View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Constructing Light Spanners Deterministically in Near-Linear Time

Stephen Alstrup
University of Copenhagen, Denmark
s.alstrup@di.ku.dk

Søren Dahlgaard
SupWiz, Copenhagen, Denmark
University of Copenhagen, Denmark
soerend@di.ku.dk

Arnold Filtser
Ben Gurion University of the Negev, Israel
arnold273@gmail.com

Morten Stöckel
University of Copenhagen, Denmark
most@di.ku.dk

Christian Wulff-Nilsen
University of Copenhagen, Denmark
koolooz@di.ku.dk

Abstract

Graph spanners are well-studied and widely used both in theory and practice. In a recent breakthrough, Chechik and Wulff-Nilsen [11] improved the state-of-the-art for light spanners by constructing a $(2k-1)(1+\varepsilon)$-spanner with $O(n^{1+1/k})$ edges and $O_\varepsilon(n^{1/k})$ lightness. Soon after, Filtser and Solomon [19] showed that the classic greedy spanner construction achieves the same bounds. The major drawback of the greedy spanner is its running time of $O(mn^{1+1/k})$ (which is faster than [11]). This makes the construction impractical even for graphs of moderate size. Much faster spanner constructions do exist but they only achieve lightness $\Omega_\varepsilon(kn^{1/k})$, even when randomization is used.

The contribution of this paper is deterministic spanner constructions that are fast, and achieve similar bounds as the state-of-the-art slower constructions. Our first result is an $O_\varepsilon(n^{2+1/k+\varepsilon'})$ time spanner construction which achieves the state-of-the-art bounds. Our second result is an $O_\varepsilon(m+n\log n)$ time construction of a spanner with $(2k-1)(1+\varepsilon)$ stretch, $O(\log k\cdot n^{1+1/k})$ edges and $O_\varepsilon(\log k\cdot n^{1/k})$ lightness. This is an exponential improvement in the dependence on k compared to the previous result with such running time. Finally, for the important special case where $k=\log n$, for every constant $\varepsilon>0$, we provide an $O(m+n^{1+\varepsilon})$ time construction that produces an $O(\log n)$-spanner with $O(n)$ edges and $O(1)$ lightness which is asymptotically optimal. This is the first known sub-quadratic construction of such a spanner for any $k=\omega(1)$.

To achieve our constructions, we show a novel deterministic incremental approximate distance oracle. Our new oracle is crucial in our construction, as known randomized dynamic oracles require the assumption of a non-adaptive adversary. This is a strong assumption, which has seen recent attention in prolific venues. Our new oracle allows the order of the edge insertions to not be fixed in advance, which is critical as our spanner algorithm chooses which edges to insert based on the answers to distance queries. We believe our new oracle is of independent interest.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners; Theory of computation → Dynamic graph algorithms

Keywords and phrases Spanners, Light Spanners, Efficient Construction, Deterministic Dynamic Distance Oracle

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.4

Related Version The reader is encouraged to read the arXiv version `https://arxiv.org/abs/1709.01960`, [1].

© Stephen Alstrup, Søren Dahlgaard, Arnold Filtser, Morten Stöckel, and Christian Wulff-Nilsen;
licensed under Creative Commons License CC-BY

27th Annual European Symposium on Algorithms (ESA 2019).
Editors: Michael A. Bender, Ola Svensson, and Grzegorz Herman; Article No. 4; pp. 4:1–4:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Stephen Alstrup*: Research partly supported by Innovationsfonden DK, DABAI (5153-00004A) and by VILLUM Foundation grant 16582, Basic Algorithms Research Copenhagen (BARC).
Søren Dahlgaard: Research partly supported by Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research.
Arnold Filtser: Supported in part by ISF grant No. (1817/17) and by BSF grant No. 2015813.
Morten Stöckel: Research partly supported by Villum Fonden.
Christian Wulff-Nilsen: Research partly supported by the Starting Grant 7027-00050B from the Independent Research Fund Denmark under the Sapere Aude research career programme.

Acknowledgements We are grateful to Michael Elkin, Ofer Neiman, and Sebastian Forster for fruitful discussions.

1 Introduction

A fundamental problem in graph data structures is compressing graphs such that certain metrics are preserved as well as possible. A popular way to achieve this is through *graph spanners*. Graph spanners are sparse subgraphs that approximately preserve pairwise shortest path distances for all vertex pairs. Formally, we say that a subgraph $H = (V, E', w)$ of an edge-weighted undirected graph $G = (V, E, w)$ is a *t-spanner* of G if for all $u, v \in V$ we have $d_H(u, v) \leq t \cdot d_G(u, v)$, where d_X is the shortest path distance function for graph X and w is the edge weight function. Under such a guarantee, we say that our graph spanner H has *stretch* t. In the following, we assume that the underlying graph G is connected; if it is not, we can consider each connected component separately when computing a spanner.

Graph spanners originate from the 80's [24, 25] and have seen applications in e.g. synchronizers [25], compact routing schemes [30, 26, 8], broadcasting [18], and distance oracles [32].

The two main measures of the sparseness of a spanner H is the size (number of edges) and the *lightness*, which is defined as the ratio $w(H)/w(MST(G))$, where $w(H)$ resp. $w(MST(G))$ is the total weight of edges in H resp. a minimum spanning tree (MST) of G. It has been established that for any positive integer k, a $(2k-1)$-spanner of $O(n^{1+1/k})$ edges exists for any n-vertex graph [3]. This stretch-size tradeoff is widely believed to be optimal due to a matching lower bound implied by Erdős' girth conjecture [16], and there are several papers concerned with constructing spanners efficiently that get as close as possible to this lower bound [31, 5, 28].

Obtaining spanners with small lightness (and thus total weight) is motivated by applications where edge weights denote e.g. establishing cost. The best possible total weight that can be achieved in order to ensure finite stretch is the weight of an MST, thus making the definition of lightness very natural. The size lower bound of the unweighted case provides a lower bound of $\Omega(n^{1/k})$ lightness under the girth conjecture, since H must have size and weight $\Omega(n^{1+1/k})$ while the MST has size and weight $n-1$. Obtaining this lightness has been the subject of an active line of work [2, 7, 14, 11, 19]. Throughout this paper we say that a spanner is *optimal* when its bounds coincide asymptotically with those of the girth conjecture. Obtaining an efficient spanner construction with optimal stretch-lightness trade-off remains one of the main open questions in the field of graph spanners.

Light spanners. Historically, the main approach of obtaining a spanner of bounded lightness has been through different analyses of the classic *greedy spanner*. Given $t \geq 1$, the greedy t-spanner is constructed as follows: iterate through the edges in non-decreasing order of weight and add an edge e to the partially constructed spanner H if the shortest path distance in H between the endpoints of e is greater than t times the weight of e. The study of this

spanner algorithm dates back to the early 90's with its first analysis by Althöfer et al. [2]. They showed that this simple procedure with stretch $2k-1$ obtains the optimal $O(n^{1+1/k})$ size, and has lightness $O(n/k)$. The algorithm was subsequently analyzed in [7, 14, 19] with stretch $(1+\varepsilon)(2k-1)$ for any $0<\varepsilon<1$. Recently, a break-through result of Chechik and Wulff-Nilsen [11] showed that a significantly more complicated spanner construction obtains nearly optimal stretch, size and lightness giving the following theorem.

▶ **Theorem 1** ([11]). *Let $G=(V,E,w)$ be an edge-weighted undirected n-vertex graph and let k be a positive integer. Then for any $0<\varepsilon<1$ there exists a $(1+\varepsilon)(2k-1)$-spanner of size $O(n^{1+1/k})$ and lightness $O_\varepsilon(n^{1/k})$.*[1]

Following the result of [11] it was shown by Filtser and Solomon [19] that this bound is matched by the greedy spanner. In fact, they show that the greedy spanner is *existentially optimal*, meaning that if there is a t-spanner construction achieving an upper bound $m(n,t)$ resp. $l(n,t)$ on the size resp. lightness of any n-vertex graph then this bound also holds for the greedy t-spanner. In particular, the bounds in Theorem 1 also hold for the greedy spanner.

Efficient spanners. A major drawback of the greedy spanner is its $O(m\cdot(n^{1+1/k}+n\log n))$ construction time [2]. Similarly, Chechik and Wulff-Nilsen [11] only state their construction time to be polynomial, but since they use the greedy spanner as a subroutine, it has the same drawback. Adressing this problem, Elkin and Solomon [15] considered efficient construction of light spanners. They showed how to construct a spanner with stretch $(1+\varepsilon)(2k-1)$, size $O_\varepsilon(kn^{1+1/k})$ and lightness $O_\varepsilon(kn^{1/k})$ in time $O(km+\min(n\log n, m\alpha(n)))$. Improving on this, a recent paper of Elkin and Neiman [13] uses similar ideas to obtain stretch $(1+\varepsilon)(2k-1)$, size $O(\log k\cdot n^{1+1/k})$ and lightness $O(kn^{1/k})$ in expected time $O(m+\min(n\log n, m\alpha(n)))$.

Several papers also consider efficient constructions of sparse spanners, which are not necessarily light. Baswana and Sen [5] gave a $(2k-1)$-spanner with $O(kn^{1+1/k})$ edges in $O(km)$ expected time. This was later derandomized by Roditty et al. [27] (while keeping the same sparsity and running time). Recently, Miller et al. [23] presented a randomized algorithm with $O(m+n\log k)$ running time and $O(\log k\cdot n^{1+1/k})$ size at the cost of a constant factor in the stretch $O(k)$.

It is worth noting that for super-constant k, none of the above spanner constructions obtain the optimal $O(n^{1+1/k})$ size or $O(n^{1/k})$ lightness even if we allow $O(k)$ stretch. If we are satisfied with nearly-quadratic running time, Elkin and Solomon [15] gave a spanner with $(1+\varepsilon)(2k-1)$ stretch, $O_\varepsilon(n^{1+1/k})$ size and $O_\varepsilon(kn^{1/k})$ lightness in $O(kn^{2+1/k})$ time by extending a result of Roditty and Zwick [28] who got a similar result but with unbounded lightness. However, this construction still has a factor of k too much in the lightness. Thus, the fastest known spanner construction obtaining optimal size and lightness is the classic greedy spanner – even if we allow $O(k)$ stretch or $o(kn^{1/k})$ lightness.

We would like to emphasize that the case $k=\log n$ is of special interest. This is the point on the tradeoff curve allowing spanners of linear size and constant lightness. Prior to this paper, the state of the art for efficient spanner constructions with constant lightness suffered from distortion at least $O(\log^2 n)$. See the discussion after Corollary 5 for further details.

A summary of spanner algorithms can be seen in Table 1.

[1] O_ε notation hides polynomial factors in $1/\varepsilon$.

Table 1 Table of related spanner constructions. In the top of the table we list non-efficient spanner constructions. In the middle we list known efficient spanner construction. In the bottom we list our contributions. Results marked * are different analyses of the greedy spanner. Results marked # are randomized. Lightness complexities marked ** appear in the full version [1] and W denotes the maximum edge weight of the input graph. The bounds hold for any constant $\varepsilon, \varepsilon' > 0$.

Stretch	Size	Lightness	Construction	Ref
$(2k-1)$	$O\left(n^{1+1/k}\right)$	$O\left(n/k\right)$	$O\left(mn^{1+1/k}\right)$	[ADD+93][2]*
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(mn^{1+1/k}\right)$	[CDNS92][7]*
$(2k-1)$	$O(n^{1+1/k})$	$\Omega(W)$ **	$O\left(kn^{2+1/k}\right)$	[RZ11][28]
$(2k-1)$	$O\left(kn^{1+1/k}\right)$	$\Omega\left(n^{1+1/k}\right)$ **	$O\left(kmn^{1/k}\right)$	[TZ05][31]#
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(kn^{2+1/k}\right)$	[ES16][15]
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k} \cdot k/\log k\right)$	$O\left(mn^{1+1/k}\right)$	[ENS15][14]*
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$n^{\Theta(1)}$	[CW18][11]
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O\left(mn^{1+1/k}\right)$	[FS16][19]*
$(2k-1)$	$O\left(kn^{1+1/k}\right)$	$\Omega(W)$ **	$O\left(km\right)$	[BS07,RTZ05][5, 27]
$(2k-1)(1+\varepsilon)$	$O\left(kn^{1+1/k}\right)$	$O\left(kn^{1/k}\right)$	$O\left(km + n\log n\right)$	[ES16][15]
$O(k)$	$O(\log k \cdot n^{1+1/k})$	$\Omega(W)$	$O(m + n \cdot \log k)$	[MPVX15][23]#
$(2k-1)(1+\varepsilon)$	$O(\log k \cdot n^{1+1/k})$	$O\left(k \cdot n^{1/k}\right)$	$O(m + n \cdot \log n)$	[EN17][13]#
$(2k-1)(1+\varepsilon)$	$O\left(\log k \cdot n^{1+1/k}\right)$	$\Omega(W)$	$O(m)$	Theorem 7
$(2k-1)(1+\varepsilon)$	$O\left(\log k \cdot n^{1+1/k}\right)$	$O\left(\log k \cdot n^{1/k}\right)$	$O(m + n \cdot \log n)$	Theorem 3
$(2k-1)(1+\varepsilon)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O(n^{2+1/k+\varepsilon'})$	Theorem 2
$O(k)$	$O\left(n^{1+1/k}\right)$	$O\left(n^{1/k}\right)$	$O\left(m + n^{1+\varepsilon'+1/k}\right)$	Theorem 4
$O(\log n)/\delta$	$O\left(n\right)$	$1+\delta$	$O\left(m + n^{1+\varepsilon'}\right)$	Corollary 5

1.1 Our results

We present the first spanner obtaining the same near-optimal guarantees as the greedy spanner in significantly faster time by obtaining a $(1+\varepsilon)(2k-1)$ spanner with optimal size and lightness in $O_\varepsilon(n^{2+1/k+\varepsilon'})$ time. We also present a variant of this spanner, improving the running time to $O(m + n\log n)$ by paying a $\log k$ factor in the size and lightness. Finally, we present an optimal $O_\varepsilon(\log n)$-spanner which can be constructed in $O(m + n^{1+\varepsilon})$ time. This special case is of particular interest in the literature (see e.g. [4, 22]). Furthermore, all of our constructions are deterministic, giving the first subquadratic deterministic construction without the additional dependence on k in the size of the spanner. As an important tool, we introduce a new deterministic approximate incremental distance oracle which works in near-linear time for maintaining small distances approximately. We believe this result is of independent interest.

More precisely, we show the following theorems.

▶ **Theorem 2.** *Given a weighted undirected graph $G = (V, E, w)$ with m edges and n vertices, any positive integer k, and $\varepsilon, \varepsilon' > 0$ where ε arbitrarily close to 0 and ε' is a constant, one can deterministically construct an $(1+\varepsilon)(2k-1)$-spanner of G with $O_\varepsilon(n^{1+1/k})$ edges and lightness $O_\varepsilon(n^{1/k})$ in $O(n^{2+1/k+\varepsilon'})$ time.*

▶ **Theorem 3.** *Given a weighted undirected graph $G = (V, E, w)$ with m edges and n vertices, a positive integer $k \geq 640$, and $\epsilon > 0$, one can deterministically construct a $(2k-1)(1+\varepsilon)$-spanner of G with $O_\varepsilon(\log k \cdot n^{1+1/k})$ edges and lightness $O_\varepsilon\left(\log k \cdot n^{1/k}\right)$ in time $O\left(m + n\log n\right)$.*

Note that in Theorem 3 we require k to be larger than 640. This is not a significant limitation, as for $k = O(1)$ [15] is already optimal.

Our $O(\log n)$-spanner is obtained as a corollary of the following more general result.

▶ **Theorem 4.** *Given a weighted undirected graph $G = (V, E, w)$ with m edges and n vertices, any positive integer k and constant $\varepsilon' > 0$, one can deterministically construct an $O(k)$-spanner of G with $O(n^{1+1/k})$ edges and lightness $O(n^{1/k})$ in $O(m + n^{1+\varepsilon'+1/k})$ time.*

We note that the stretch $O(k)$ of Theorem 4 (and Corollary 5 below) hides an exponential factor in $1/\varepsilon'$, thus we only note the result for constant ε'. Bartal et. al. [4] showed that given a spanner construction that for every n-vertex weighted graph produces a $t(n)$-stretch spanner with $m(n,t)$ edge and $l(n,t)$ lightness in $T(n,m)$ time, then for every parameter $0 < \delta < 1$ and every graph G, one can construct a t/δ-spanner with $m(n,t)$ edges and $1 + \delta \cdot l(n,t)$ lightness in $T(n,m) + (m)$ time. Plugging $k = \log n$ and using this reduction we get

▶ **Corollary 5.** *Let $G = (V, E, w)$ be a weighted undirected n-vertex graph, let $\varepsilon' > 0$ be a constant and $\delta > 0$ be a parameter arbitrarily close to 0. Then one can construct a spanner of G with:*

1. *$O(\log n)/\delta$ stretch, $O(n)$ edges and $1 + \delta$ lightness in time $O(m + n^{1+\varepsilon'})$.*
2. *$O(\log n \log\log n)/\delta$ stretch, $O(n \log\log n)$ edges and $1+\delta$ lightness in time $O(m+n\log n)$.*

Corollary 5 above should be compared to previous attempts to efficiently construct a spanner with constant lightness. Although not stated explicitly, the state of the art algorithms of [15, 13], combined with the lemma from [4], provide an efficient spanner construction with $1 + \delta$ lightness, $O(n \log\log n)$ edges and only $O(\log^2 n/\delta)$ stretch.

We emphasize, that Corollary 5 is the first sub-quadratic construction of spanner with optimal size and lightness for any non-constant k.

In order to obtain Theorem 4 we construct the following deterministic incremental approximate distance oracle with near-linear total update time for maintaining small distances. We believe this result is of independent interest, and discuss it in more detail in the related work section below and in Section 3.

▶ **Theorem 6.** *Let G be a graph that undergoes a sequence of m edge insertions. For any constant $\varepsilon' > 0$ and parameter $d \geq 1$ there exists a data structure which processes the m insertions in total time $O(m^{1+\varepsilon'} \cdot d)$ and can answer queries at any point in the sequence of the following form. Given a pair of nodes u, v, the oracle gives, in $O(1)$ time, an estimate $\hat{d}(u,v)$ such that $\hat{d}(u,v) \geq d(u,v)$ and if $d(u,v) \leq d$ then $\hat{d}(u,v) = O(1) \cdot d(u,v)$.*

Theorem 6 assumes that ε' is constant; the O-notation hides a factor exponential in $1/\varepsilon'$ for both total update time and stretch whereas the query time bound only hides a factor of $1/\varepsilon'$.

We also obtain the following sparse, but not necessarily light, spanner in linear time as a subroutine in proving Theorem 3.

▶ **Theorem 7.** *Given a weighted undirected graph $G = (V, E, w)$ with m edges and n vertices, a positive integer k, and $\epsilon > 0$, one can deterministically construct a $(2k-1)(1+\varepsilon)$-spanner of G with $O_\varepsilon(n^{1+1/k} \cdot \log k)$ edges in time $O(m)$.*

1.2 Related work

Closely related to graph spanners are *approximate distance oracles (ADOs)*. An ADO is a data structure which, after preprocessing a graph G, is able to answer distance queries approximately. Distance oracles are studied extensively in the literature (see e.g. [31, 33, 9, 10]) and often use spanners as a building block. The state of the art static distance oracle

is due to Chechik [10], where a construction of space $O(n^{1+1/k})$, stretch $2k-1$, and query time $O(1)$ is given. Our distance oracle of Theorem 6 should be compared to the result of Henzinger, et al. [21], who gave a deterministic construction for incremental (or decremental) graphs with a total update time of $O_\varepsilon(mn \log n)$, a query time of $O(\log \log n)$ and stretch $1+\varepsilon$. For our particular application, we require near-linear total update time and only good stretch for short distances, which are commonly the most troublesome when constructing spanners. It should be added that Henzinger et al. give a general deterministic data structure for choosing centers, i.e., vertices which are roots of shortest path trees maintained by the data structure. While this data structure may be fast when the total number of centers is small, we need roughly n centers and it is not clear how this number can be reduced. Having this many centers requires at least order mn time with their data structure.

To achieve our fast update time bound, we are interested in trading worse stretch for distances above parameter d for construction time. Roditty and Zwick [29] gave a randomized distance oracle for this case, however their construction does not work against an adaptive adversary as is required for our application, where the edges to be inserted are determined by the output to the queries of the oracle (see Section 3 for more discussion on this). Removing the assumption of a non-adaptive adversary in dynamic graph algorithms has seen recent attention at prestigious venues, e.g. [34, 6]. Our new incremental approximate distance oracle for short distances given in Theorem 6 is deterministic and thus is robust against such an adversary, and we believe it may be of independent interest as a building block in deterministic dynamic graph algorithms.

For unweighted graphs, there is a folklore spanner construction by Halperin and Zwick [20] which is optimal on all parameters. The construction time is $O(m)$, it has $O(n^{1+1/k})$ edges and $2k-1$ stretch. In Section 6 we will use this spanner as a building block in proving Theorem 3.

2 Preliminaries

Consider a weighted graph $G=(V,E,w)$, we will abuse notation and refer to as E both a set of edges and the graph itself. d_G will denote the shortest path metric (that is $d_G(v,u)$ is the weight of the lightest path between v,u in G. Given a subset V' of V, $G[V']$ is the induced graph by V'. That is it has V' as it vertices, $E \cap \binom{V'}{2}$ as its edges and w as weight function. The *diameter* of a vertex set V' in a graph G' $\mathrm{diam}_{G'}(V') = \max_{u,v\in V'} d_{G'}(u,v)$ is the maximal distance between two vertices in V' under the shortest path metric induced by G'. For a set of edges A with weight function w, the *aspect ratio* of A is $\max_{e\in A} w(e)/\min_{e\in A} w(e)$. The *sparsity* of A is simply $|A|$ its size.

We will assume that $k = O(\log n)$ as the guarantee for lightness and sparsity will not be improved by picking larger k. Instead of proving $(1+\varepsilon)(2k-1)$ bound on stretch, we will prove only $(1+O(\varepsilon))(2k-1)$ bound. This is good enough, as Post factum we can scale ε accordingly. By O_ε we denote asymptotic notation which hides polynomial factors of $1/\varepsilon$, that is $O_\varepsilon(f) = O(f)\cdot \mathrm{poly}(\frac{1}{\varepsilon})$.

3 Paper overview

General framework. Theorems 2 to 4 are generated via a general framework. The framework is fed two algorithms for spanner constructions: A_1, an algorithm suitable for graphs with small aspect ratio, and A_2, an algorithm that returns a sparse spanner, but with potentially unbounded lightness. We consider a partition of the edges into groups according to their weights. For treating most of the groups we use exponentially growing clusters, partitioning

the edges according to weight. Each such group has bounded aspect ratio, and thus we can use A_1. Due to the exponential growth rate, we show that the contribution of all the different groups is converging. Thus only the first group is significant. However, with this approach we need a special treatment for edges of small weight. This is, as using the previous approach, the number of clusters needed to treat light edges is unbounded. Nevertheless, these edges have small impact on the lightness and we may thus use algorithm A_2, which ignores this property.

The main work in proving Theorems 2 to 4 is in designing the algorithms A_1 and A_2 described briefly below.

Approximate greedy spanner. The major time consuming ingredient of the greedy spanner algorithm is its shortest path computations. By instead considering approximate shortest path computations we significantly speed this process up. We are the first to apply this idea on general graphs, while it has previously been applied by [12, 19] on particular graph families. Specifically, we consider the following algorithm: given some parameters $t < t'$, initialize $H \leftarrow \emptyset$ and consider the edges $(u, v) \in E$ according to increasing order of weight. If $d_H(u, v) > t' \cdot w(u, v)$ the algorithm is obliged to add (u, v) to H. If $d_H(u, v) < t \cdot w(u, v)$, the algorithm is forbidden to add (u, v) to H. Otherwise, the algorithm is free to include the edge or not. As a result, we will get spanner with stretch t', which has the same lightness and sparsity guarantees of the greedy t-spanner. Note however, that the resulting spanner is not necessarily a subgraph of any greedy spanner.

We obtain both Theorem 2 and Theorem 4 using this approach via an incremental approximate distance oracle. It is important to note that the edges inserted into H using this approach depend on the answers to the distance queries. It is therefore not possible to use approaches that do not work against an *adaptive adversary* such as the result of Roditty and Zwick [29], which is based on random sampling. Furthermore, this is the case even if we allow the spanner construction itself to be randomized. In order to obtain Theorem 2, we use our previously described framework coupled with the "approximately greedy spanner" using an incremental $(1 + \varepsilon)$-approximate distance oracle of Henzinger et al. [21]. For Theorem 4, we present a novel incremental approximate distance oracle, which is described below. This is the main technical part of the paper and we believe that it may be of independent interest.

Deterministic distance oracle. The main technical contribution of the paper and key ingredient in proving Theorem 4 is our new deterministic incremental approximate distance oracle of Theorem 6. The oracle supports approximate distance queries of pairs within some distance threshold, d. In particular, we may set d to be some function of the stretch of the spanner in Theorem 4. Similar to previous work on distance oracles, we have some parameter, k, and maintain k sets of nodes $\emptyset = A_{k-1} \subseteq \ldots \subseteq A_0 = V$, and for each $u \in A_i$ we maintain a ball of radius $r \leq d_i$. Here, d_i is a distance threshold depending on the parameter d and which set A_i we are considering, and r is chosen such that the total degree of nodes in the ball of radius r from u is relatively small. The implementation of each ball can be thought of as an incremental Even-Shiloach tree. The set A_{i+1} is then chosen as a maximal set of nodes with disjoint balls. Here we use the fact that the vertices in A_{i+1} are centers of disjoint balls in A_i to argue that A_{i+1} is much smaller than A_i. The decrease in size of A_{i+1} pays for an increase in the maximum ball radius d_i at each level. The ball of a node u may grow in size during edge insertions. In this case, we freeze the ball associated with u, shrink the radius r associated with u, and create a new ball with the new radius. Thus, for each A_i we end up with $O(\log d)$ different radii for which we pick a maximal set of nodes with disjoint balls. For

each node $u_i \in A_i$ we may then associate a node $u_{i+1} \in A_{i+1}$ whose ball intersects with u_i's. We use these associated nodes in the query to ensure that the path distance we find is not "too far away" from the actual shortest path distance. Consider a query pair (u, v). Then the query algorithm iteratively finds a sequence of vertices $u = u_0 \in A_0, u_1 \in A_1, ..., u_i \in A_i$; d_i is picked such that if v is not in the ball centered at u_i with radius d_i then the shortest path distance between u and v is at least d and the algorithm outputs ∞. Otherwise, the algorithm uses the shortest path distances stored in the balls that it encounters to output the weight of a uv-path $(u = u_0) \rightsquigarrow u_1 \rightsquigarrow \ldots \rightsquigarrow u_i \rightsquigarrow v$ as an approximation of the shortest path distance between u and v.

Almost linear spanner. Chechik and Wulff-Nilsen [11] implicitly used our general framework, but used the (time consuming) greedy spanner both as their A_2 component and as a sub-routine in A_1. We show an efficient alternative to the algorithm of [11]. For the A_2 component we provide a novel sparse spanner construction (Theorem 7, see paragraph below). For A_1, we perform a hierarchical clustering, while avoiding the costly exact diameter computations used in [11]. Finally, we replace the greedy spanner used as a sub-routine of [11] by an efficient spanner that exploits bounded aspect ratio (see Lemma 13). This spanner can be seen as a careful adaptation of Elkin and Solomon [15] analyzed in the case of bounded aspect ratio. The idea here is (again) a hierarchical partitioning of the vertices into clusters of exponentially increasing size. However, here the growth rate is only $(1 + \varepsilon)$. Upon each clustering we construct a super graph with clusters as vertices and graph edges from the corresponding weight scale as inter-cluster edges. To decide which edges in each scale add to our spanner, we execute the extremely efficient spanner of Halperin and Zwick [20] for unweighted graphs.

Linear time sparse spanner. As mentioned above we provide a novel sparse spanner construction as a building block in proving Theorem 3. Our construction is based on partitioning edges into $O_\varepsilon(\log k)$ "well separated" sets $E_1, E_2, \ldots$, such that the ratio between $w(e)$ and $w(e')$ for edges $e, e' \in E_i$ is either a constant or at least k. This idea was previously employed by Elkin and Neiman [13] based on [23]. For these well-separated graphs, Elkin and Neiman used an involved clustering scheme based on growing clusters according to exponential distribution, and showed that the expected number of inter-cluster edges, in all levels combined, is small enough. We provide a linear time *deterministic* algorithm with an arguably simpler clustering scheme. Our clustering is based upon the clusters defined implicitly by the spanner for unweighted graphs of Halperin and Zwick [20]. In particular, we introduce a charging scheme, such that each edge added to our spanner is either paid for by a large cluster with many coins, or significantly contributing to reduce the number of clusters in the following level.

4 A framework for creating light spanners efficiently

In this section we describe a general framework for creating spanners, which we will use to prove our main results. The framework is inspired by a standard clustering approach (see e.g. [15] and [11]). The spanner framework takes as input two spanner algorithms for restricted graph classes, A_1 and A_2, and produces a spanner algorithm for general graphs. The algorithm A_1 works for graphs with unit weight MST edges and small aspect ratio, and A_2 creates a small spanner with no guarantee for the lightness. The main work in showing Theorems 2, 3, and 4 is to construct the algorithms, A_1 and A_2, that go into Lemma 8 below. The framework is described in the following lemma. The proof appears in the full version [1].

▶ **Lemma 8.** *Let $G = (V, E)$ be a weighted graph with n nodes and m edges and let $k > 0$ be an integer, $g > 1$ a fixed parameters and $\varepsilon > 0$. Assume that we are given two spanner construction algorithms A_1 and A_2 with the following properties:*

- *A_1 computes a spanner of stretch $f_1(k)$, size $O_\varepsilon(s_1(k)\cdot n^{1+1/k})$ and lightness $O_\varepsilon(l_1(k)\cdot n^{1/k})$ in time $T_1(n, m, k)$ when given a graph with maximum weight g^k, where all MST edges have weight 1. Moreover, T_1 has the property that $\sum_{i=0}^{\infty} T_1\left(\frac{n}{g^{ik}}, m_i, k\right) = O\left(T_1(n, m, k)\right)$, where $\sum_i m_i = m + O(n)$.*
- *A_2 computes a spanner of stretch $f_2(k)$ and size $O_\varepsilon(s_2(k) \cdot n^{1+1/k})$ in time $T_2(n, m, k)$.*

Then one can compute a spanner of stretch $\max((1 + \varepsilon)f_1(k), f_2(k))$, size $O_\varepsilon((s_1(k) + s_2(k))n^{1+1/k})$, and lightness $O_\varepsilon((l_1(k) + s_2(k)) \cdot n^{1/k})$ in time $O(T_1(n, m, k) + T_2(n, m, k) + m + n \log n)$.

5 Efficient approximate greedy spanner

In this section we will show how to efficiently implement algorithms A_1 and A_2 of Lemma 8 in order to obtain Theorems 2 and 4. We do this by implementing an "approximate-greedy" spanner, which uses an incremental approximate distance oracle to determine whether an edge should be added to the spanner or not.

We first prove Theorem 4 and then show in Section 5.2 how to modify the algorithm to give Theorem 2. We will use Theorem 6 as a main building block, but defer the proof of this theorem to Section 8. Our A_1 is obtained by the following lemma giving stretch $O(k)$ and optimal size $O(n^{1+1/k})$ and lightness $O(n^{1/k})$ for small weights.

▶ **Lemma 9.** *Let $G = (V, E, w)$ be an undirected graph with $m = |E|$ and $n = |V|$ and integer edge weights bounded from above by W. Let k be a positive integer and let $\varepsilon' > 0$ be a constant. Then one can deterministically construct an $O(k)$-spanner of G with size $O(n^{1+1/k})$ and lightness $O(n^{1/k})$ in time $O\left(m + kWn^{1+1/k+\varepsilon'}\right)$.*

We note that Lemma 9 above requires integer edge weights, but we may obtain this by simply rounding up the weight of each edge losing at most a factor of 2 in the stretch. Alternatively we can use the approach of Lemma 12 in Section 5.2 to reduce this factor of 2 to $(1 + \varepsilon)$.

Our A_2 will be obtained by the following lemma, which is essentially a modified implementation of Lemma 9.

▶ **Lemma 10.** *Let $G = (V, E, w)$ be an edge-weighted graph with $m = |E|$ and $n = |V|$. Let k be a positive integer and let $\varepsilon' > 0$ be a constant. Then one can deterministically construct an $O(k)$-spanner of G with size $O(n^{1+1/k})$ in time $O\left(m + kn^{1+1/k+\varepsilon'}\right)$.*

Combining Lemma 8 of Section 4 with Lemmas 9 and 10 above immediately gives us a spanner with stretch $O(k)$, size $O(n^{1+1/k})$ and lightness $O(n^{1/k})$ in time $O(m + n^{1+1/k+\varepsilon''})$ for any constant $\varepsilon'' > 0$. This is true because we may assume that $k \leq \gamma \log n$ for any constant $\gamma > 0$, and thus by picking γ and ε' accordingly we have that the running time given by Lemma 8 can be bounded by

$$O\left(m + kWn^{1+1/k+\varepsilon'} + kn^{1+1/k+\varepsilon'}\right) = O\left(m + kg^k n^{1+1/k+\varepsilon'}\right) = O\left(m + n^{1+1/k+\varepsilon''}\right) .$$

5.1 Details of the almost-greedy spanner

Set $\varepsilon = 1$ [2]. Our algorithm for Lemma 9 is described below in Algorithm 1. It computes a spanner of stretch $c_1(1+\varepsilon)(2k-1)$, where $c_1 = O(1)$ is the stretch of our incremental approximate distance oracle in Theorem 6. Let $t = c_1(1+\varepsilon)(2k-1)$ throughout the section. The proof of Lemma 9 is postponed to the full version [1].

Algorithm 1 `Approximate-Greedy`.

input : Graph $G = (V, E, w)$, Parameters ε, k
output : Spanner H

```
Create H = (V, ∅)
Initialize incremental distance oracle (Theorem 6) on H with d = t · W
for (u, v) ∈ E in non-decreasing order do
    if d̂_H(u, v) > t · w(u, v) then
        Add (u, v) to H
return H
```

Next, we sketch the proof Lemma 10, by explaining how to modify the proof of Lemma 9.

Proof of Lemma 10. Recall that c_1 is defined as the constant stretch provided by Theorem 6. We use Algorithm 1 with the following modifications: (1) we pick $d = c_1(2k-1)$, (2) when adding an edge to the distance oracle we add it as an unweighted edge, (3) we add an edge if its endpoints are not already connected by a path of at most d edges according to the approximate distance oracle.

The stretch of the spanner follows by the same stretch argument as in Lemma 9 and the fact that we consider the edges in non-decreasing order. To see that the size of the spanner is $O(n^{1+1/k})$ consider an edge (u, v) added to H by the modified algorithm. Since (u, v) was added to H we know that the distance estimate was at least $c_1(2k-1)$. It thus follows from Theorem 6 that u and v have distance at least $2k$ in H and therefore H has girth at least $2k+1$. It now follows that H has $O(n^{1+1/k})$ edges by a standard argument. The running time of this modified algorithm follows directly from Theorem 6. ◀

5.2 Near-quadratic time implementation

The construction of the previous section used our result from Theorem 6 to efficiently construct a spanner losing a constant factor exponential in $1/\varepsilon$ in the stretch. We may instead use the seminal result of Even and Shiloach [17] to obtain the same result with stretch $(1+\varepsilon)(2k-1)$ at the cost of a slower running time as detailed in Theorem 2. Below is described a version of the result of [17] which we will use.

▶ **Theorem 11** ([17]). *There exists a deterministic incremental APSP data structure for graphs with integer edge weights, which answers distance queries within a given threshold d in $O(1)$ time and has total update time $O(mnd)$.*

Here, the threshold means that if the distance between two nodes is at most d, the data structure outputs the exact distance and otherwise it outputs ∞ (or some other upper bound).

[2] In Section 5.2 we let $0 < \varepsilon < 1$ here to be arbitrary small parameter.

To obtain Theorem 2 we use the framework of Section 4. For the algorithm A_2 we may simply use the deterministic spanner construction of Roditty and Zwick [28] giving stretch $2k-1$ and size $O(n^{1+1/k})$ in time $O(kn^{2+1/k})$. For A_1 we will show the following lemma.

▶ **Lemma 12.** *Let $G=(V,E,w)$ be an undirected graph with $m=|E|$ and $n=|V|$, edge weights bounded from above by W and where all MST edges have weight 1. Let k be a positive integer. Then one can deterministically construct a $(1+\varepsilon)(2k-1)$-spanner of G with size $O_\varepsilon(n^{1+1/k})$ and lightness $O_\varepsilon(n^{1/k})$ in time $O_\varepsilon(m\log n + kWn^{2+1/k})$.*

Proof sketch. The final spanner will be a union of two spanners. Since Theorem 11 requires integer weights. We therefore need to treat edges with weight less than $1/\varepsilon$ separately. For these edges we use the algorithm of Roditty and Zwick [28] to produce a spanner with stretch $2k-1$, size $O(n^{1+1/k})$ and thus total weight $O(n^{1+1/k}/\varepsilon)$.

For the remaining edges with weight at least $1/\varepsilon$ we now round up the weight to the nearest integer incurring a stretch of at most a factor of $1+\varepsilon$. We now follow the approach of Algorithm 1 using the incremental APSP data structure of Theorem 11 and a threshold in line 4 of $(1+\varepsilon)(2k-1)\cdot w(u,v)$ instead. We use the distance threshold $d=(1+\varepsilon)(2k-1)\cdot W$.

The final spanner, H, is the union of the two spanners above. The stretch, size and lightness of the spanner follows immediately from the proof of Lemma 9. For the running time, we add in the additional time to sort the edges and query the distances to obtain a total running time of

$$O_\varepsilon(m\log n + d\cdot|E(H)|\cdot|V(H)|) = O_\varepsilon\Big(m\log n + kWn^{2+1/k}\Big)\ . \qquad \blacktriangleleft$$

Now, recall that $W=g^k$, where $k\le \log n$ and $g>1$ is a fixed parameter of our choice. By picking g such that $g^{2k}\le n^{\varepsilon'}$ we get a running time of $O(n^{2+1/k+\varepsilon'})$ for A_1. Theorem 2 now follows from Lemma 8.

6 Almost Linear Spanner

Our algorithm builds on the spanner of Chechik and Wulff-Nilsen [11]. Here we first describe their algorithm and then present the modifications. Chechik and Wulff-Nilsen implicitly used our general framework, and thus provide two different algorithms A_1^{CW} and A_2^{CW}. A_2^{CW} is simply the greedy spanner algorithm.

A_1^{CW} starts by partitioning the non-MST edges into k buckets, such that the ith bucket contains all edges with weight in $[g^{i-1},g^i)$. The algorithm is then split into k levels with the ith bucket being treated in the ith level. In the ith level, the vertices are partitioned into i-clusters, where the i-clusters refine the $(i-1)$-clusters. Each i-cluster has diameter $O(kg^i)$ and contains at least $\Omega(kg^i)$ vertices. This is similar to the (i,ε)-clusters in Section 4 with the modification of having two types of clusters, *heavy* and *light*. A cluster is heavy if it has many incident i-level edges and light otherwise. For a light cluster, we add all the incident i-level edges to the spanner directly. For the heavy clusters, Chechik and Wulff-Nilsen [11] create a special auxiliary cluster graph and run the greedy spanner on this to decide which edges should be added.

To bound the lightness of the constructed spanner, they show that each time a heavy cluster is constructed the number of clusters in the next level is reduced significantly. Then, using a clever potential function, they show that the contribution of all the greedy spanners is bounded. It is interesting to note, that in order to bound the weight of a single greedy spanner, they use the analysis of [14]. Implicitly, [14] showed that on graphs with $O(\text{poly}(k))$ aspect ratio, the greedy $(1+\varepsilon)(2k-1)$-spanner has $O_\varepsilon(n^{1/k})$ lightness and $O(n^{1+1/k})$ edges.

There are three time-consuming parts in [11]: 1) The clustering procedure iteratively grows the i-clusters as the union of several $(i-1)$-clusters, but uses expensive exact diameter calculations in the original graph. 2) They employ the greedy spanner several times as a subroutine during A_1^{cw} for graphs with $O(poly(k))$ aspect ratio. 3) They use the greedy spanner as A_2^{cw}.

In order to handle 1) above we will grow clusters purely based on the number of nodes in the $(i-1)$-clusters (in similar manner to (i,ε)-clusters), thus making the clustering much more efficient without losing anything significant in the analysis. To handle 2) We will use the following lemma in place of the greedy spanner.

▶ **Lemma 13.** *Given a weighted undirected graph $G=(V,E,w)$ with m edges and n vertices, a positive integer k, $\epsilon>0$, such that all the weights are within $[a, a\cdot\Delta)$, and the MST have weight $O(na)$. One can deterministically construct a $(2k-1)(1+\varepsilon)$-spanner of G with $O_\epsilon(n^{1+\frac{1}{k}})$ edges and lightness $O_\epsilon\left(n^{\frac{1}{k}}\cdot\log(\Delta)\right)$ in time $O(m+n\log n))$.*

The core of Lemma 13 already appears in [15], while here we analyze it for the special case where the aspect ratio is bounded by Δ. The main ingredient is an efficient spanner construction by Halperin and Zwick [20] for unweighted graphs. The proof of Lemma 13 is deferred to the full version [1]. Replacing the greedy spanner by Lemma 13 above is the sole reason for the additional $\log k$ factor in the lightness of Theorem 3.

Imitating the analysis of [11] with the modified ingredients, we are able to prove the following lemma, which we will use as A_1 in our framework.

▶ **Lemma 14.** *Given a weighted undirected graph $G=(V,E,w)$ with m edges and n vertices, a positive integer $k\geq 640$, and $\epsilon>0$, such that all MST edges have unit weight, and all weights bounded by g^k, one can deterministically construct a $(2k-1)(1+\varepsilon)$-spanner of G with $O_\varepsilon(n^{1+1/k})$ edges and lightness $O_\epsilon\left(\log k\cdot n^{\frac{1}{k}}\right)$ in time $O(m+nk))$.*

To address the third time-consuming part we instead use the algorithm of Theorem 7 as A_2. Replacing the greedy algorithm by Theorem 7 is the sole reason for the additional $\log k$ factor in the sparsity of Theorem 3.

Combining Lemma 14, Theorem 7 and Lemma 8 we get Theorem 3. The proof of Lemma 14 deferred to the full version [1].

7 Proof of Theorem 7

The basic idea in the algorithm of Theorem 7, is to partition the edges E of G into $O_\varepsilon(\log k)$ sets $E_1, E_2, \ldots$, such that the edges in E_i are "well separated". That is, for every $e, e' \in E_i$, the ratio between $w(e)$ and $w(e')$ is either a constant or at least k. By hierarchical execution of a modified version of [20], with appropriate clustering, we show how to efficiently construct a spanner of size $O(n^{1+1/k})$ for each such "well separated" graph. Thus, taking the union of these spanners, Theorem 7 follows. The details appear in the full version [1].

8 Deterministic Incremental Distance Oracles for Small Distances

In this section, we present a deterministic incremental approximate distance oracle which can answer approximate distance queries between vertex pairs whose actual distance is below some threshold parameter d. This oracle will give us Theorem 6 and finish the proof of Theorem 4. In fact, we will show the following more general result. Theorem 6 follows directly by setting $k=1/\varepsilon$ in the theorem below.

▶ **Theorem 15.** *Let $G = (V, E)$ be an n-vertex undirected graph that undergoes a series of edge insertions. Let G have positive integer edge weights and set $E = \emptyset$ initially. Let $\varepsilon > 0$ and positive integers k and d be given. Then a deterministic approximate distance oracle for G can be maintained under any sequence of operations consisting of edge insertions and approximate distance queries. Its total update time is $O_\varepsilon(m^{1+1/k}(3+\varepsilon)^{k-1}d(k+\log d)\log n)$ where m is the total number of edge insertions; the value of m does not need to be specified to the oracle in advance. Given a query vertex pair (u, v), the oracle outputs in $O(k \log n)$ time an approximate distance $\tilde{d}(u, v)$ such that $\tilde{d}(u, v) \geq d(u, v)$ and such that if $d(u, v) \leq d$ then $\tilde{d}(u, v) \leq (2(3+\varepsilon)^{k-1} - 1)d(u, v)$.*

As discussed in Section 3, a main advantage of our oracle is that, unlike, e.g., the incremental oracle of Roditty and Zwick [29], it works against an adaptive adversary. Hence, the sequence of edge insertions does not need to be fixed in advance and we allow the answer to a distance query to affect the future sequence of insertions. This is crucial for our application since the sequence of edges inserted into our approximate greedy spanner depends on the answers to the distance queries.

We assume in the following that $m \geq n$; if this is not the case, we simply extend the sequence of updates with $n - m$ dummy updates. We will present an oracle satisfying Theorem 15 except that we require it to be given m in advance. An oracle without this requirement can be obtained from this as follows. Initially, an oracle is set up with $m = n$. Whenever the number of edge insertions exceeds m, m is doubled and a new oracle with this new value of m replaces the old oracle and the sequence of edge insertions for the old oracle are applied to the new oracle. By a geometric sums argument, the total update time for the final oracle dominates the time for all the previous oracles. Hence, presenting an oracle that knows m in advance suffices to show the theorem.

Before describing our oracle, we need some definitions and notation. For an edge-weighted tree T rooted at a vertex u, let $d_T(v)$ denote the distance from u to v in T, where $d_T(v) = \infty$ if $v \notin V(T)$. Let $r(T) = \max_{v \in V(T)} d_T(v)$. Given a graph H and $W \subseteq V(H)$, we let $\deg_H(W) = \sum_{v \in W} \deg_H(v)$ and given a subgraph S of H, we let $\deg_H(S) = \deg_H(V(S))$. For a vertex u in an edge-weighted graph H and a value $r \geq 0$, we let $B_H(u, r)$ denote the ball with center u and radius r in H, i.e., $B_H(u, r) = \{v \in V(H) | d_H(u, v) \leq r\}$. When H is clear from context, we simply write $B(u, r)$.

We use a superscript (t) to denote a dynamic object (such as a graph or edge set) or variable just after the t'th edge insertion where $t = 0$ refers to the object prior to the first insertion and $t = m$ refers to the object after the final insertion. For instance, we refer to G just after the t'th update as $G^{(t)}$.

In the following, let ε, k, and d be the values and let $G = (V, E)$ be the dynamic graph of Theorem 15. For each $i \in \{0, \ldots, k-1\}$, define $m_i = 2m^{(i+1)/k}$ and let d_i be the smallest power of $(1+\varepsilon)$ of value at least $(3+2\varepsilon)^i d$. For each $u \in V$ and each $t \in \{0, \ldots, m\}$, let $d_i^{(t)}(u)$ be the largest power of $(1+\varepsilon)$ of value at most d_i such that $\deg_{G^{(t)}}(B^{(t)}(u, d_i^{(t)}(u))) \leq m_i$. We let $B_i^{(t)}(u) = B^{(t)}(u, d_i^{(t)}(u))$ and let $T_i^{(t)}(u)$ be a shortest path tree from u in $B_i^{(t)}(u)$. Note that $T_i^{(t)}(u)$ need not be uniquely defined; in the following, when we say that a tree is equal to $T_i^{(t)}$, it means that the tree is equal to some shortest path tree from u in $B_i^{(t)}(u)$.

The data structure in the following lemma will be used as black box in our distance oracle. One of its tasks is to efficiently maintain trees $T_i^{(t)}(u)$. The proof of Lemma 16 is deferred to the full version [1].

▶ **Lemma 16.** *Let $U \subseteq V$ be a dynamic set with $U^{(0)} = \emptyset$ and let $i \in \{0, \ldots, k-1\}$ be given. There is a deterministic dynamic data structure which supports any sequence of update operations, each of which is one of the following types:*

`Insert-Edge`(u, v)*: this operation is applied whenever an edge* (u, v) *is inserted into* E*,*
`Insert-Vertex`(u)*: inserts vertex* u *into* U*.*
Let $t_{\max}$ *denote the total number of operations and for each vertex* u *inserted into* U*, let* t_u *denote the update in which this happens. The data structure outputs in each update* $t \in \{1, \ldots, t_{\max}\}$ *a (possibly empty) set of trees* $\overline{T}_i^{(t)}(u)$ *rooted at* u *for each* $u \in U^{(t)}$ *satisfying either* $t > t_u$ *and* $d_i^{(t)}(u) < d_i^{(t-1)}(u)$ *or* $t = t_u$ *and* $d_i^{(t)}(u) < d_i$*. For each such tree* $\overline{T}_i^{(t)}(u)$*,* $r(\overline{T}_i^{(t)}(u)) \leq (1+\varepsilon) d_i^{(t)}(u) \leq d_i$ *and* $\deg_{G^{(t)}}(\overline{T}_i^{(t)}(u)) > m_i$*. Total update time is* $O(m) + O_\varepsilon(|U^{(t_{\max})}| m_i d_i \log n)$*.*

At any point, the data structure supports in $O(1)$ *time a query for the value* $d_i^{(t)}(u)$ *and in* $O(\log n)$ *time a query for the value* $d_{T_i(u)}(v)$ *and for whether* $v \in V(T_i(u))$*, for any query vertices* $u \in U$ *and* $v \in V$*.*

The construction and analysis of the distance oracle appear in the full version [1].

References

1 Stephen Alstrup, Søren Dahlgaard, Arnold Filtser, Morten Stöckel, and Christian Wulff-Nilsen. Constructing Light Spanners Deterministically in Near-Linear Time. *CoRR*, abs/1709.01960, 2017. `arXiv:1709.01960`.

2 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. `doi:10.1007/BF02189308`.

3 Baruch Awerbuch. Complexity of Network Synchronization. *J. ACM*, 32(4):804–823, October 1985. `doi:10.1145/4221.4227`.

4 Yair Bartal, Arnold Filtser, and Ofer Neiman. On notions of distortion and an almost minimum spanning tree with constant average distortion. *Journal of Computer and System Sciences*, 2019. `doi:10.1016/j.jcss.2019.04.006`.

5 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. See also ICALP'03.

6 Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proc. 48th ACM Symposium on Theory of Computing (STOC)*, pages 398–411, 2016.

7 Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New Sparseness Results on Graph Spanners. In *Proc. 8th ACM Symposium on Computational Geometry (SoCG)*, pages 192–201, 1992.

8 Shiri Chechik. Compact Routing Schemes with Improved Stretch. In *Proc.ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–41, 2013.

9 Shiri Chechik. Approximate Distance Oracles with Constant Query Time. In *Proc. 46th ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 2014.

10 Shiri Chechik. Approximate Distance Oracles with Improved Bounds. In *Proc. 47th ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2015.

11 Shiri Chechik and Christian Wulff-Nilsen. Near-Optimal Light Spanners. *ACM Trans. Algorithms*, 14(3):33:1–33:15, 2018. `doi:10.1145/3199607`.

12 Gautam Das and Giri Narasimhan. A Fast Algorithm for Constructing Sparse Euclidean Spanners. *Int. J. Comput. Geometry Appl.*, 7(4):297–315, 1997. `doi:10.1142/S0218195997000193`.

13 Michael Elkin and Ofer Neiman. Efficient Algorithms for Constructing Very Sparse Spanners and Emulators. *ACM Trans. Algorithms*, 15(1):4:1–4:29, 2019. `doi:10.1145/3274651`.

14 Michael Elkin, Ofer Neiman, and Shay Solomon. Light Spanners. *SIAM J. Discrete Math.*, 29(3):1312–1321, 2015. `doi:10.1137/140979538`.

15 Michael Elkin and Shay Solomon. Fast Constructions of Lightweight Spanners for General Graphs. *ACM Transactions on Algorithms*, 12(3):29:1–29:21, 2016. See also SODA'13.

16 Paul Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications*, pages 29–36, 1964.

17 Shimon Even and Yossi Shiloach. An On-Line Edge-Deletion Problem. *J. ACM*, 28(1):1–4, 1981.

18 Arthur M. Farley, Andrzej Proskurowski, Daniel Zappala, and Kurt Windisch. Spanners and message distribution in networks. *Discrete Applied Mathematics*, 137(2):159–171, 2004.

19 Arnold Filtser and Shay Solomon. The Greedy Spanner is Existentially Optimal. In *Proc. of the 2016 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 9–17, 2016.

20 Shay Halperin and Uri Zwick. Linear time deterministic algorithm for computing spanners for unweighted graphs, 1996.

21 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Dynamic Approximate All-Pairs Shortest Paths: Breaking the O(mn) Barrier and Derandomization. *SIAM Journal on Computing*, 45(3):947–1006, 2016. See also FOCS'13.

22 Ioannis Koutis and Shen Chen Xu. Simple Parallel and Distributed Algorithms for Spectral Graph Sparsification. *TOPC*, 3(2):14:1–14:14, 2016. doi:10.1145/2948062.

23 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved Parallel Algorithms for Spanners and Hopsets. In *Proc. 27th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 192–201, 2015.

24 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

25 David Peleg and Jeffrey D. Ullman. An Optimal Synchronizer for the Hypercube. In *Proc. 6thACM Symposium on Principles of Distributed Computing (PODC)*, pages 77–85, 1987.

26 David Peleg and Eli Upfal. A Tradeoff Between Space and Efficiency for Routing Tables. In *Proc 20th ACM Symposium on Theory of Computing (STOC)*, pages 43–52, 1988.

27 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic Constructions of Approximate Distance Oracles and Spanners. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 261–272, 2005.

28 Liam Roditty and Uri Zwick. On Dynamic Shortest Paths Problems. *Algorithmica*, 61(2):389–401, 2011.

29 Liam Roditty and Uri Zwick. Dynamic Approximate All-Pairs Shortest Paths in Undirected Graphs. *SIAM Journal on Computing*, 41(3):670–683, 2012. See also FOCS'04.

30 Mikkel Thorup and Uri Zwick. Compact Routing Schemes. In *Proc. 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

31 Mikkel Thorup and Uri Zwick. Approximate Distance Oracles. *Journal of the ACM*, 52(1):1–24, January 2005. See also STOC'01. doi:10.1145/1044731.1044732.

32 Christian Wulff-Nilsen. Approximate Distance Oracles with Improved Preprocessing Time. In *Proc. 23rd ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–208, 2012.

33 Christian Wulff-Nilsen. Approximate Distance Oracles with Improved Query Time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 539–549, 2013.

34 Christian Wulff-Nilsen. Fully-Dynamic Minimum Spanning Forest with Improved Worst-Case Update Time. *CoRR*, abs/1611.02864, 2016. To appear at STOC'17. arXiv:1611.02864.