cdot n^{1/k} \cdot \Lambda)$, where Λ is the logarithm of the *aspect ratio* of the input graph, defined as the ratio between the maximum and minimum edge weights in the graph. The running time of the algorithm of Awerbuch et al. [1992] is $O(m \cdot k \cdot n^{1/k} \cdot \Lambda)$.

In the two decades that passed since the results of Peleg and Schäffer [1989], Peleg and Ullman [1989], Althöfer et al. [1993], Chandra et al. [1992], and Awerbuch et al. [1992] graph spanners turned out to be extremely useful. Among their applications is compact routing [Peleg and Ullman 1989; Peleg and Upfal 1989; Thorup and Zwick 2001b], distance oracles and labels [Peleg 1999; Thorup and Zwick 2001a; Roditty et al. 2005], network synchronization [Awerbuch 1985], and computing almost shortest paths [Cohen 1993; Roditty and Zwick 2004; Elkin 2005; Elkin and Zhang 2006; Feigenbaum et al. 2005]. Graph spanners also became a subject of intensive research for their own sake [Cohen 1993; Elkin and Peleg 2004; Elkin 2005; Baswana and Sen 2003; Thorup and Zwick 2006; Feigenbaum et al. 2005; Elkin and Zhang 2006; Woodruff 2006; Pettie 2009; Baswana et al. 2010].

In particular, a lot of research attention was devoted to devising efficient algorithms for constructing sparse spanners for weighted graphs. In 1993, Cohen [1993] devised a randomized algorithm for constructing $((2k-1)\cdot(1+\epsilon))$-spanners with $O(k \cdot n^{1+1/k} \cdot (1/\epsilon) \cdot \log n)$ edges. Her algorithm requires expected $O(m \cdot n^{1/k} \cdot k \cdot (1/\epsilon) \cdot \log n))$ time. Improving upon Cohen [1993], Baswana and Sen [2003] devised a randomized algorithm that constructs $(2k-1)$-spanners with expected $O(k \cdot n^{1+1/k})$ edges, within expected $O(k \cdot m)$ time. Roditty et al. [2005] derandomized this algorithm without any loss in parameters, or in running time. Roditty and Zwick [2004] devised a deterministic algorithm for constructing $(2k-1)$-spanners with $O(n^{1+1/k})$ edges in $O(k \cdot n^{2+1/k})$ time. In the discussion section at the end of their paper Roditty and Zwick [2004] write:

Another interesting property of the (original) greedy algorithm, shown by Chandra et al. [1992], is that the total weight of the edges in the $(2k-1)$-spanner that it constructs is at most $O(n^{(1+\epsilon)/k} \cdot \omega(MST(G)))$,[1] *for any $\epsilon > 0$. Unfortunately, this property no longer holds for the modified greedy algorithm. Again, it is an interesting open problem to obtain an efficient spanner construction algorithm that does have this property.*

In the current article, we devise such a construction. Specifically, our algorithm constructs $((2k-1)\cdot(1+\epsilon))$-spanners with $O(n^{1+1/k}\cdot(k+(1/\epsilon)^{2+1/k}))$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{2+1/k})$, and does so in time $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$, where $\alpha(n)$ is an inverse Ackermann function. In other words, the running time of our algorithm is near optimal and is drastically better than the running time $O(m \cdot (n^{1+1/k} + n \cdot \log n))$ of Chandra et al. [1992] and than that of Awerbuch et al. [1992] ($O(m \cdot k \cdot n^{1/k} \cdot \Lambda)$). We pay for this speed up by a small increase (by a factor of $k + (1/\epsilon)^{2+1/k}$) in the number of edges and a small increase (by a factor of $(1/\epsilon)$) in the lightness. (This comparison

[1]Actually, as mentioned previously, for stretch $2k-1$ the weight in Chandra et al. [1992] is $O(k \cdot n^{(1+\epsilon)/(k-1)} \cdot (1/\epsilon)^2 \cdot \omega(MST(G)))$.

Table I. A Concise Comparison of Previous and Our Constructions of Light Spanners. All the Constructions Mentioned in this Table are Deterministic. Our Results are Indicated by Bold Fonts. See also Corollary 2.6 and Theorem 3.7 for Slightly Different Dependencies on $1/\epsilon$ in Our Constructions

Results	Stretch	Number of edges	Lightness	Running time
Althöfer et al. [1993]	$2k-1$	$O(n^{1+1/k})$	$O(n/k)$	$O(m\cdot(n^{1+1/k}+n\cdot\log n))$
Chandra et al. [1992]	$(2k-1)\cdot(1+\epsilon)$	$O(n^{1+1/k})$	$O(k\cdot n^{1/k}\cdot(1/\epsilon)^{1+1/k})$	$O(m\cdot(n^{1+1/k}+n\cdot\log n))$
Awerbuch et al. [1992]	$O(k)$	$O(k\cdot n^{1+1/k}\cdot\Lambda)$	$O(k^2\cdot n^{1/k}\cdot\Lambda)$	$O(m\cdot k\cdot n^{1/k}\cdot\Lambda)$
Our faster construction	$\boldsymbol{(2k-1)\cdot(1+\epsilon)}$	$\boldsymbol{O(n^{1+1/k}\cdot(k+(1/\epsilon)^{2+1/k}))}$	$\boldsymbol{O(k\cdot n^{1/k}\cdot(1/\epsilon)^{2+1/k})}$	$\boldsymbol{O(k\cdot m+\min\{n\cdot\log n, m\cdot\alpha(n)\})}$
Our slower construction	$\boldsymbol{(2k-1)\cdot(1+\epsilon)}$	$\boldsymbol{O(n^{1+1/k}\cdot(1/\epsilon)^{2+1/k})}$	$\boldsymbol{O(k\cdot n^{1/k}\cdot(1/\epsilon)^{2+1/k})}$	$\boldsymbol{O(k\cdot n^{2+1/k})}$

is with Chandra et al. [1992]. Our algorithm strictly outperforms the algorithm of Awerbuch et al. [1992].) We also have another variant of our algorithm with a slightly higher running time ($O(k\cdot n^{2+1/k})$), and with $O(n^{1+1/k}\cdot(1/\epsilon)^{2+1/k})$ edges and lightness $O(k\cdot n^{1/k}\cdot(1/\epsilon)^{2+1/k})$.

Note that the relationship between the stretch and number of edges in both our results is essentially the same as in the state-of-the-art bound [Chandra et al. 1992]. (Though in the latter the number of edges does not depend on ϵ.) Specifically, the number of edges in our slower construction (that runs in $O(k\cdot n^{2+1/k})$ time) is, up to a factor of $(1/\epsilon)^{2+1/k}$, the same as in Chandra et al. [1992]. The faster variant of our algorithm (that runs in near-optimal time of $O(k\cdot m+\min\{n\cdot\log n, m\cdot\alpha(n)\})$, pays for the speedup by increasing the number of edges by a factor of k. We remark that the trade-off between the stretch and lightness in our result (and in the result of Chandra et al. [1992]) is almost optimal, that is, it almost matches the trade-off between the stretch and number of edges in the unweighted case; specifically, there is a slack of $1+\epsilon$ in the stretch, and a slack of $k\cdot(1/\epsilon)^{2+1/k}$ in the lightness. See Table I for a concise comparison of our and previous results on light spanners. (The lightness of all other spanner constructions [Cohen 1993; Baswana and Sen 2003; Roditty et al. 2005; Roditty and Zwick 2004] is unbounded.)

Chandra et al. [1992] also showed that the greedy algorithm gives rise to a construction of $O(\log^2 n)$-spanners with $O(n)$ edges and constant lightness, and to a construction of $O(\log n)$-spanners with $O(n)$ edges and lightness $O(\log n)$. The running time of these constructions is $O(m\cdot n\cdot\log n)$. Our algorithm also constructs spanners with the same (up to constant factors) parameters. The running time required by our algorithm to construct these spanners is $O(n^2\cdot\log n)$.

1.2. Streaming Algorithms

In the streaming model of computation the input graph $G=(V,E,\omega)$ arrives as a "stream," that is, the algorithm reads edges one after another. The algorithm is required to process edges efficiently, and to store only a limited amount of information. In the context of computing spanners the natural memory limitation is the size of the spanner. Multipass streaming algorithms also allow several (ideally, just a few) passes over the input stream.

The streaming model of computation was introduced by Alon et al. [1999] and by Feigenbaum et al. [2002]. The study of graph problems in the streaming model was introduced by Feigenbaum et al. [2005]. In particular, Feigenbaum et al. [2005] devised a randomized one-pass streaming algorithm for computing a $(2k+1)$-spanner with expected $O(k\cdot\log n\cdot n^{1+1/k})$ edges, using $O(k\cdot\log n\cdot n^{1/k})$ processing time per edge. This result was improved in Elkin [2011], who devised a randomized one-pass streaming algorithm that computes $(2k-1)$-spanners with expected $O(k\cdot n^{1+1/k})$ edges, using $O(1)$

processing time per edge. See also Baswana [2008] for another streaming algorithm for constructing spanners. Elkin and Zhang [2006] devised a multipass streaming algorithm for constructing sparse $(1+\epsilon,\beta)$-spanners. The number of passes in their algorithm is $O(\beta)$.

To our knowledge, there are currently no efficient streaming algorithms for computing *light* spanners. We show that our algorithm can be implemented efficiently in the streaming model *augmented with the sorting primitives* (henceforth, *augmented streaming model*). This model, introduced by Aggarwal et al. [2004] in 1994, allows one to have "sort passes." As a result of a sort pass, in consequent passes one can assume that the input stream that the algorithm reads is sorted. (See Aggarwal et al. [2004] for the justification of this model. The authors in Aggarwal et al. [2004] argue that "streaming computations with an added sorting primitive are a natural and efficiently implementable class of massive data set computations.")

The algorithm of Chandra et al. [1992] can be viewed as an algorithm in this model. After the initial sorting pass, it requires one pass over the input stream. As a result, it constructs a $((2k-1)\cdot(1+\epsilon))$-spanner with $O(n^{1+1/k})$ edges and lightness $O(k\cdot n^{1/k}\cdot(1/\epsilon)^{1+1/k})$, for an arbitrarily small $\epsilon>0$. The processing time per edge of this algorithm is, however, $O(n^{1+1/k}+n\cdot\log n)$, that is, prohibitively large.

We show that a variant of our algorithm computes $((2k-1)\cdot(1+\epsilon))$-spanners with expected $O(k\cdot n^{1+1/k}\cdot(1/\epsilon)^{1+1/k})$ edges and expected lightness $O(k^2\cdot n^{1/k}\cdot(1/\epsilon)^{1+1/k})$. It performs two passes over the input stream, which follow an initial sorting pass. In the first pass the worst-case (amortized, respectively) processing time per edge of our algorithm is $O(\frac{\log n}{\log\log n})$ ($O(\alpha(n))$, respectively). The processing time per edge of our algorithm in its second pass over the input stream is $O(1)$.

1.3. Our Techniques

Our algorithm is based on a transformation, which given a black-box construction of sparse (possibly heavy) spanners with a certain stretch t, efficiently produces sparse and *light* spanners with roughly the same stretch. We use this transformation in conjunction with a number of known algorithms that produce sparse spanners, but do not provide any bound on their lightness.

Our transformation generalizes a *metric transformation* from Chandra et al. [1992]. Specifically, the metric transformation of Chandra et al. [1992] converts constructions of sparse spanners for metrics into constructions of sparse and light spanners (for the same metric). The *generalized transformation* that we devise applies to weighted not necessarily complete graphs. (Observe that a metric can be viewed as a complete weighted graph.)

There are a number of technical difficulties that we overcome in our way to the generalized transformation. Next, we briefly discuss one of them. The construction of Chandra et al. [1992] hierarchically partitions the point set of the input metric into clusters. Then it selects a representative point from each cluster, and invokes its input black-box construction of sparse spanners on the metric induced by the representatives. One can try to mimic this approach in graphs by replacing each missing metric edge between representatives by the shortest path between them. This approach, however, is doomed to failure, as the overall number of edges taken into the spanner in this way might be too large. To overcome this difficulty we carefully select *representative edges* that are inserted into a certain auxiliary graph. Then the black-box input construction is applied to the auxiliary graph. As a result, we obtain a spanner of the auxiliary graph, which we call *auxiliary spanner*. This auxiliary spanner $\mathcal{Q}=(\mathcal{U},\mathcal{E})$ is a graph over a new vertex set $\mathcal{U}$, that is, $\mathcal{U}$ is not a subset of the original vertex set V. Next, we "project" the auxiliary spanner $\mathcal{Q}$ onto the original graph, that is, we translate edges of

$\mathcal{E}$ into edges of the original edge set E. This needs to be done carefully, to avoid blowing up the stretch and lightness. Also, it is crucial that this translation procedure will be efficient. Interestingly, we do not project vertices of $\mathcal{U}$ onto vertices of V, but rather edges of $\mathcal{E}$ onto edges of E. In particular, for a vertex $u \in \mathcal{U}$ and two edges $(u, x), (u, y) \in \mathcal{E}$, they may be translated into two vertex-disjoint edges $(u', x'), (u'', y') \in E$. As a result, a path in $\mathcal{Q}$ does not translate into a path in G, but rather into a collection of possibly vertex-disjoint edges. We show that these edges can be carefully glued into a path. This gluing, however, comes at a price of slightly increasing the stretch.

1.4. Related Work

The large body of work on constructing graph spanners efficiently was already discussed in Section 1.1. The problem of constructing light spanners efficiently was also studied in the context of *geometric spanners* (see Das and Narasimhan [1997] and Gudmundsson et al. [2002], and the references therein). In particular, light geometric spanners are closely related to approximation algorithms for the traveling salesman problem (see Arora [1998], Mitchell [1999], Rao and Smith [1998], and Bartal et al. [2012]).

1.5. Organization

In Section 2, we present and analyze our algorithm in the centralized model of computation. The algorithm is described in Section 2.1, and its analysis appears in Section 2.2. In Section 3, we present a few variants of our basic algorithm (from Section 2.1). In particular, the streaming variant of our algorithm is presented in Section 3.5.

1.6. Preliminaries

Consider an arbitrary weighted graph $G = (V, E, \omega)$, where $\omega(e)$ stands for the weight of edge $e \in E$. For a subset E' of edges from E, its weight $\omega(E')$ is defined as the sum of all edge weights in it, that is, $\omega(E') = \sum_{e \in E'} \omega(e)$. For a subgraph $H' = (V, E', \omega)$ of G ($E' \subseteq E$), its weight $\omega(H')$ is given by $\omega(H') = \omega(E') = \sum_{e \in E'} \omega(e)$; in particular, the weight of a path Π in G with edge set $E(\Pi)$ is $\omega(\Pi) = \omega(E(\Pi)) = \sum_{e \in \Pi} \omega(e)$. For a pair $u, v \in V$ of vertices, let $dist_G(u, v)$ denote the (weighted) *distance* between u and v in G, defined as the minimum weight of any path between u and v in G. Finally, the *aspect ratio* of the graph G is the ratio between the maximum and minimum edge weights in it.

We write $n = |V|, m = |E|$. We will also use a parameter ℓ, which will satisfy $\ell = O(\log n)$. This parameter reflects the number of different scales of edge weights, which are processed separately by our algorithm (see Section 2.1 for its precise definition).

We will use the following results as a black box.

THEOREM 1.1. *[Halperin and Zwick 1996] [unweighted graphs] For any unweighted graph $G = (V, E)$ and any integer $k \geq 1$, a $(2k - 1)$-spanner with $O(n^{1+1/k})$ edges can be built in $O(m)$ time.*

THEOREM 1.2. *[Baswana and Sen 2003; Roditty et al. 2005] [weighted graphs I] For any weighted graph $G = (V, E, \omega)$ and any integer $k \geq 1$, a $(2k - 1)$-spanner with $O(k \cdot n^{1+1/k})$ edges can be built in $O(k \cdot m)$ time.*

Remark: The algorithm of Baswana and Sen [2003] is randomized, but was later derandomized in Roditty et al. [2005]. Henceforth, the algorithm provided by Theorem 1.2 is deterministic.

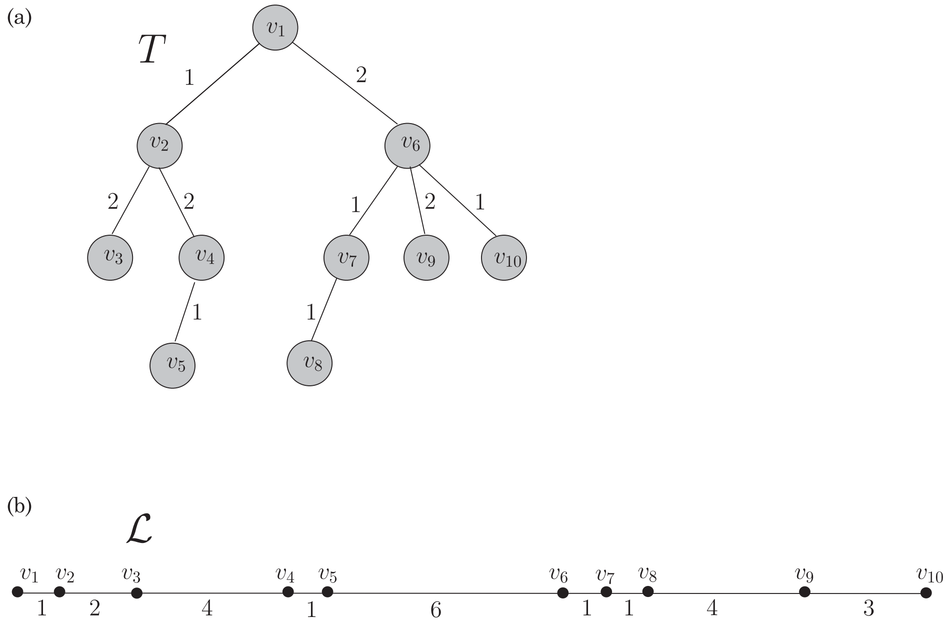
(a)
T
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10
1 2 2 2 1 2 1 1 1
(b)
ℒ
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10
1 2 4 1 6 1 1 4 3

Fig. 1. An illustration of (I) a tree T for G over $V = \{v_1, \ldots, v_{10}\}$ satisfying $\omega(T) = 13$, and (II) the corresponding Hamiltonian path $\mathcal{L}$, with $L = \omega(\mathcal{L}) = 23$.

THEOREM 1.3. *[Roditty and Zwick 2004] [weighted graphs II] For any weighted graph $G = (V, E, \omega)$ and any integer $k \geq 1$, a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges can be built in $O(k \cdot n^{2+1/k})$ time.*

THEOREM 1.4. *[Elkin 2011] [integer-weighted graphs] For any integer-weighted graph $G = (V, E, \omega)$ and any integer $k \geq 1$, a $(2k-1)$-spanner with expected $O(k \cdot n^{1+1/k})$ edges can be built in $O(SORT(m))$ time, where $SORT(m)$ is the time needed to sort m integers.*

Remark: The algorithm of Elkin [2011] is randomized: while the guarantees $2k-1$ and $O(SORT(m))$ on the stretch and running time, respectively, are deterministic, the guarantee $O(k \cdot n^{1+1/k})$ on the size of the spanner is in expectation. The fastest known randomized [Han and Thorup 2002] (deterministic [Han 2004], respectively) algorithm for sorting m integers requires expected $O(m \cdot \sqrt{\log\log n})$ (worst-case $O(m \cdot \log\log n)$, respectively) running time.

We will henceforth refer to the algorithms of Theorems 1.1, 1.2, 1.3, and 1.4 as Algorithms $UnwtdSp$, $WtdSp$, $WtdSp_2$, and $IntWtdSp$, respectively.

2. THE BASIC CONSTRUCTION

2.1. The Algorithm

In this section, we devise an algorithm *LightSp* that builds a light spanner efficiently.

Let $G = (V, E, \omega)$ be an arbitrary weighted graph, with $n = |V|, m = |E|$. Let $k \geq 1$ be an integer parameter that determines the stretch bound of the spanner.

We start with building an MST (or an $O(1)$-approximate MST) $T = (V, E_T)$ for G. Let $M_T = (V, dist_T)$ be the (shortest-path) metric induced by T. We then compute the Hamiltonian path $\mathcal{L} = (v_1, v_2, \ldots, v_n)$ of M_T drawn by taking the preorder traversal of T. For each $1 \leq i \leq n-1$, we set $dist_{\mathcal{L}}(v_i, v_{i+1}) = dist_T(v_i, v_{i+1})$. As a result, for any pair $u, v \in V$ of vertices, $dist_{\mathcal{L}}(u, v) \geq dist_T(u, v)$ (see Figure 1 for an illustration). Define

$L = \omega(\mathcal{L})$; it is well known (Corman et al. [2001], ch. 36) that $L \leq 2 \cdot \omega(T)$, and so $L = O(\omega(MST(G)))$.

Let $1 < \rho \leq 2$ be a parameter to be determined later, and define $\ell = \lceil \log_\rho n \rceil$. We partition the edges of E into $\ell + 1$ edge sets. The first edge set E_0 contains all edges with weight in the range $W_0 = (0, \frac{L}{n}]$. For each $1 \leq j \leq \ell$, the jth edge set E_j contains all edges with weight in the range $W_j = (\xi_j, \rho \cdot \xi_j]$, where $\xi_j = \rho^{j-1} \cdot \frac{L}{n}$.

Define $V_0 = V$, and let $n_0 = |V_0| = n$. We use Algorithm *WtdSp* to build a $(2k-1)$-spanner $H'_0 = (V_0, E'_0)$ for $G_0 = (V, E_0)$.

Algorithm *LightSp* proceeds in ℓ iterations $j = 1, 2, \ldots, \ell$.

(1) First, we divide the path $\mathcal{L}$ into $n_j = \frac{q \cdot L}{\xi_j} = \frac{q \cdot n}{\rho^{j-1}}$ intervals of length $\mu_j = \frac{\xi_j}{q}$ each, where $q > \frac{1}{2k-1}$ is a parameter to be determined later. (Note that the length of an interval refers to its weight rather than the number of edges in it. The path $\mathcal{L}$ is viewed as a one-dimensional interval of length L, which is then subdivided into shorter intervals.) These intervals induce a partition of V in the obvious[2] way; denote these intervals and the corresponding vertex sets by $I_j^{(1)}, I_j^{(2)}, \ldots, I_j^{(n_j)}$ and $V_j^{(1)}, V_j^{(2)}, \ldots, V_j^{(n_j)}$, respectively (see Figure 2 for an illustration). While computing this partition of V, we will store the index i of the vertex set $V_j^{(i)}$ to which any vertex $v \in V$ belongs in some variable $ind_j(v)$, for each $1 \leq i \leq n_j$. For each interval $I_j^{(i)}$, $1 \leq i \leq n_j$, we define a new "dummy" vertex $r_j^{(i)}$ (i.e., not present in G), which will serve as the *representative* of the interval $I_j^{(i)}$. The vertices $r_j^{(i)}$ (intervals $I_j^{(i)}$, respectively), $1 \leq i \leq n_j$, will also be referred to as the *j-level representatives* (*j-level intervals*, respectively). For any vertex $v \in V$, its j-level representative is given by $r_j(v) = r_j^{(ind_j(v))}$. We also define a dummy vertex set V_j, which contains all the j-level representatives. Observe that $|V_j| \leq \min\{n, n_j\} = \min\{n, \frac{q \cdot n}{\rho^{j-1}}\}$.

(2) We then compute an edge set $\tilde{E}_j$ over V_j as follows. First, we remove from the jth edge set E_j all edges that have both their endpoints in the same j-level interval to obtain the edge set $\bar{E}_j$. For each $e = (u, v) \in \bar{E}_j$, let $\hat{e} = (r_j(u), r_j(v))$ be a dummy edge of weight $\omega(e)$ between the corresponding j-level representatives, and note that $r_j(u) \neq r_j(v)$; we say that $\hat{e} = r(e)$ is the *representative edge* of e, and that $e = s(\hat{e})$ is the *source edge* of $\hat{e}$. Define $\hat{E}_j = \{\hat{e} = r(e) \mid e \in \bar{E}_j\}$, and let $\hat{G}_j = (V_j, \hat{E}_j)$ be the corresponding multigraph; note that $\hat{G}_j$ does not contain self-loops. Next, we transform $\hat{G}_j$ into a simple graph $\tilde{G}_j = (V_j, \tilde{E}_j)$ by removing all edges but the one of minimum weight, for every pair of incident vertices in $\hat{G}_j$. During this process, we store with every edge $\tilde{e} \in \tilde{E}_j$ its source edge $e = s(\tilde{e}) \in \bar{E}_j$.

(3) We proceed to building a $(2k-1)$-spanner $H'_j = (V_j, E'_j)$ for $\tilde{G}_j = (V_j, \tilde{E}_j)$ using Algorithm *WtdSp*.

(4) Next, we replace each edge $e' \in E'_j$ by its source edge $s(e')$. (Note that $e' \in E'_j \subseteq \tilde{E}_j$ is a dummy edge that connects some two distinct j-level intervals; denote them by I and I'. Moreover, its weight $\omega(e')$ is the smallest weight of an E_j-edge that connects I and I'. Hence, the source edge $s(e')$ is the minimum weight E_j-edge that connects the intervals I and I'.) Denote the resulting edge set by E_j^* and define $H_j^* = (V_j, E_j^*)$; note that $E_j^* \subseteq \bar{E}_j \subseteq E_j$. We will refer to H_j^* as the *j-level spanner*.

[2]A vertex that "lies" on the boundary of two consecutive intervals can be assigned to either one of them arbitrarily.

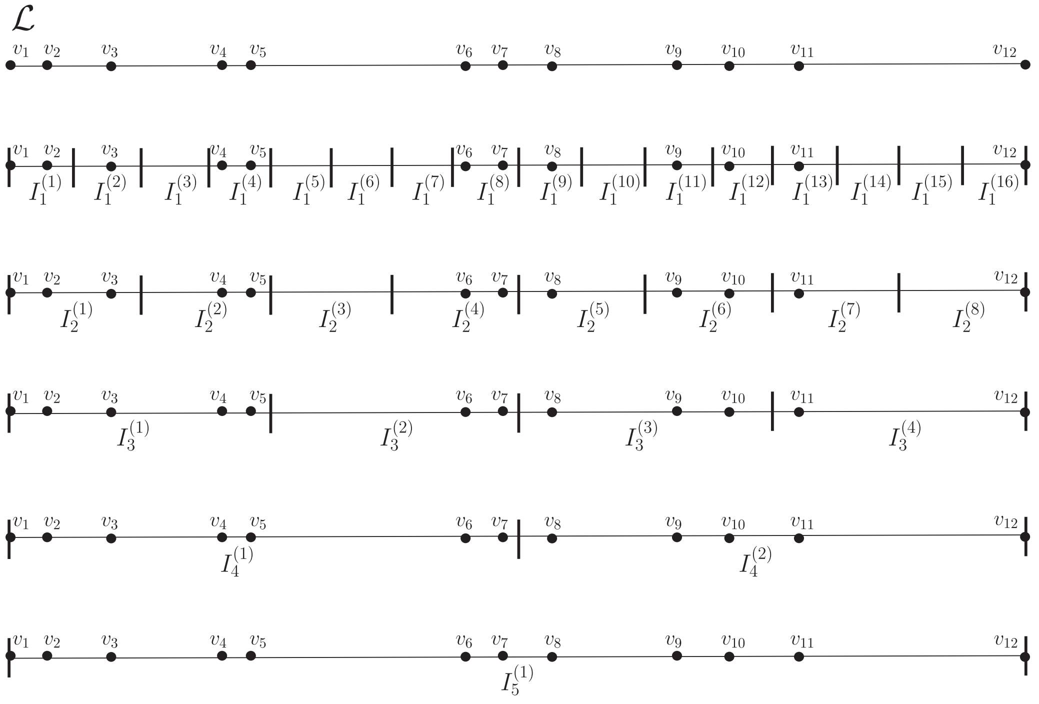
$\mathcal{L}$
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12}
$I_1^{(1)}$ $I_1^{(2)}$ $I_1^{(3)}$ $I_1^{(4)}$ $I_1^{(5)}$ $I_1^{(6)}$ $I_1^{(7)}$ $I_1^{(8)}$ $I_1^{(9)}$ $I_1^{(10)}$ $I_1^{(11)}$ $I_1^{(12)}$ $I_1^{(13)}$ $I_1^{(14)}$ $I_1^{(15)}$ $I_1^{(16)}$
$I_2^{(1)}$ $I_2^{(2)}$ $I_2^{(3)}$ $I_2^{(4)}$ $I_2^{(5)}$ $I_2^{(6)}$ $I_2^{(7)}$ $I_2^{(8)}$
$I_3^{(1)}$ $I_3^{(2)}$ $I_3^{(3)}$ $I_3^{(4)}$
$I_4^{(1)}$ $I_4^{(2)}$
$I_5^{(1)}$

Fig. 2. An illustration of the Hamiltonian path $\mathcal{L} = (v_1, v_2, \ldots, v_{12})$ and the corresponding hierarchical partition, for the case $\rho = 2, q = 4/3$. Here $\ell = \lceil \log_2 12 \rceil = 4$, and $n_1 = q \cdot n = 16$. Letting $\mathcal{V}_j$ denote the j-level partition of V, $1 \le j \le 4$, we have $\mathcal{V}_1 = \{\{v_1, v_2\}, \{v_3\}, \{v_4, v_5\}, \{v_6, v_7\}, \{v_8\}, \{v_9\}, \{v_{10}\}, \{v_{11}\}, \{v_{12}\}\}$; $\mathcal{V}_2 = \{\{v_1, v_2, v_3\}, \{v_4, v_5\}, \{v_6, v_7\}, \{v_8\}, \{v_9, v_{10}\}, \{v_{11}\}, \{v_{12}\}\}$; $\mathcal{V}_3 = \{\{v_1, v_2, \ldots, v_5\}, \{v_6, v_7\}, \{v_8, v_9, v_{10}\}, \{v_{11}, v_{12}\}\}$; $\mathcal{V}_4 = \{\{v_1, v_2, \ldots, v_7\}, \{v_8, v_9, \ldots, v_{12}\}\}$. As $\ell = 4$, the five-level partition $\mathcal{V}_5 = \{\{v_1, v_2, \ldots, v_{12}\}\}$ is not considered by the algorithm.

(5) We then compute an edge set $\tilde{E}_j$ over V_j as follows. First, we remove from the jth edge set E_j all edges that have both their endpoints in the same j-level interval to obtain the edge set $\bar{E}_j$. For each $e = (u, v) \in \bar{E}_j$, let $\hat{e} = (r_j(u), r_j(v))$ be a dummy edge of weight $\omega(e)$ between the corresponding j-level representatives, and note that $r_j(u) \neq r_j(v)$; we say that $\hat{e} = r(e)$ is the *representative edge* of e, and that $e = s(\hat{e})$ is the *source edge* of $\hat{e}$. Define $\hat{E}_j = \{\hat{e} = r(e) \mid e \in \bar{E}_j\}$, and let $\hat{G}_j = (V_j, \hat{E}_j)$ be the corresponding multigraph; note that $\hat{G}_j$ does not contain self-loops. Next, we transform $\hat{G}_j$ into a simple graph $\tilde{G}_j = (V_j, \tilde{E}_j)$ by removing all edges but the one of minimum weight, for every pair of incident vertices in $\hat{G}_j$. During this process, we store with every edge $\tilde{e} \in \tilde{E}_j$ its source edge $e = s(\tilde{e}) \in \bar{E}_j$.

(6) We proceed to building a $(2k-1)$-spanner $H'_j = (V_j, E'_j)$ for $\tilde{G}_j = (V_j, \tilde{E}_j)$ using Algorithm *WtdSp*.

(7) Next, we replace each edge $e' \in E'_j$ by its source edge $s(e')$. (Note that $e' \in E'_j \subseteq \tilde{E}_j$ is a dummy edge that connects some two distinct j-level intervals; denote them by I and I'. Moreover, its weight $\omega(e')$ is the smallest weight of an E_j-edge that connects I and I'. Hence, the source edge $s(e')$ is the minimum weight E_j-edge that connects the intervals I and I'.) Denote the resulting edge set by E^*_j and define $H^*_j = (V_j, E^*_j)$; note that $E^*_j \subseteq \bar{E}_j \subseteq E_j$. We will refer to H^*_j as the *j-level spanner*.

Finally, we define $E^* = E_T \cup E'_0 \cup \bigcup_{j=1}^{\ell} E^*_j$, and return the graph $H^* = (V, E^*)$ as our spanner.

2.2. The Analysis

In this section, we analyze the properties of the constructed graph H^*.

2.2.1. Stretch. We start with analyzing the stretch of the graph $H^* = (V, E^*)$.

Define $str(k, q) = (2k-1)\cdot(1+\frac{1}{q})+\frac{1}{q}$. Next, we show that the stretch of H^* is at most $str(k, q)$.

In other words, we prove that $dist_{H^*}(u, v) \le str(k, q) \cdot dist_G(u, v)$, for any edge $e = (u, v) \in E$.

If $e \in E_0$, then we have $dist_{H^*}(u, v) \le dist_{H'_0}(u, v) \le (2k-1) \cdot dist_G(u, v) \le str(k, q) \cdot dist_G(u, v)$.

We henceforth assume that $e \in E_j$, for some $1 \le j \le \ell$. The next lemma shows that $dist_{H^*}(u, v) \le str(k, q) \cdot \omega(e)$. (This lemma in conjunction with the triangle inequality imply that $dist_{H^*}(u, v) \le str(k, q) \cdot dist_G(u, v)$, for any edge $e = (u, v) \in E$, which provides the required stretch bound on H^*.)

LEMMA 2.1. *For any edge $e = (u, v) \in E_j$, $1 \le j \le \ell$, there is a path of length at most $str(k, q) \cdot \omega(e)$ in $H^*_j \cup T = (V, E^*_j \cup E_T)$ between u and v.*

PROOF. Since e is in E_j, its weight is in $W_j = (\xi_j, \rho \cdot \xi_j]$. In particular, we have $\omega(e) > \xi_j$.

For any pair $x, y \in V$ of vertices, let $\Pi_T(x, y)$ denote the path in the MST T between them.

Suppose first that u and v belong to the same j-level interval. Note that the length of all j-level intervals is $\mu_j = \frac{\xi_j}{q}$, and so $dist_T(u, v) \le dist_{\mathcal{L}}(u, v) \le \frac{\xi_j}{q}$. Hence, the weight of the path $\Pi_T(u, v)$ between u and v in T is at most

$$\frac{\xi_j}{q} < \frac{\omega(e)}{q} < str(k, q) \cdot \omega(e). \tag{1}$$

Thus, $\Pi_T(u, v)$ is a path of the required length in $H^*_j \cup T$ between u and v.

We henceforth assume that u and v belong to distinct j-level intervals. Let $\tilde{e} = (r_j(u), r_j(v)) \in \tilde{E}_j$ be the respective edge in $\tilde{E}_j$, where $r_j(u)$ and $r_j(v)$ are the j-level representatives of u and v, respectively. By the construction of $\tilde{E}_j$, we have $\omega(\tilde{e}) \le \omega(e)$. Let $\Pi'(r_j(u), r_j(v)) = (r_j(u) = u'_0, u'_1, \ldots, u'_h = r_j(v))$ be a path of length at most $(2k-1) \cdot dist_{\tilde{G}_j}(r_j(u), r_j(v))$ in H'_j between $r_j(u)$ and $r_j(v)$. We have

$$\omega(\Pi'(r_j(u), r_j(v))) \le (2k-1) \cdot dist_{\tilde{G}_j}(r_j(u), r_j(v)) \le (2k-1) \cdot \omega(\tilde{e}) \le (2k-1) \cdot \omega(e). \tag{2}$$

Notice that the edges of H'_j (and of $\Pi'(r_j(u), r_j(v))$) are not taken as is to the graph H^*_j. That is, each edge $e'_i = (u'_i, u'_{i+1}) \in H'_j$ is replaced by its source edge $s(e'_i) = (u^{(i)}_i, u^{(i)}_{i+1}) \in E^*_j$, for $i \in [0, h-1]$. However, while any two consecutive edges $e'_i = (u'_i, u'_{i+1})$ and $e'_{i+1} = (u'_{i+1}, u'_{i+2})$ along the original path $\Pi'(r_j(u), r_j(v))$ share a common endpoint, namely, u'_{i+1}, the respective source edges $s(e'_i) = (u^{(i)}_i, u^{(i)}_{i+1})$ and $s(e'_{i+1}) = (u^{(i+1)}_{i+1}, u^{(i+1)}_{i+2})$ may be vertex disjoint; in particular, it may happen that $u^{(i)}_{i+1} \ne u^{(i+1)}_{i+1}$. To connect the two endpoints $u^{(i)}_{i+1}$ and $u^{(i+1)}_{i+1}$ of these source edges in $H^*_j \cup T$, we take the path $\Pi_T(u^{(i)}_{i+1}, u^{(i+1)}_{i+1})$ in T between them. More specifically, for any edge $e'_i = (u'_i, u'_{i+1})$ of the

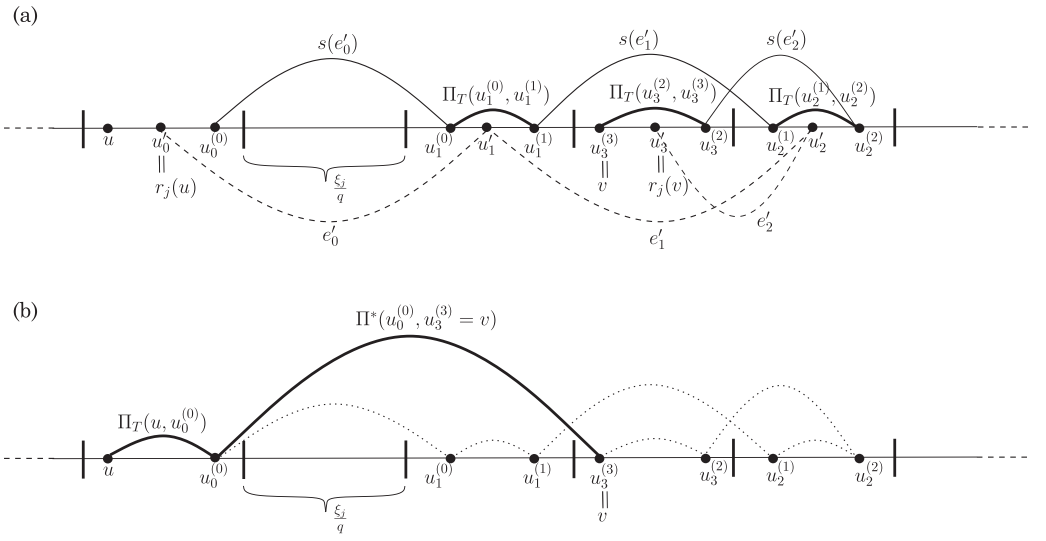

Fig. 3. (a) An illustration of the path $\Pi^*(u_0^{(0)}, u_h^{(h)} = v)$, for $h = 3$. Edges $e'_i = (u'_i, u'_{i+1})$ of the path $\Pi'(r_j(u), r_j(v))$ in H'_j are depicted by dashed lines. The MST-paths $\Pi_T(u_i^{(i-1)}, u_i^{(i)}) \in T$ between vertices $u_i^{(i-1)}$ and $u_i^{(i)}$ that belong to the same interval are depicted by thick solid lines. All interval lengths are equal to $\frac{\xi_j}{q}$. All the other edges (i.e., edges $s(e'_i)$ in H_j^*) are depicted by thin solid lines. (b) An illustration of the path $\Pi^*(u, v)$, obtained as the concatenation of the path $\Pi_T(u, u_0^{(0)}) \in T$ and the path $\Pi^*(u_0^{(0)}, u_3^{(3)} = v)$), both depicted by thick solid lines. The dotted lines in the figure represent different subpaths of the path $\Pi^*(u_0^{(0)}, u_3^{(3)} = v)$.

original path $\Pi'(r_j(u), r_j(v)) = (r_j(u) = u'_0, u'_1, \ldots, u'_h = r_j(v))$, $i \in [0, h-1]$, let

$$\Pi\big(u_i^{(i)}, u_{i+1}^{(i+1)}\big) \;=\; s\big(e'_i\big) \circ \Pi_T\big(u_{i+1}^{(i)}, u_{i+1}^{(i+1)}\big)$$

be the path in $H_j^* \cup T$ between $u_i^{(i)}$ and $u_{i+1}^{(i+1)}$ obtained from the concatenation of the source edge $s(e'_i) = (u_i^{(i)}, u_{i+1}^{(i)}) \in E_j^*$ and the path $\Pi_T(u_{i+1}^{(i)}, u_{i+1}^{(i+1)})$; we define $u_h^{(h)} = v$. Observe that $u_{i+1}^{(i)}$ and $u_{i+1}^{(i+1)}$ belong to the same j-level interval, for any $i \in [0, h-1]$, which implies that $\omega(\Pi_T(u_{i+1}^{(i)}, u_{i+1}^{(i+1)})) \le \frac{\xi_j}{q}$. Also, since $s(e'_i) \in E_j^* \subseteq E_j$, it holds that $\omega(e'_i) > \xi_j$. Recall that the weight of the source edge $s(e'_i)$ of e'_i is equal to that of $e'_i = (u'_i, u'_{i+1})$, and so $\omega(s(e'_i)) = \omega(e'_i) > \xi_j$. Consequently, we have

$$\omega\big(\big(\Pi(u_i^{(i)}, u_{i+1}^{(i+1)}\big)\big) = \omega(s(e'_i)) + \omega\big(\Pi_T\big(u_{i+1}^{(i)}, u_{i+1}^{(i+1)}\big)\big) \le \omega(e'_i) + \frac{\xi_j}{q} \le \omega(e'_i) \cdot \left(1 + \frac{1}{q}\right). \tag{3}$$

Let $\Pi^*(u_0^{(0)}, u_h^{(h)} = v)$ be the path in $H_j^* \cup T$ between $u_0^{(0)}$ and v obtained from $\Pi'(r_j(u), r_j(v))$ by replacing each edge $e'_i = (u'_i, u'_{i+1})$ in it by the path $\Pi(u_i^{(i)}, u_{i+1}^{(i+1)})$. ($u_0^{(0)}$ is the endpoint of the source edge $s(e'_0) = (u_0^{(0)}, u_1^{(0)})$, which belongs to u's interval; see Figure 3(a) for an illustration.) Equations (2) and (3) yield

$$\omega\big(\Pi^*\big(u_0^{(0)}, u_h^{(h)} = v\big)\big) \le \omega(\Pi'(r_j(u), r_j(v))) \cdot \left(1 + \frac{1}{q}\right) \le (2k-1) \cdot \omega(e) \cdot \left(1 + \frac{1}{q}\right).$$

Finally, let $\Pi^*(u,v) = \Pi_T(u, u_0^{(0)}) \circ \Pi^*(u_0^{(0)}, u_h^{(h)} = v)$ be the path in $H_j^* \cup T$ between u and v obtained from the concatenation of the path $\Pi_T(u, u_0^{(0)})$ in T between u and $u_0^{(0)}$ and the path $\Pi^*(u_0^{(0)}, u_h^{(h)} = v)$ (see Figure 3(II) for an illustration). Since u and $u_0^{(0)}$ belong to the same j-level interval, it holds that $\omega(\Pi_T(u, u_0^{(0)})) \le \frac{\xi_j}{q}$. Recall that $\omega(e) > \xi_j$. We conclude that

$$\begin{aligned}\omega(\Pi^*(u,v)) &= \omega\big(\Pi_T\big(u,u_0^{(0)}\big)\big) + \omega\big(\Pi^*(u_0^{(0)}, u_h^{(h)} = v)\big) \le \frac{\xi_j}{q} + (2k-1)\cdot\omega(e)\cdot\left(1+\frac{1}{q}\right)\\ &\le \omega(e)\cdot\left((2k-1)\cdot\left(1+\frac{1}{q}\right)+\frac{1}{q}\right) = str(k,q)\cdot\omega(e).\end{aligned}$$

Hence, $\Pi^*(u,v)$ is a path of length at most $str(k,q)\cdot\omega(e)$ in $H_j^* \cup T$ between u and v, as required. □

For the rest of our analysis, define $\varphi = \max\{0, \log_\rho q\}$. Note that $\varphi = 0$ for $q \le 1$, while $\varphi > 0$ for $q > 1$.

2.2.2. Number of Edges. The next lemma bounds the number of edges in the spanner $H^* = (V, E^*)$.

LEMMA 2.2. $|E^*| = O(k \cdot n^{1+1/k} \cdot (1 + \frac{q^{1+1/k}}{\rho^{1+1/k}-1}))$.

PROOF. Recall that $|V_j| \le \min\{n, n_j\} = \min\{n, \frac{q\cdot n}{\rho^{j-1}}\} \le n$, for $1 \le j \le \ell$. By construction, $E^* = E_T \cup E'_0 \cup \bigcup_{j=1}^{\ell} E_j^*$. Clearly, $|E_T| = n-1$. By Theorem 1.2, $|E'_0| = O(k\cdot|V_0|^{1+1/k}) = O(k\cdot n^{1+1/k})$. Also, it holds that $|E_j^*| = |E'_j| = O(k\cdot|V_j|^{1+1/k}) = O(k\cdot n^{1+1/k})$, for each $1 \le j \le \ell$.

Next, recall that $\varphi = \max\{0, \log_\rho q\}$, and consider an index $\varphi + 1 \le j \le \ell$. Observe that $\frac{q}{\rho^{j-1}} \le 1$. Hence, $n \ge \frac{q\cdot n}{\rho^{j-1}} = n_j$, and so $|V_j| \le n_j = \frac{q\cdot n}{\rho^{j-1}}$. It follows that

$$|E_j^*| = O\left(k\cdot\left(\frac{q\cdot n}{\rho^{j-1}}\right)^{1+1/k}\right) = O\left(k\cdot q^{1+1/k}\cdot n^{1+1/k}\cdot\left(\frac{1}{\rho^{1+1/k}}\right)^{j-1}\right). \tag{4}$$

Since $\rho > 1$, we have

$$\sum_{\varphi+1\le j\le\ell}\left(\frac{1}{\rho^{1+1/k}}\right)^{j-1} \le \sum_{i=0}^{\infty}\left(\frac{1}{\rho^{1+1/k}}\right)^{i} = \frac{1}{\rho^{1+1/k}-1}+1.$$

Observe that $|\{j \mid 1 \le j < \varphi+1\}| = \lceil\varphi\rceil$. It follows that

$$\begin{aligned}|E^*| &= |E_T| + |E'_0| + \sum_{j=1}^{\ell}|E_j^*| = n - 1 + O(k\cdot n^{1+1/k}) + \sum_{1\le j<\varphi+1}|E_j^*| + \sum_{\varphi+1\le j\le\ell}|E_j^*|\\ &= O(k\cdot n^{1+1/k}) + \sum_{1\le j<\varphi+1}O(k\cdot n^{1+1/k}) + O(k\cdot q^{1+1/k}\cdot n^{1+1/k})\cdot\sum_{\varphi+1\le j\le\ell}\left(\frac{1}{\rho^{1+1/k}}\right)^{j-1}\\ &= O\left(k\cdot n^{1+1/k}\cdot\left(1+\lceil\varphi\rceil+\frac{q^{1+1/k}}{\rho^{1+1/k}-1}\right)\right).\end{aligned} \tag{5}$$

Next, we argue that

$$\lceil\varphi\rceil = O\left(\frac{q^{1+1/k}}{\rho^{1+1/k}-1}\right). \tag{6}$$

This assertion is immediate if $\varphi = 0$. We henceforth assume that $\varphi > 0$. In this case $q > 1$, and we have $\lceil\varphi\rceil = \lceil\log_\rho q\rceil \le \log_\rho q + 1$. To prove Equation (6), it suffices to show that $\log_\rho q \le \frac{3}{\ln 2}(\frac{q^{1+1/k}}{\rho^{1+1/k}-1})$. Substituting $\log_\rho q = \frac{\ln q}{\ln \rho}$ and rearranging, we need to show that

$$\frac{\rho^{1+1/k}-1}{\ln\rho} \le \frac{3}{\ln 2}\left(\frac{q^{1+1/k}}{\ln q}\right). \tag{7}$$

We proceed by substituting $x = \ln\rho$. Since $1 < \rho \le 2$, the parameter x takes values in the range $(0, \ln 2]$. Moreover, for any x in this range, it holds that $e^x \le (\frac{1}{\ln 2})x + 1$. It follows that

$$\frac{\rho^{1+1/k}-1}{\ln\rho} = \frac{(e^x)^{1+1/k}-1}{x} \le \frac{(e^x)^2-1}{x} \le \frac{((\frac{1}{\ln 2})x+1)^2-1}{x} \le \left(\frac{1}{\ln 2}\right)^2 x + \frac{2}{\ln 2} \le \frac{3}{\ln 2}.$$

Noting that $q^{1+1/k} > \ln q$, we conclude that $\frac{\rho^{1+1/k}-1}{\ln\rho} \le \frac{3}{\ln 2} \le \frac{3}{\ln 2}(\frac{q^{1+1/k}}{\ln q})$. This shows that Equation (7) holds, which, in turn, proves the validity of Equation (6). Finally, Equations (5) and (6) imply that

$$|E^*| = O\left(k\cdot n^{1+1/k}\cdot\left(1+\lceil\varphi\rceil+\frac{q^{1+1/k}}{\rho^{1+1/k}-1}\right)\right) = O\left(k\cdot n^{1+1/k}\cdot\left(1+\frac{q^{1+1/k}}{\rho^{1+1/k}-1}\right)\right). \quad \square$$

2.2.3. Weight. The next lemma estimates the weight of $H^* = (V, E^*)$.

LEMMA 2.3. $\omega(E^*) = O\left(k^2\cdot n^{1/k}\cdot\frac{q^{1+1/k}}{\rho-1}\right)\cdot\omega(MST(G))$.

PROOF. First, observe that the edge weights in E_0 are bounded above by $\frac{L}{n}$. Since $|E'_0| = O(k\cdot n^{1+1/k})$, it follows that

$$\omega(E'_0) \le O(k\cdot n^{1+1/k})\cdot\frac{L}{n} = O(k\cdot n^{1/k}\cdot L). \tag{8}$$

The edge weights in E_j are bounded above by $\xi_j\cdot\rho = \rho^j\cdot\frac{L}{n}$, for each $1 \le j \le \ell$. Consider first indices j in the range $1 \le j < \varphi+1$. If such an index exists, we must have $\varphi = \log_\rho q > 0$. Since $|E^*_j| = O(k\cdot n^{1+1/k})$, we have $\omega(E^*_j) = O(k\cdot n^{1+1/k})\cdot(\rho^j\cdot\frac{L}{n}) = O(k\cdot n^{1/k}\cdot\rho^j\cdot L)$. It follows that

$$\sum_{1\le j<\varphi+1}\omega(E^*_j) = \sum_{1\le j<\log_\rho q+1}\omega(E^*_j) = O(k\cdot n^{1/k}\cdot L)\cdot\sum_{1\le j<\log_\rho q+1}\rho^j = O(k\cdot n^{1/k}\cdot L\cdot q). \tag{9}$$

If no such index exists, then $\varphi = 0$ and $q < 1$. In this case, the sum $\sum_{1\le j<\varphi+1}\omega(E^*_j)$ is obviously zero. However, since Equation (9) holds here as well, we will use it to avoid a redundant case analysis.

Next, consider indices j with $\varphi+1 \le j \le \ell$. By Equation (4), $|E^*_j| = O(k\cdot q^{1+1/k}\cdot n^{1+1/k}\cdot(\frac{1}{\rho^{1+1/k}})^{j-1})$. Hence,

$$\begin{aligned}\omega(E^*_j) &= O\left(k\cdot q^{1+1/k}\cdot n^{1+1/k}\cdot\left(\frac{1}{\rho^{1+1/k}}\right)^{j-1}\right)\cdot\left(\rho^j\cdot\frac{L}{n}\right)\\ &= O\left(k\cdot q^{1+1/k}\cdot n^{1/k}\cdot\left(\frac{1}{\rho^{1/k}}\right)^{j-1}\cdot L\right).\end{aligned}$$

Observe that

$$\sum_{\varphi+1 \le j \le \ell} \left(\frac{1}{\rho^{1/k}}\right)^{j-1} \le \sum_{i=0}^{\infty} \left(\frac{1}{\rho^{1/k}}\right)^{i} = O\left(\frac{k}{\rho-1}\right).$$

It follows that

$$\begin{aligned} \sum_{\varphi+1 \le j \le \ell} \omega(E_j^*) &= O(k \cdot q^{1+1/k} \cdot n^{1/k} \cdot L) \cdot \sum_{\varphi+1 \le j \le \ell} \left(\frac{1}{\rho^{1/k}}\right)^{j-1} \\ &= O\left(k^2 \cdot q^{1+1/k} \cdot n^{1/k} \cdot L \cdot \frac{1}{\rho-1}\right). \end{aligned} \tag{10}$$

By Equations (8), (9), and (10),

$$\begin{aligned} \omega(E^*) &= \omega(T) + \omega(E_0') + \sum_{1 \le j < \varphi+1} \omega(E_j^*) + \sum_{\varphi+1 \le j \le \ell} \omega(E_j^*) \\ &= \omega(T) + O(k \cdot n^{1/k} \cdot L) + O(k \cdot n^{1/k} \cdot L \cdot q) + O\left(k^2 \cdot q^{1+1/k} \cdot n^{1/k} \cdot L \cdot \frac{1}{\rho-1}\right) \\ &= O\left(k^2 \cdot n^{1/k} \cdot \frac{q^{1+1/k}}{\rho-1}\right) \cdot \omega(MST(G)). \end{aligned}$$

(Since $q > \frac{1}{2k-1}$, we have $k^2 \cdot q^{1+1/k} = \Omega(k)$.) □

2.2.4. Running Time. The next lemma bounds the running time of our construction.

LEMMA 2.4. *The graph H^* can be built within time $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$.*

PROOF. The tree T can be built in time $O(\min\{m + n \cdot \log n, m \cdot \alpha(m, n)\})$ [Chazelle 2000]. Computing the Hamiltonian path $\mathcal{L}$ of M_T, as well as its weight $L = \omega(\mathcal{L})$, takes another $O(n)$ time. Having computed L, we can partition the edges of E to the $\ell + 1$ edge sets $E_0, E_1, \ldots, E_\ell$ in $O(m)$ time in the obvious way.

By Theorem 1.2, building the spanner H_0' for $G_0 = (V, E_0)$ requires $O(k \cdot |E_0|) = O(k \cdot m)$ time.

Next, we bound the running time of a single iteration j in the main loop, for $1 \le j \le \ell$.

(1) For each $1 \le j \le \ell$, we allocate an array A_j of size n_j. For each $x \in V_j$, the entry $A_j[x]$ will contain the linked list of x's neighbors in the multigraph $\hat{G}_j$ (with the weights of the representative edges).

For each edge $e = (u, v) \in E$ we determine the index $j \in [\ell]$ such that $\omega(e) \in W_j$ (and so $e \in E_j$). By the locations of u and v on the path $\mathcal{L}$, we determine the j-level representatives $r_j(u)$ and $r_j(v)$ of u and v, respectively. We also determine whether the edge crosses between different j-level intervals. If this is the case (i.e., $e \in \bar{E}_j$), then we insert the edge into the array A_j. (In other words, we update the linked lists of both $r_j(u)$ and $r_j(v)$ in A_j.) In this way we form the multigraphs $\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_\ell$ within $O(|E|) = O(m)$ time. To prune these multigraphs into simple graphs $\tilde{G}_1, \tilde{G}_2, \ldots, \tilde{G}_\ell$, we need to remove all edges but the one of minimum weight, for every pair of incident vertices in $\hat{G}_j$. This can be done deterministically within $O(m)$ time and space.

(2) By Theorem 1.2, the time needed to build the $(2k-1)$-spanner $H_j' = (V_j, E_j')$ for $\tilde{G}_j = (V, \tilde{E}_j)$ is $O(k \cdot |\tilde{E}_j|) = O(k \cdot |E_j|)$. Summing over all ℓ iterations, the overall time is $\sum_{j=1}^{\ell} O(k \cdot |E_j|) = O(k \cdot m)$.

(3) Replacing each edge of E'_j by its source edge takes $O(1)$ time, thus the graph $H^*_j = (V, E^*_j)$ is built in $O(|E'_j|) = O(|E_j|)$ time. Summing over all ℓ iterations, the overall time is $\sum_{j=1}^{\ell} O(|E_j|) = O(m)$.

It follows that the total running time of the construction is
$O(\min\{m + n \cdot \log n, m \cdot \alpha(m, n)\}) + O(k \cdot m) \; = \; O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$. □

We remark that there is a randomized algorithm for constructing MST within $O(m+n)$ time [Karger et al. 1995], where the time bound holds with high probability. Employing it reduces the running time of our algorithm to be (with high probability) just $O(k \cdot m)$.

2.2.5. Summary. We summarize the properties of the spanner $H^* = (V, E^*)$ in the following theorem. (We substituted $\rho = 2$ to optimize the parameters of the construction.)

THEOREM 2.5. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|, m = |E|$. For any integer $k \geq 1$ and any number $q > \frac{1}{2k-1}$, a $((2k-1) \cdot (1 + \frac{1}{q}) + \frac{1}{q})$-spanner with $O(k \cdot n^{1+1/k} \cdot (1 + q^{1+1/k}))$ edges and lightness $O(k^2 \cdot n^{1/k} \cdot q^{1+1/k})$, can be built in $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$ time.*

By substituting $q = \Theta(1/\epsilon)$ in Theorem 2.5, for some small $\epsilon > 0$, we obtain:

COROLLARY 2.6. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|, m = |E|$. For any integer $k \geq 1$ and any small $\epsilon > 0$, a $((2k-1) \cdot (1+\epsilon))$-spanner with $O(k \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$ edges and lightness $O(k^2 \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$, can be built in $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$ time.*

Note also that by substituting $q = \Theta(k/\epsilon)$, we get a $(2k - 1 + \epsilon)$-spanner with $O(k^2 \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$ edges and lightness $O(k^3 \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$, within the same time.

3. VARIANTS OF THE CONSTRUCTION

In this section, we devise some variants of the basic construction of Section 2 by applying a small (but significant) modification to Algorithm *LightSp*.

Notice that Algorithm *LightSp* employs Algorithm *WtdSp* as a black box for building (i) the spanner $H'_0 = (V_0, E'_0)$ for the graph $G_0 = (V, E_0)$, and (ii) the spanner $H'_j = (V_j, E'_j)$ for the graph $\tilde{G}_j = (V_j, \tilde{E}_j)$ in step 3 of the main loop, for each $1 \leq j \leq \ell$. Instead of employing Algorithm *WtdSp*, we can employ some of the other black-box spanners that are summarized in Section 1.6.

3.1. First Variant (Slightly Increasing the Stretch)

There is a significant difference between the spanner H'_0, and the spanners H'_j for $j \geq 1$. While the aspect ratio of the graph G_0 may be unbounded, the aspect ratio of all the graphs $\tilde{G}_j$, $1 \leq j \leq \ell$, is bounded above by ρ. This suggests that we may view the graphs $\tilde{G}_j$ as unweighted, and will thus be able to use Algorithm *UnwtdSp* for building the spanners H'_j for them. As a result, the stretch of each spanner H'_j will increase by a factor of ρ.

Denote by $\mathcal{H}^*$ the variant of H^* obtained by employing Algorithm *UnwtdSp* for building the spanners H'_j, $1 \leq j \leq \ell$. (We still use Algorithm *WtdSp* to build the spanner H'_0.)

We will next analyze the properties of the resulting construction $\mathcal{H}^*$. This analysis is very similar to the analysis of the basic construction that is given in Section 2.2, hence we aim for conciseness.

Stretch. We first show how to adapt the stretch analysis of Section 2.2.1 to the new construction $\mathcal{H}^*$.

The proof of Lemma 2.1 carries through except for Equation (2) that needs to be changed. Observe that all weights of edges in $\tilde{E}_j$ belong to $W_j = (\xi_j, \rho \cdot \xi_j]$. Hence, they differ from each other by at most a multiplicative factor of ρ. Note that H'_j is an *unweighted* $(2k-1)$*-spanner* for $\tilde{G}_j$, that is, a subgraph of $\tilde{G}_j$ that contains, for every edge $e \in \tilde{E}_j$, a path with at most $(2k-1)$ edges between its endpoints. It is easy to see that H'_j provides a $(\rho \cdot (2k-1))$-spanner for $\tilde{G}_j$. Hence, instead of Equation (2), we will now have

$$\omega(\Pi'(r_j(u), r_j(v))) \leq \rho \cdot (2k-1) \cdot dist_{\tilde{G}_j}(r_j(u), r_j(v)) \leq \rho \cdot (2k-1) \cdot \omega(\tilde{e}) \leq \rho \cdot (2k-1) \cdot \omega(e).$$

As a result, the upper bound on the weight of the path $\Pi^*(r_j(u), r_j(v))$ will also increase by a factor of ρ, and we will have $\omega(\Pi^*(r_j(u), r_j(v))) \leq \rho \cdot (2k-1) \cdot \omega(e) \cdot (1+\frac{1}{q})$. Consequently, we will get that $\omega(\Pi^*(u, v)) \leq \omega(e) \cdot (\rho \cdot (2k-1) \cdot (1+\frac{1}{q}) + \frac{1}{q})$. Hence, the stretch of $\mathcal{H}^*$ is $\rho \cdot (2k-1) \cdot (1+\frac{1}{q}) + \frac{1}{q}$.

Number of Edges. Next, we show how to adapt the size analysis for the new construction $\mathcal{H}^*$.

The only change from the proof of Lemma 2.2 is that now the upper bound on $|E^*_j| = |E'_j|$, for $1 \leq j \leq \ell$, is smaller by a factor of k. The reason is that we now use Algorithm $UnwtdSp$ rather than Algorithm $WtdSp$ to build the spanner $H'_j = (V_j, E'_j)$, for $1 \leq j \leq \ell$; compare Theorem 1.1 with Theorem 1.2. Note that the bound on $|E'_0|$ remains unchanged, though, as we still use Algorithm $WtdSp$ to build the spanner H'_0. We will get

$$\begin{aligned}
|E^*| &= |E_T| + |E'_0| + \sum_{j=1}^{\ell} |E^*_j| = n - 1 + O(k \cdot n^{1+1/k}) + \sum_{1 \leq j < \varphi+1} |E^*_j| + \sum_{\varphi+1 \leq j \leq \ell} |E^*_j| \\
&= O(k \cdot n^{1+1/k}) + \lceil \varphi \rceil \cdot O(n^{1+1/k}) + O(q^{1+1/k} \cdot n^{1+1/k}) \cdot \sum_{\varphi+1 \leq j \leq \ell} \left(\frac{1}{\rho^{1+1/k}}\right)^{j-1} \\
&= O\left(n^{1+1/k} \cdot \left(k + \frac{q^{1+1/k}}{\rho^{1+1/k} - 1}\right)\right).
\end{aligned}$$

Weight. We now show how to adapt the weight analysis for the new construction $\mathcal{H}^*$.

The only change from the proof of Lemma 2.3 is that now the upper bound on $\omega(E^*_j)$, $1 \leq j \leq \ell$, is smaller by a factor of k. The bound on $\omega(E'_0)$ remains unchanged (see Equation (8)). Hence,

$$\begin{aligned}
\omega(E^*) &= \omega(T) + \omega(E'_0) + \sum_{1 \leq j < \varphi+1} \omega(E^*_j) + \sum_{\varphi+1 \leq j \leq \ell} \omega(E^*_j) \\
&= \omega(T) + O(k \cdot n^{1/k} \cdot L) + O(n^{1/k} \cdot L \cdot q) + O\left(k \cdot q^{1+1/k} \cdot n^{1/k} \cdot L \cdot \frac{1}{\rho - 1}\right) \\
&= O\left(k \cdot n^{1/k} \cdot \left(1 + \frac{q^{1+1/k}}{\rho - 1}\right)\right) \cdot \omega(MST(G)).
\end{aligned}$$

Running time. Clearly, the running time of $\mathcal{H}^*$ is not higher than that of the basic construction.

We summarize the properties of the resulting construction $\mathcal{H}^*$ in the following theorem.

THEOREM 3.1. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|, m = |E|$. For any integer $k \geq 1$, and any numbers $q > \frac{1}{2k-1}$ and $1 < \rho \leq 2$, a $(\rho \cdot (2k-1) \cdot (1 + \frac{1}{q}) + \frac{1}{q})$-spanner with $O(n^{1+1/k} \cdot (k + \frac{q^{1+1/k}}{\rho^{1+1/k}-1}))$ edges and lightness $O(k \cdot n^{1/k} \cdot (1 + \frac{q^{1+1/k}}{\rho-1}))$, can be built in $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$ time.*

By substituting $q = \Theta(1/\epsilon)$, $\rho = 1 + \Theta(\epsilon)$ in Theorem 3.1, for an arbitrary small $\epsilon > 0$, we obtain:

COROLLARY 3.2. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|, m = |E|$. For any integer $k \geq 1$ and any small $\epsilon > 0$, a $\big((2k-1) \cdot (1+\epsilon)\big)$-spanner with $O(n^{1+1/k} \cdot (k + (1/\epsilon)^{2+1/k}))$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{2+1/k})$, can be built in $O(k \cdot m + \min\{n \cdot \log n, m \cdot \alpha(n)\})$ time.*

Corollary 3.2 in the particular case $k = O(\log n)$ gives rise to an $O(\log n)$-spanner with $O(n \cdot \log n)$ edges and lightness $O(\log n)$. The running time of this construction is $O(m \cdot \log n)$.

It is instructive to compare this result with that of Corollary 2.6. If ϵ is constant, then the lightness in Corollary 3.2 is better by a factor of k. However, for $\epsilon \ll 1/k$, the number of edges and lightness in Corollary 2.6 are smaller than in Corollary 3.2.

3.2. Second Variant (Increasing the Running Time)

Denote by $\check{\mathcal{H}}$ the variant of H^* obtained by employing Algorithm *UnwtdSp* for building the spanners H'_j, $1 \leq j \leq \ell$, and employing Algorithm *WtdSp*$_2$ (due to Roditty and Zwick [2004]) to build the spanner H'_0. It is easy to see that the stretch bound of the resulting construction $\check{\mathcal{H}}$ is equal to that of $\mathcal{H}^*$. Also, the size and weight bounds of $\check{\mathcal{H}}$ are better than those of H^* and $\mathcal{H}^*$, but the running time (which is dominated by the running time of Algorithm *WtdSp*$_2$) is higher. The analysis of this variant is very similar to the analysis of the basic construction (in Section 2.2) and the analysis of the first variant (in Section 3.1), hence we aim for conciseness.

Stretch. The stretch analysis of the new construction $\check{\mathcal{H}}$ is identical to the analysis of the first variant (from Section 3.1).

Number of Edges. Next, we show how to adapt the size analysis for the new construction $\check{\mathcal{H}}$.

The only change from the size analysis of the first variant is that now the upper bound on $|E'_0|$ is smaller by a factor of k. The reason is that we now use Algorithm *WtdSp*$_2$ rather than Algorithm *WtdSp* to build the spanner $H'_0 = (V_0, E'_0)$; compare Theorem 1.3 with Theorem 1.2. Hence, we have

$$\begin{aligned}
|E^*| &= |E_T| + |E'_0| + \sum_{j=1}^{\ell} |E^*_j| = n - 1 + O(n^{1+1/k}) + \sum_{1 \leq j < \varphi+1} |E^*_j| + \sum_{\varphi+1 \leq j \leq \ell} |E^*_j| \\
&= O(n^{1+1/k}) + \lceil \varphi \rceil \cdot O(n^{1+1/k}) + O(q^{1+1/k} \cdot n^{1+1/k}) \cdot \sum_{\varphi+1 \leq j \leq \ell} \left(\frac{1}{\rho^{1+1/k}} \right)^{j-1} \\
&= O\left(n^{1+1/k} \cdot \left(1 + \frac{q^{1+1/k}}{\rho^{1+1/k} - 1} \right) \right).
\end{aligned}$$

Weight. We now show how to adapt the weight analysis for the new construction $\check{\mathcal{H}}$.

The only change from the weight analysis of the first variant is that now the upper bound on $\omega(E'_0)$ is smaller by a factor of k. (This is because now the upper bound on $|E'_0|$ is $O(n^{1+1/k})$ rather than $O(k \cdot n^{1+1/k})$.) Hence, we have

$$\begin{aligned}\omega(E^*) &= \omega(T) + \omega(E'_0) + \sum_{1 \le j < \varphi+1} \omega(E^*_j) + \sum_{\varphi+1 \le j \le \ell} \omega(E^*_j) \\ &= \omega(T) + O(n^{1/k} \cdot L) + O(n^{1/k} \cdot L \cdot q) + O\left(k \cdot q^{1+1/k} \cdot n^{1/k} \cdot L \cdot \frac{1}{\rho - 1}\right) \\ &= O\left(k \cdot n^{1/k} \cdot \frac{q^{1+1/k}}{\rho - 1}\right) \cdot \omega(MST(G)).\end{aligned}$$

Running time. Next, we show how to adapt the running time analysis for the new construction $\check{\mathcal{H}}$.

Two changes in the proof of Lemma 2.4 are in order. First, the upper bound on the time required to build the spanner H'_0 for $G_0 = (V, E_0)$ increases from $O(k \cdot m)$ to $O(k \cdot n^{2+1/k})$; compare Theorem 1.3 with Theorem 1.2. Second, the time required to build the spanner H'_j for $\tilde{G}_j = (V, \tilde{E}_j)$, $1 \le j \le \ell$, is smaller by a factor of k. (This is because now Algorithm $UnwtdSp$ is invoked on each graph $\tilde{G}_j$, rather than Algorithm $WtdSp$.) Hence, the overall time required to build all spanners H'_j is upper bounded by $\sum_{j=1}^{\ell} O(|E_j|) = O(m)$. It follows that the total running time of the construction is now $O(\min\{m + n \cdot \log n, m \cdot \alpha(m, n)\}) + O(k \cdot n^{2+1/k}) + O(m) = O(k \cdot n^{2+1/k})$.

We summarize the properties of the resulting construction $\check{\mathcal{H}}$ in the following theorem.

THEOREM 3.3. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|$. For any integer $k \ge 1$, and any numbers $q > \frac{1}{2k-1}$ and $1 < \rho \le 2$, a $(\rho \cdot (2k-1) \cdot (1 + \frac{1}{q}) + \frac{1}{q})$-spanner with $O(n^{1+1/k} \cdot (1 + \frac{q^{1+1/k}}{\rho^{1+1/k}-1}))$ edges and lightness $O(k \cdot n^{1/k} \cdot \frac{q^{1+1/k}}{\rho-1})$, can be built in $O(k \cdot n^{2+1/k})$ time.*

By substituting $q = \Theta(1/\epsilon)$, $\rho = 1 + \Theta(\epsilon)$ in Theorem 3.3, for an arbitrary small $\epsilon > 0$, we obtain:

COROLLARY 3.4. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|$. For any integer $k \ge 1$ and any small $\epsilon > 0$, a $((2k-1) \cdot (1+\epsilon))$-spanner with $O(n^{1+1/k} \cdot (1/\epsilon)^{2+1/k})$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{2+1/k})$, can be built in $O(k \cdot n^{2+1/k})$ time.*

Note that the trade-off between the stretch, number of edges, and lightness exhibited in Corollary 3.4 is, up to factors polynomial in $1/\epsilon$, the same as in Chandra et al. [1992]. On the other hand, our running time is $O(k \cdot n^{2+1/k})$, instead of $O(m \cdot (n^{1+1/k} + n \cdot \log n))$ in Chandra et al. [1992].

By substituting $q = \Theta(1/k)$, $\rho = 2$ in Theorem 3.3, we obtain:

COROLLARY 3.5. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|$. For any integer $k \ge 1$, an $O(k^2)$-spanner with $O(n^{1+1/k})$ edges and lightness $O(n^{1/k})$, can be built in $O(k \cdot n^{2+1/k})$ time.*

Corollary 3.5 in the particular case $k = O(\log n)$ gives rise to an $O(\log^2 n)$-spanner with $O(n)$ edges and lightness $O(1)$. We can extend this result to get a general trade-off between the three parameters by substituting $k = O(\log n)$, $\rho = 2$ in Theorem 3.3, and taking $1 \le \ell = \frac{1}{q} = O(\log n)$.

COROLLARY 3.6. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|$. For any number $1 \le \ell = O(\log n)$, an $O(\log n \cdot \ell)$-spanner with $O(n)$ edges and lightness $O(\frac{\log n}{\ell})$, can be built in $O(n^2 \cdot \log n)$ time.*

The trade-off of Corollary 3.6 was also given by Chandra et al. [1992], but their running time is $O(m \cdot n \cdot \log n)$.

3.3. Third Variant (Increasing the Running Time Some More)

Denote by $\hat{\mathcal{H}}$ the variant of H^* obtained by employing Algorithm $WtdSp_2$ for building all the spanners H'_j, $1 \le j \le \ell$, as well as the spanner H'_0. It is easy to see that the stretch bound of the resulting construction $\hat{\mathcal{H}}$ is equal to that of the basic construction H^*. Also, the size and weight bounds of $\hat{\mathcal{H}}$ are exactly the same as those of the second variant $\check{\mathcal{H}}$, but the running time is slightly higher (by a factor of $(1 + q^2)$) than that of $\check{\mathcal{H}}$. The analysis of this variant is very similar to the preceding, and is thus omitted.

We summarize the properties of the resulting construction $\hat{\mathcal{H}}$ in the following theorem. (We substituted $\rho = 2$ to optimize the parameters of the construction.)

THEOREM 3.7. *Let $G = (V, E, \omega)$ be a weighted graph, with $n = |V|$. For any integer $k \ge 1$, and any small $\epsilon > 0$, a $((2k-1) \cdot (1+\epsilon))$-spanner with $O(n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$, can be built in $O(k \cdot n^{2+1/k} \cdot (1/\epsilon)^{2+1/k})$ time.*

This result is closely related to Corollary 3.4, but the dependencies on ϵ are slightly different.

3.4. Fourth Variant (Integer-Weighted Graphs)

The fourth variant of our construction applies only to integer-weighted graphs. Denote by $\mathcal{H}_{int}$ the variant of H^* obtained by employing Algorithm $IntWtdSp$ for building all the black-box spanners, that is, the spanners H'_j, $1 \le j \le \ell$, and the spanner H'_0. Disregarding the running time, the new construction achieves exactly the same bounds as the basic construction H^*. However, since the size bound of the spanners H'_j, $0 \le j \le \ell$, built via Algorithm $IntWtdSp$ is expected and not deterministic, both the size and lightness bounds of the constructed spanner H^* are expected. The running time of this variant consists of two parts. Specifically, it is the time required to compute the MST and the time required to compute a spanner. On an integer-weighted graph the first task can be done in $O(m + n)$ time [Fredman and Willard 1994], while the second task requires $O(SORT(m))$ time (see Theorem 1.4 in Elkin [2011]). Consequently, the overall running time is $O(SORT(m))$, and this bound is deterministic. The stretch bound is deterministic as well.

We summarize the properties of the new construction $\mathcal{H}_{int}$ in the following theorem.

THEOREM 3.8. *Let $G = (V, E, \omega)$ be an integer-weighted graph, with $n = |V|, m = |E|$. For any integer $k \ge 1$ and any small $\epsilon > 0$, a $((2k-1) \cdot (1+\epsilon))$-spanner with expected $O(k \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k}))$ edges and expected lightness $O(k^2 \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$, can be built in $O(SORT(m))$ time.*

3.5. Spanners in the Streaming Model

In this section, we analyze our algorithm in the augmented streaming model. Specifically, this is the model introduced by Aggarwal et al. [2004], which allows sorting passes over the input.

Our algorithm relies on a streaming algorithm for constructing sparse (but possibly heavy) spanners from Elkin [2011], summarized in the following Theorem 3.9. We remark that Theorem 3.9 also applies to multigraphs.

THEOREM 3.9. *[Elkin 2011] For any unweighted n-vertex graph $G = (V, E)$ and any integer $k \geq 1$, there exists a one-pass streaming algorithm that computes a $(2k-1)$-spanner with $O(k \cdot n^{1+1/k})$ edges (expected). The processing time per edge of the algorithm is $O(1)$, and its space requirement is $O(k \cdot n^{1+1/k})$ (expected).*

Also, in the augmented streaming model (if the algorithm accepts a sorted stream of edges as input) the algorithm produces $(2k-1)$-spanners with $O(k \cdot n^{1+1/k})$ edges (expected) for weighted graphs as well. The processing time per edge and space requirements of this algorithm are the same as in the unweighted case.

Our algorithm will run $\ell + 1$ copies of the algorithm for weighted graphs from Theorem 3.9. We will denote these copies $\mathcal{A}_j$, for $0 \leq j \leq \ell$.

Our algorithm starts with a sorting pass over the stream of edges. After this pass, in the consecutive pass the algorithm reads edges in a nondecreasing order of weights. The objective of the first pass (after the sorting pass) is to compute an MST of the input graph. To accomplish this, the algorithm maintains a Union-Find data structure (see Ch. 21 in Corman et al. [2001]). It is known [Blum 1986; Alstrup et al. 1999] that all operations can be performed in worst-case $O(\frac{\log n}{\log\log n})$ time, and in total $O(n + m \cdot \alpha(n))$ time, using $O(n)$ space.

As a result of the first pass, the MST T is computed. The Hamiltonian path $\mathcal{L}$ of M_T is computed between the passes. In addition, the algorithm maintains the location on $\mathcal{L}$ of every vertex $v \in V$. It also initializes $\ell + 1$ arrays A_j of size n_j each, $0 \leq j \leq \ell$.

Then the algorithm performs the second pass over the sorted stream of edges. For each edge $e = (u, v)$ that the algorithm reads in the second pass, the algorithm determines the index j such that $\omega(e) \in W_j$. Then it tests if the edge crosses between different j-level intervals, that is, belongs to $\bar{E}_j$. If it does not, this edge is skipped. Otherwise the algorithm passes the edge e to the jth copy $\mathcal{A}_j$ of the weighted streaming algorithm from Theorem 3.9. The streaming algorithm $\mathcal{A}_j$ will ultimately produce the spanner H'_j for $\hat{G}_j = (V_j, \hat{E}_j)$. (Here we follow the notation of Section 2.1.) If the streaming algorithm $\mathcal{A}_j$ decides to insert e into the spanner H'_j, it will also insert its source edge $s(e)$ into the ultimate spanner $H^* = (V, E^*)$. As a result, the spanner H^* will contain the union of the j-level spanners, for $0 \leq j \leq \ell$. The edge set of the MST will also be inserted into H^* (either between the passes, or after the second pass).

This completes the description of the algorithm. By Theorem 3.9, the (expected) space requirement in the second pass is $\sum_{j=0}^{\ell} O(k \cdot n_j^{1+1/k}) = O(k \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$. The processing time per edge is $O(1)$.

We summarize our streaming algorithm in the following theorem.

THEOREM 3.10. *In the augmented streaming model, for any weighted graph $G = (V, E, \omega)$, any integer $k \geq 1$, and any $\epsilon > 0$, our algorithm requires two passes after the sorting pass. It computes a $((2k-1) \cdot (1+\epsilon))$-spanner with expected $O(k \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$ edges and expected lightness $O(k^2 \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$. The expected space requirement is $O(k \cdot n^{1+1/k} \cdot (1/\epsilon)^{1+1/k})$. The worst-case processing time per edge of the first (second, respectively) pass is $O(\frac{\log n}{\log\log n})$ ($O(1)$, respectively). Moreover, the overall processing time of the first pass is $O(m \cdot \alpha(n))$.*

As was mentioned in the Introduction, the algorithm of Chandra et al. [1992] can be viewed as an algorithm in this model. It constructs $((2k-1) \cdot (1+\epsilon))$-spanners with $O(n^{1+1/k})$ edges and lightness $O(k \cdot n^{1/k} \cdot (1/\epsilon)^{1+1/k})$. It requires one pass after the initial sorting pass, but its processing time-peg-edge is very large (specifically, it is $O(n^{1+1/k} + n \cdot \log n)$).

REFERENCES

G. Aggarwal, M. Datar, S. Rajagopalan, and M. Ruhl. 2004. On the streaming model augmented with a sorting primitive. In *Proc. of 45th FOCS*. 540–549.

N. Alon, Y. Matias, and M. Szegedy. 1999. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.* 58, 1 (1999), 137–147.

S. Alstrup, A. M. Ben-Amram, and T. Rauhe. 1999. Worst-case and amortised optimality in union-find (extended abstract). In *Proc. of 31st STOC*. 499–506.

I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares. 1993. On sparse spanners of weighted graphs. *Discr. Comput. Geom.* 9 (1993), 81–100.

S. Arora. 1998. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM* 45, 5 (1998), 753–782.

B. Awerbuch. 1985. Communication-time trade-offs in network synchronization. In *Proc. of 4th PODC*. 272–276.

B. Awerbuch, A. Baratz, and D. Peleg. 1992. *Efficient Broadcast and Light-Weight Spanners*. Technical Report CS92-22. Weizmann Institute.

Y. Bartal, L. Gottlieb, and R. Krauthgamer. 2012. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. In *Proc. of 44th STOC*. 663–672.

S. Baswana. 2008. Streaming algorithm for graph spanners—Single pass and constant processing time per edge. *Inf. Process. Lett.* 106, 3 (2008), 110–114.

S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. 2010. Additive spanners and (α, β)-spanners. *ACM Trans. Algor.* 7, 1 (2003), 5.

S. Baswana and S. Sen. 2003. A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Proc. of 30th ICALP*. 384–296.

N. Blum. 1986. On the single-operation worst-case time complexity of the disjoint set union problem. *SIAM J. Comput.* 15, 4 (1986), 1021–1024.

B. Chandra, G. Das, G. Narasimhan, and J. Soares. 1992. New sparseness results on graph spanners. In *Proc. of 8th SOCG*. 192–201.

B. Chazelle. 2000. The complexity of computing partial sums off-line. *J. ACM* 47, 6 (2000), 1028–1047.

E. Cohen. 1993. Fast algorithms for constructing t-spanners and paths with stretch t. In *Proc. of 34th FOCS*. 648–658.

T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms* (2nd. ed.). McGraw-Hill, Boston, MA.

G. Das and G. Narasimhan. 1997. A fast algorithm for constructing sparse Euclidean spanners. *Int. J. Comput. Geometry Appl.* 7, 4 (1997), 297–315.

M. Elkin. 2005. Computing almost shortest paths. *ACM Trans. Algor.* 1, 2 (2005), 283–323.

M. Elkin. 2011. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algor.* 7, 2 (2011), 20.

M. Elkin and D. Peleg. 2004. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM J. Comput.* 33, 3 (204), 608–631.

M. Elkin and J. Zhang. 2006. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. *Distrib. Comput.* 18, 5 (2006), 375–385.

J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. 2005. Graph distances in the streaming model: The value of space. In *Proc. of 16th SODA*. 745–754.

J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. 2002. An approximate l1-difference algorithm for massive data streams. *SIAM J. Comput.* 32, 1 (2002), 131–151.

M. L. Fredman and D. E. Willard. 1994. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci.* 48, 3 (1994), 533–551.

J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. 2002. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.* 31, 5 (2002), 1479–1500.

S. Halperin and U. Zwick. 1996. Linear time deterministic algorithm for computing spanners for unweighted graphs.

Y. Han. 2004. Deterministic sorting in $O(n \log\log n)$ time and linear space. *J. Algorithms* 50, 1 (2004), 96–105.

Y. Han and M. Thorup. 2002. Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space. In *Proc. of 43rd FOCS*. 135–144.

D. R. Karger, P. N. Klein, and R. E. Tarjan. 1995. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM* 42, 2 (1995), 321–328.

J. S. B. Mitchell. 1999. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.* 28, 4 (1999), 1298–1309.

D. Peleg. 1999. Proximity-preserving labeling schemes and their applications. In *Proc. of 25th WG*. 30–41.

D. Peleg and A. Schäffer. 1989. Graph spanners. *J. Graph Theory* 13, 1 (1989), 99–116.

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

D. Peleg and E. Upfal. 1989. A trade-off between space and efficiency for routing tables. *J. ACM* 36, 3 (1989), 510–530.

S. Pettie. 2009. Low distortion spanners. *ACM Trans. Algor.* 6, 1 (2009).

S. Rao and W. D. Smith. 1998. Approximating geometrical graphs via "spanners" and "banyans". In *Proc. of 30th STOC*. 540–550.

L. Roditty, M. Thorup, and U. Zwick. 2005. Deterministic constructions of approximate distance oracles and spanners. In *Proc. of 32nd ICALP*. 261–272.

L. Roditty and U. Zwick. 2004. On dynamic shortest paths problems. In *Proc. of 32nd ESA*. 580–591.

M. Thorup and U. Zwick. 2001a. Approximate distance oracles. In *Proc. of 33rd STOC*. 183–192.

M. Thorup and U. Zwick. 2001b. Compact routing schemes. In *Proc. of 13th SPAA*. 1–10.

M. Thorup and U. Zwick. 2006. Spanners and emulators with sublinear distance errors. In *Proc. of 17th SODA*. 802–809.

D. P. Woodruff. 2006. Lower bounds for additive spanners, emulators, and more. In *Proc. of 47th FOCS*. 389–398.

Received September 2012; revised April 2015; accepted October 2015