Distrib. Comput.
DOI 10.1007/s00446-009-0091-7

# Distributed algorithms for ultrasparse spanners and linear size skeletons

**Seth Pettie**

Received: 24 April 2008 / Accepted: 7 October 2009
© Springer-Verlag 2009

**Abstract** We present efficient algorithms for computing very sparse low distortion spanners in distributed networks and prove some non-trivial lower bounds on the tradeoff between time, sparseness, and distortion. All of our algorithms assume a synchronized distributed network, where relatively short messages may be communicated in each time step. Our first result is a fast distributed algorithm for finding an $O(2^{\log^* n}\log n)$-spanner with size $O(n)$. Besides being nearly optimal in time and distortion, this algorithm appears to be the first that constructs an $O(n)$-size skeleton without requiring unbounded length messages or time proportional to the diameter of the network. Our second result is a new class of efficiently constructible $(\alpha, \beta)$-spanners called *Fibonacci spanners* whose distortion improves with the distance being approximated. At their sparsest Fibonacci spanners can have nearly linear size, namely $O(n(\log\log n)^{\phi})$, where $\phi = (1+\sqrt{5})/2$ is the golden ratio. As the distance increases the multiplicative distortion of a Fibonacci spanner passes through four discrete stages, moving from logarithmic to log-logarithmic, then into a period where it is constant, tending to 3, followed by another period tending to 1. On the lower bound side we prove that many recent sequential spanner constructions have no efficient counterparts in distributed networks, even if the desired distortion only needs to be achieved on the average or for a tiny fraction of the vertices. In particular, any distance preservers, purely additive spanners, or spanners with sublinear additive distortion must either be very dense, slow to construct, or have very weak guarantees on distortion.

This work is supported by NSF CAREER grant no. CCF-0746673.

S. Pettie (✉)
Department of Electrical Engineering and Computer Science,
University of Michigan, Ann Arbor, MI, USA
e-mail: pettie@umich.edu

## 1 Introduction

Many applications in distributed computation use, explicitly or otherwise, a sparse substitute for the underlying communications network that retains the character of the original network. At the very least the substitute should preserve connectivity [18]. In many situations one needs the substitute to preserve the distance metric of the network up to some tolerable distortion. This type of requirement arises in the construction of "synchronizers" [30], compact routing tables with small stretch [1,2,37], distance labeling schemes [38], and communication-efficient approximate shortest path algorithms [19,24]. Following [23] we define an $(\alpha, \beta)$-spanner of an undirected, unweighted graph $G = (V, E)$ to be a subgraph $S \subseteq E$ such that

$$\delta_S(u, v) \le \alpha \cdot \delta(u, v) + \beta$$

for any $u, v \in V$, where $\delta_S$ is the distance within $S$. An $(\alpha, 0)$-spanner is also called an $\alpha$-spanner and a $(1, \beta)$-spanner an *additive* $\beta$-spanner. There is clearly a tradeoff between the sparseness of $S$ and the distortion and, one would presume, a tradeoff between distortion and the computational resources required to construct the spanner. At present we have only a crude understanding of the possible sparseness-distortion tradeoffs and the resources required to find good spanners. Assuming Erdős's *girth conjecture* is true (see [25,38,40]) it is known that any $(\alpha, \beta)$-spanner with $\alpha + \beta \le 2k$ has size $\Omega(n^{1+1/k})$. Moreover, Woodruff [41] showed that, independent of the girth conjecture, any additive $(2k-2)$-spanner has size $\Omega(k^{-1}n^{1+1/k})$. On the upper bound side, there are several sequential algorithms [4,10,35,36,38] for constructing $(2k-1)$-spanners with size $O(n^{1+1/k})$, for any fixed $k \ge 2$, but only two known constructions of additive $O(1)$-spanners. Aingworth et al. [3] (see also [7,17,22,33,35,39]) constructed additive

2-spanners with size $O(n^{3/2})$ and Baswana et al. [7] constructed additive 6-spanners with size $O(n^{4/3})$. All known sparser additive spanners [7,33] have additive distortion polynomial in $n$.

Besides the purely additive and purely multiplicative extremes, $(\alpha, \beta)$-spanners come in two varieties. In one class, [7,19,23] $\alpha$ and $\beta$ are parameters of the construction and essentially fixed. In contrast, some recent constructions [33,39] give spanners that are $(\alpha, \beta)$-spanners for a whole suite of $\alpha, \beta$ pairs, so one must optimize one's choice of $\alpha$ and $\beta$ as a function of the distance being approximated. For example, the spanners in [33] have size slightly less than $O(kn^{1+(3/4)^{k+3}})$ and guarantee that if $\delta(u, v) = d > k^k$ then in the spanner $S$, $\delta_S(u, v) = d + O(kd^{1-1/k})$. We call spanners of this type [33,39] sublinear additive spanners.

## 1.1 Distributed constructions

At present the state-of-the-art in distributed spanners lags behind the best sequential constructions. There are, for example, no fast distributed algorithms for finding purely additive or sublinear additive spanners and the best distributed algorithms for finding purely multiplicative spanners [7,10,13,14,18] fall short of their sequential counterparts. Although there are distributed algorithms for $(1 + \epsilon, \beta)$-spanners [19,24], a close inspection of their parameters shows they are much weaker than their sequential counterparts [23,33,39]. Before discussing existing results and our contributions, we review the features of the computational model assumed in most prior work on spanners. See Peleg [30] for a more leisurely discussion of modeling considerations.

The graph for which we want a sparse spanner is identical to the underlying communications network, where each vertex holds a processor with a unique and arbitrary $O(\log n)$-bit identifier. The computation proceeds in synchronized time steps in which each processor can communicate one message to each neighbor in the graph. Any local computation performed is free. One measure of the feasibility of an algorithm is the number of time steps required to compute a spanner. We also separate algorithms by their *maximum* message length,[1] measured in units of $O(\log n)$ bits, and distinguish between randomized and deterministic algorithms.

## 1.2 Prior work and our contributions

All the results discussed below are summarized in Fig. 1. Although the focus of this paper is unweighted graphs, we point out that Baswana and Sen's [10] randomized algorithm for constructing $(2k - 1)$-spanners in *weighted* graphs is optimal in all respects (assuming the girth conjecture), save for a factor of $k$ in the spanner size. Without the aid of randomness, finding sparse spanners is a much more difficult problem. Derbel, Gavoille, Peleg, and Viennot, [15] recently gave a deterministic algorithm that finds, in optimal $O(k)$ time, a $(2k - 1)$-spanner with size $O(kn^{1+1/k})$. As a special case this algorithm finds an $O(\log n)$-spanner with size $O(n \log n)$. A previous deterministic algorithm of Derbel and Gavoille [13] is worse than [15] in most respects but it can return a sparse, $O(n \log \log n)$-size spanner in $O(n^{o(1)})$ time with just polylog$(n)$ distortion. A disadvantage of [13,15] and other deterministic algorithms is their use of unbounded length messages.

In relatively recent work Dubhashi et al. [18] presented algorithms that compute a linear size $O(\log n)$-spanner in $O(\log^3 n)$ time and a linear size subgraph (with no distortion guarantee) in $O(\log n)$ time. Their $O(\log n)$-spanner algorithm is randomized, assumes unbounded length messages, and places a considerable burden on the resources[2] of the processors. Our first result is a randomized algorithm for finding linear size spanners that is considerably more realistic than [18]. Using messages with length $O(\log^\epsilon n)$, our algorithm computes an $O(\epsilon^{-1} 2^{\log^* n} \log n)$-spanner with size $O(n)$ in $O(\epsilon^{-1} 2^{\log^* n} \log n)$ time steps. (Fig. 1 simplifies this expression by fixing $\epsilon = o(1) > 1/2^{\log^* n}$.) The time and distortion of our algorithm are very close to optimal. Moreover, this appears to be the first algorithm [29] with these properties that computes *any* $O(n)$-size subgraph.

The landscape of $(\alpha, \beta)$-spanners is more complex. Baswana et al.'s [7] $(k, k - 1)$-spanners save about a factor of 2 in the multiplicative stretch over [10] but the algorithm requires $\Omega(\sqrt{n})$-length messages. The $(1+\epsilon, \beta)$-spanners of Elkin and Zhang [24] (see also [19]) can be constructed with shorter messages but they can only get so sparse. At their sparsest they become $(1 + \epsilon, \beta)$-spanners with size $O(\beta n)$, where:

$$\beta = (\epsilon^{-1} t^2 \log n \log \log n)^{t \log \log n}$$

and $\tilde{O}(n^{1/t})$ is the message length. Both of the above algorithms are randomized. Derbel et al. [14–16] gave various deterministic constructions of $(1 + \epsilon, c)$-spanners (for $c \in \{2, 4, 6\}$) with different tradeoffs. All of these constructions use unbounded length messages; see Fig. 1 for details. Very recently Derbel et al. [16] presented *deterministic*

[1] In Peleg's terminology [30] the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models allow, respectively, unbounded length messages and unit length messages. However, we find it more revealing to pin down the precise message length.

[2] We excluded these considerations from our model. Nonetheless, we point out that the Dubhashi et al. [18] algorithm *could* end up communicating the topology of the whole graph to one vertex, which then runs an $\tilde{O}(n^2)$ time sequential algorithm [4,36].

| WEIGHTED GRAPHS — $\alpha$-SPANNERS | | | | | |
|---|---|---|---|---|---|
| $\alpha$ | Spanner Size | Time | Msg. Size | Rand.? | Notes |
| $2k-1$ | $O(kn^{1+1/k})$ | $O(k)$ | unit | rand. | [11,8,5] |
| UNWEIGHTED GRAPHS — $\alpha$-SPANNERS | | | | | |
| $\alpha$ | | | | | |
| $2k-1$ | $O(kn^{1+1/k})$ | $O(k)$ | unbounded | det. | [16] |
| $O(k^{\log_2 5})$ | $O(n^{1+1/k}\log k)$ | $n^{O(1/\sqrt{\log n})}$ | unbounded | det. | [14] |
| | | $O(\text{polylog}(n))$ | unbounded | rand. | |
| $O(\log n)$ | $O(n)$ | $O(\log^3 n)$ | unbounded | rand. | [19] |
| $2^{O(\log^* n)}\log n$ | $O(n)$ | $O(\log^{1+o(1)} n)$ | unit | rand. | **NEW** |
| UNWEIGHTED GRAPHS — $(\alpha, \beta)$-SPANNERS | | | | | |
| $(\alpha, \beta)$ | | | | | |
| $(k, k-1)$ | $O(kn^{1+1/k})$ | $O(k)$ | $O(n^{\frac{1}{2}+\frac{1}{2k}})$ | rand. | [8] |
| $(1+\epsilon, \beta_{k,\epsilon,t})$ | $O(\beta_{k,\epsilon,t}n^{1+1/k})$ | $O(\beta_{k,\epsilon,t})$ | $O(n^{\frac{1}{2k}+\frac{1}{t}})$ | rand. | [25], $\beta_{k,\epsilon,t} = O(\frac{t^2 k \log k}{\epsilon})^{t(\log k+1)}$ |
| $(1+\epsilon, \beta_{k,\epsilon})$ | $O(\beta_{k,\epsilon}n^{1+1/k})$ | $n^{O(1/\sqrt{\log n})}$ | unbounded | det. | [17], $\beta_{k,\epsilon} = O(\frac{\log k}{\epsilon})^{\log k}$ |
| $(1+\epsilon, \beta_{k,\epsilon})$ | $O(\beta_{k,\epsilon}n^{1+1/k})$ | $O(\beta_{k,\epsilon})$ | unbounded | det. | [17], $\beta_{k,\epsilon} = O(1/\epsilon)^{k-1}$ |
| $(1+\epsilon, 2)$ | $O(\epsilon^{-1}n^{3/2})$ | $O(\epsilon^{-1})$ | unbounded | det. | [16] |
| $(1+\epsilon, 4)$ | $O(n^{3/2})$ | $n^{O(1/\sqrt{\log n})}$ | unbounded | det. | [15] |
| | | $O(\epsilon^{-1}+\log n)$ | unbounded | rand. | |
| $(1+\epsilon, 6)$ | $O(\epsilon^{-1}n^{4/3})$ | $n^{O(1/\sqrt{\log n})}$ | unbounded | det. | [17] |
| $(O(\frac{\log n}{\log\log\log n}), 0)$ | $O(n(\frac{\log\log n}{\epsilon})^{\phi})$ | $O(\beta_{\epsilon,t})$ | $\tilde{O}(n^{1/t})$ | rand. | **NEW**, $\phi = \frac{1+\sqrt{5}}{2}$ |
| $(O(\log\log n), \beta_t)$ | | | | | $\beta_t \approx 2^t(\log n)^{\log_\phi 2} \approx 2^t(\log n)^{1.44}$ |
| $(3+\delta, \beta_{\delta,t})$ | | | | | $\beta_{\delta,t} \approx \beta_t^{-\log\delta}$ |
| $(1+\epsilon, \beta_{\epsilon,t})$ | | | | | $\beta_{\epsilon,t} \approx (\frac{\log_\phi \log n+t}{\epsilon})^{\log_\phi \log n+t}$ |

**Fig. 1** The state of the art in distributed spanner algorithms. The last *four lines* only illustrate the properties of our spanners at their sparsest level. See Sect. 4 for the general tradeoff between sparseness and distortion

algorithms for finding sparse $(1+\epsilon, \beta)$-spanners. However, their tradeoffs between sparseness, distortion, and construction time are significantly worse than the randomized algorithms presented here and in [24].

Our second result is a new class of subgraphs called *Fibonacci* spanners, so named due to the frequent occurrence of Fibonacci-like sequences in their analysis. They provide a spectrum of sparseness-distortion tradeoffs that are distinct improvements over previous distributed constructions [14–16,24] and are essentially the same as the best sequential constructions [23,32]. However, for simplicity we illustrate their properties only at their sparsest level, with size $O(n(\frac{\log\log n}{\epsilon})^{\phi})$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. For sufficiently distant vertices a Fibonacci spanner degenerates into a $(1+\epsilon, \beta)$-spanner, where $\beta \approx (\epsilon^{-1}\log_\phi \log n)^{\log_\phi \log n}$. However, for pairs of vertices that are closer than $\beta$ (arguably the more important case) Fibonacci spanners offer a multiplicative distortion that improves with distance in four discrete stages. For every pair of vertices it functions as an $O(\frac{\log n}{\log\log\log n})$-spanner.[3] For vertices at distance at least $(\log n)^{\log_\phi 2}$, which is roughly $\log^{1.44} n$, it functions as an $O(\log\log n)$-spanner, and for vertices at distance $(\log n)^{k\log_\phi 2}$ it functions as a $(3+O(2^{-k}))$-spanner. For distances greater than $\beta$ it becomes a $(1+\epsilon)$-spanner. When $t$ is taken into account (recall that $\tilde{O}(n^{1/t})$ is the maximum message length) our $\beta$ is $(\epsilon^{-1}(\log_\phi \log n+t))^{\log_\phi \log n+t}$, which compares favorably with the $\beta$ of Elkin and Zhang's [24] sparsest spanner, namely $\beta = (\epsilon^{-1}t^2\log n\log\log n)^{t\log\log n}$. In both algorithms the running time is $O(\beta)$ for their respective $\beta$s.

## 1.3 Lower bounds

Derbel et al. [14] posed the question of whether there are efficient distributed algorithms for finding additive $\beta$-spanners. We prove a general lower bound that can be applied to distributed algorithms for additive spanners, sublinear

[3] At their sparsest the $(1+\epsilon, \beta)$-spanners of Elkin and Zhang [24] function as $O(\log n)$-spanners.

additive spanners, and $(1+\epsilon,\beta)$-spanners. In particular, if we fix the desired size of the spanner at $n^{1+\rho}$, any additive $\beta$-spanner requires $\Omega(\sqrt{n^{1-\rho}/\beta})$ time to compute, any sublinear additive spanner with distortion of the form $d + O(d^{1-\epsilon})$ requires $\Omega(n^{\epsilon(1-\rho)/(1+\epsilon)})$ time to compute, and any $(1+\epsilon,\beta)$-spanner either requires $2\epsilon^{-1}$ time to compute or has $\beta = \Omega(\epsilon^2 n^{1-\rho})$. In independent work, Derbel et al. [15] gave significantly weaker lower bounds on constructing additive spanners and incomparable lower bounds on $(1+\epsilon,\beta)$-spanners. These lower bounds are discussed in more detail in Sect. 3.

### 1.4 Related work

Several algorithms have appeared in recent years that maintain spanners in dynamic graphs. Elkin [21] and Baswana [5] found algorithms for constructing sparse $(2k-1)$-spanners in an online streaming model, where edges arrive one at a time and the algorithm can only keep $\tilde{O}(n^{1+1/k})$ edges in memory. Baswana and Sarkar [8] and Elkin [21] developed fully dynamic algorithms for maintaining $(2k-1)$-spanners in the standard centralized model of computation and Elkin [20] showed that his algorithm can be adapted to a distributed setting. The update times (worst case, expected, and amortized) of the aforementioned algorithms often differ.

There is a large body of work on spanners for geometric graphs. Narasimhan and Smid's text on the subject [28] provides a nice survey of geometric spanners.

### 1.5 Organization

In Sect. 2, we present a distributed algorithm for computing linear size spanners and skeletons. In Sect. 3, we prove a three-way tradeoff between the sparseness, distortion, and construction time of any distributed spanner algorithm. In Sect. 4, we present the Fibonacci spanner and in Sect. 5, we conclude with some open problems.

## 2 Linear size spanners and skeletons

The standard method for obtaining a linear-size spanner or skeleton is to construct a subgraph that has girth $\Omega(\log n)$. (Girth is the length of the shortest cycle.) This is the strategy taken by the classical sequential algorithm of Althöfer et al. [4] and the distributed algorithm of Dubhashi et al. [18]. However, any algorithm taking this approach seems to require that vertices survey their whole $\Omega(\log n)$-neighborhood, which can require messages linear in the size of the graph. On the other hand, the Baswana–Sen spanners [10] guarantee sparseness without disallowing short cycles.

Our algorithm uses a distributed version of a clustering technique due to Baswana and Sen [10] and in some ways resembles Borůvka's parallel minimum spanning tree algorithm. It proceeds in a sequence of $\log^* n$ phases, where the goal of each phase is to reduce the number of vertices exponentially. Each phase begins with a contracted version of the original graph and finds a series of clusterings, each with larger radius than the last. The final clusters are then contracted to single vertices in preparation for the next phase. Contraction is an important technique to reduce the spanner size to linear but compounded contraction has a price in terms of distortion. If $G'$ is a contracted version of $G$ then we can talk sensibly about the radius of a cluster or vertex in $G'$ w.r.t. either $G$ or $G'$. Notice that if a cluster has $G'$-radius $r'$ and each of its constituent vertices has $G$-radius $r$, the $G$-radius of the whole cluster is $r' \cdot (2r+1) + r$, i.e., roughly $r'$ times the *diameter* of vertices in $G'$. This doubling effect leads to an additional $2^{\log^* n}$ factor in our distortion.

### 2.1 Notation and terminology

All logarithms are base 2 unless indicated otherwise. The original input graph is $G = (V, E)$. For vertex sets $V_1$ and $V_2$, let $V_1 \times V_2 = \{(v_1, v_2) \mid v_1 \in V_1, v_2 \in V_2\}$; if $V_1$ or $V_2$ are vertices they are treated as singleton sets. If $G' = (V', E')$ is a graph and $V'' \subseteq V'$, the graph $G' \circledcirc V''$ is derived by contracting $V''$, i.e., replacing $V''$ by a new vertex $v''$ that is adjacent to all $v$ for which $V'' \times v \cap E' \neq \emptyset$. The graph $G \circledcirc V''$ is *simple*, meaning we discard any loops and redundant edges. A *clustering* $\mathscr{C} = \{C_j\}_j$ is a set of disjoint subsets of vertices and it is *complete* if every vertex appears in some cluster. The graph $G' \circledcirc \mathscr{C}$ is derived by contracting each cluster in $\mathscr{C}$. If $v$ is a vertex or cluster in some contracted version of $G$ (the original graph), $\circledcirc^{-1}(v)$ refers to the set of vertices from $G$ represented by $v$. Let $\mathscr{C}$ be a clustering of a graph $G' = (V', E')$ appearing at some stage in our algorithm, where $G'$ was derived from $G$ by removing vertices and contracting sets of vertices. It is always the case that for each $C \in \mathscr{C}$, $\circledcirc^{-1}(C)$ is spanned by a tree of spanner edges centered at some vertex $c \in \circledcirc^{-1}(C)$. This tree is *not* necessarily a fragment of the shortest path tree from $c$. When our algorithm is working with $G'$ it may select an edge $(u, v) \in E'$ for inclusion in the spanner. *Selecting* $(u, v)$ is merely shorthand for selecting a single arbitrary edge $(u', v')$ among $\circledcirc^{-1}(u) \times \circledcirc^{-1}(v) \cap E$. When this point needs emphasizing we will use the notation $\circledcirc^{-1}(S')$, where $S'$ is a set of edges in a contracted graph, to refer to the edges in the original graph identified with $S'$.

**Observation 1** Let $\mathscr{C}$ be a clustering of $G$, $S$ be a set of radius-$r$ trees spanning clusters in $\mathscr{C}$, $\mathscr{C}'$ be a clustering of $G' = G \circledcirc \mathscr{C}$, and $S'$ be a set of radius-$r'$ trees (w.r.t. $G'$) spanning clusters in $\mathscr{C}'$. Then $S \cup \circledcirc^{-1}(S')$ consists of a set of radius-$(r'(2r+1)+r)$ trees w.r.t. $G$.

**Fig. 2** The **Expand** procedure takes a complete clustering $\mathscr{C}_{in}$ of $G_{in}$ and grows the radii of a select group of clusters from $\mathscr{C}_{in}$, yielding a new clustering $\mathscr{C}_{\text{out}}$

```
Expand($G_{in}, \mathscr{C}_{in}, p$)        {$G_{in} = (V_{in}, E_{in})$ is a graph, $\mathscr{C}_{in}$ a clustering of $V_{in}$, and $p \in [0, 1)$}
1.  $\mathscr{C}_{out} \leftarrow$ Sample each member of $\mathscr{C}_{in}$ with probability $p$
2.  Concurrently, for all $v \in V_{in}$:
3.      If $v$ is not in any cluster of $\mathscr{C}_{out}$ but adjacent to some $C \in \mathscr{C}_{out}$ then
4.          Include in the spanner some edge in $v \times C \cap E_{in}$
5.          $C \leftarrow C \cup \{v\}$
6.      If $v$ is neither in nor adjacent to any cluster in $\mathscr{C}_{out}$
7.          Include in the spanner some edge in $v \times C \cap E$, for all $C \in \mathscr{C}_{in}$ adjacent to $v$
8.          Mark $v$ dead
9.  $G_{out} \leftarrow G_{in}$ after removing all dead vertices and their adjacent edges
10. Return $(G_{out}, \mathscr{C}_{out})$
```

The procedure **Expand** (Fig. 2) takes a graph $G_{\text{in}}$, a *complete* clustering $\mathscr{C}_{\text{in}}$ of $G_{\text{in}}$ and a sampling probability $p \in [0, 1)$. It selects each cluster in $\mathscr{C}_{\text{in}}$ for inclusion in a new clustering $\mathscr{C}_{\text{out}}$ with probability $p$, then attempts to *expand* the radius of each cluster in $\mathscr{C}_{\text{out}}$. Suppose a vertex $v$ is in $C_0 \in \mathscr{C}_{\text{in}}$ and adjacent to $C_1, \ldots, C_q \in \mathscr{C}_{\text{in}}$. If one of these clusters, say $C_i$, is sampled for inclusion in $\mathscr{C}_{\text{out}}$, we add $v$ to the set $C_i$ and include some edge from $v$ to $C_i$ in the spanner. If, however, none of the adjacent clusters are sampled, we include in the spanner exactly one edge from $v$ to each of $C_1, \ldots, C_q$ then mark $v$ *dead*, thereby removing it from further consideration. Thus, if $G_{\text{out}}$ is the graph induced by the set of vertices not yet dead, $\mathscr{C}_{\text{out}}$ is a complete clustering of $G_{\text{out}}$. Note that vertices join clusters in $\mathscr{C}_{\text{out}}$ *simultaneously*, so if clusters in $\mathscr{C}_{\text{in}}$ have radius $r$ with respect to $G_{\text{in}}$, the clusters in $\mathscr{C}_{\text{out}}$ will have radius $r + 1$. Of course, $G_{\text{in}}$ is most likely a contracted version of the original graph so the true radius of these clusterings could be larger.

The growth of clusters and the sampling probabilities are guided by the sequence $(s_i)_{i \geq 0}$ defined below. The parameter $D \geq 4$ controls the density of the spanner. Specifically, the expected size of the spanner will be shown to be roughly $Dn/e$.

$$s_0 = s_1 = D \quad \text{and} \quad s_i = (s_{i-1})^{s_{i-1}} \quad \text{for } i \geq 2$$

We assume for the moment, and with little loss in generality, that the number of vertices $n = s_1^2 s_2^2 \cdots s_{L-1}^2 s_L$. Before moving on we note some facts about the $(s_i)_i$ sequence.

**Lemma 1** *The sequence $(s_i)_i$ has the following properties, for $D \geq 4$.*

1. *For $n = s_1^2 \ldots s_{L-1}^2 s_L$, $L \leq \log^* n - \log^* D + 1$*
2. *For $i \geq 1$, $\log_b s_i = s_1 \cdots s_{i-1} \log_b D$*
3. *For $i \geq 1$, $s_i \geq 2^{i+1} s_1 \cdots s_{i-1}$*

*Proof* Let $2 \uparrow (i, x)$ be an exponential stack of $i$ 2s with an $x$ on top, i.e., $2 \uparrow (0, x) = x$ and $2 \uparrow (i + 1, x) = 2^{2 \uparrow (i,x)}$. One can easily prove by induction that for $i \geq 1$, $s_i \geq 2 \uparrow (i-1, D)$. Thus, $\log^* n \geq \log^* S_L \geq \log^*(2 \uparrow (L-1, D)) = L - 1 + \log^* D$. This proves part 1. For part 2, $\log_b s_1 = \log_b D$ and for $i \geq 2$ and assuming the claim holds for $i - 1$, $\log_b s_i = s_{i-1} \log_b s_{i-1} = s_{i-1} s_{i-2} \cdots s_1 \log_b D$. For part 3, the claim holds for $s_1 = D \geq 2^{1+1}$. For $i \geq 2$, $s_{i-1} \geq 2^i > i + 1$, so $s_i = s_{i-1}^{s_{i-1}} \geq s_{i-1}^{i+1} > s_{i-1} s_{i-2} \cdots s_1 2^{i+1}$. □

Our algorithm computes a series of graph-cluster pairs of the form $(G_{i,j}, \mathscr{C}_{i,j})$, where $\mathscr{C}_{i,j}$ is a complete clustering of $G_{i,j}$. Here $G_{i,j}$ represents a contracted version of the original graph after $i$ full *rounds* and $j$ *iterations* of the $(i+1)$th round. The goal of the $(i + 1)$th round is to reduce the number of vertices by a factor at least $s_i s_{i+1}$, for $i \geq 1$, and at least $s_1 = D$, for $i = 0$. Beginning with $G_{i,0}$ and a trivial clustering $\mathscr{C}_{i,0} = \{\{v\} \mid v \in V(G_{i,0})\}$ of $G_{i,0}$, our algorithm calls **Expand** $\log_{s_i}(s_i s_{i+1}) = s_i + 1$ times (or one time if $i = 0$), each with sampling probability $s_i^{-1}$. For $i \geq 1$, the effect of these calls is to grow a complete clustering $\mathscr{C}_{i,s_i+1}$ of $G_{i,s_i+1}$ that has an expected $|V(G_{i,0})|/s_i^{s_i+1} = |V(G_{i,0})|/(s_i s_{i+1})$ clusters. At the end of the $(i + 1)$th round we *contract* all clusters in $\mathscr{C}_{i,s_i+1}$ in preparation for the $(i + 2)$th round. The spanner consists of all edges selected in lines 4 and 7 in the calls to **Expand**. We use $S$ to denote the set of spanner edges selected up to some moment in time. A key invariant maintained by this algorithm is that if $C$ is a cluster in any $\mathscr{C}_{i,j}$, then $S$ contains a spanning tree of $\circledcirc^{-1}(C)$. The maximum radius of this tree (as a function of $i$ and $j$) will be investigated shortly.

The rules for generating graphs and clusters are summarized below. Note that **Expand** is a randomized procedure so the graphs and clusters are all random variables.

– The initial graph-clustering pair:

$$(G_{0,0}, \mathscr{C}_{0,0}) = (G, \{\{v\} \mid v \in V(G)\})$$

– The $i$th round begins by contracting the last clustering of the previous round, where $k = 1$ if $i = 1$ and $k = s_{i-1}+1$ otherwise:

$$(G_{i,0}, \mathscr{C}_{i,0}) = (G_{i-1,k} \circledcirc \mathscr{C}_{i-1,k}, \{\{v\} \mid v \in V(G_{i,0}))$$

– The $j$th iteration samples and expands clusters from the $(j-1)$th iteration, for $j \in [1, s_i + 1]$, if $i > 1$, and $j = 1$

otherwise:

$$(G_{i,j}, \mathscr{C}_{i,j}) = \textbf{Expand}(G_{i,j-1}, \mathscr{C}_{i,j-1}, s_i^{-1})$$

– The sampling probability is forced to be zero in the final round and iteration:

$$(G_{L-1,s_{L-1}+1}, \mathscr{C}_{L-1,s_{L-1}+1}) \\ = \textbf{Expand}(G_{L-1,s_{L-1}}, \mathscr{C}_{L-1,s_{L-1}}, 0)$$

Consider an arbitrary cluster $C \in \mathscr{C}_{i,j}$. With respect to $G_{i,j}$, $C$ is spanned in $S$ by a radius-$j$ tree. However, with respect to the original graph the tree spanning $\circledcirc^{-1}(C)$ may have a significantly wider radius. Lemma 2 follows from Observation 1 and properties of the **Expand** procedure.

**Lemma 2** *Let $r_{i,j}$ be the maximum radius of the tree spanning $\circledcirc^{-1}(C)$, where $C \in \mathscr{C}_{i,j}$, and let $d_{i,j}$ be the expected nominal "density" $\mathbb{E}[n/|\mathscr{C}_{i,j}|]$. If $j$ is omitted it is assumed to be zero. Then $\{r_{i,j}\}_{i,j}$ and $\{d_{i,j}\}_{i,j}$ obey the following equalities:*

1. $r_0 = 0$.
2. $r_{i,j} = j(2r_i + 1) + r_i < (j + \frac{1}{2})(2r_i + 1)$.
3. *For* $j \in \{0, 1\}$, $d_{0,j} = s_0^j = D^j$. *For* $j \in [0, s_1 + 1]$, $d_{1,j} = s_1^{j+1}$.
4. *For* $i > 1$, $j \in [0, s_i + 1]$, $d_{i,j} = s_1^2 s_2^2 \cdots s_{i-1}^2 \cdot (s_i)^{j+1}$.

*Proof* A call to $\textbf{Expand}(G_{i,j}, \mathscr{C}_{i,j}, s_i^{-1})$ returns a graph $G_{i,j+1}$ that is not larger than $G_{i,j}$ (some dead vertices may be removed) and a set of clusters $\mathscr{C}_{i,j+1}$ whose radii, w.r.t. $G_{i,j}$, are at most one more than those in $\mathscr{C}_{i,j}$. Furthermore, the expected size of $\mathscr{C}_{i,j+1}$ is $|\mathscr{C}_{i,j}|/s_i$. It follows that, for any $i \ge 1$, after a series of $j \le \log_{s_i}(s_i s_{i+1})$ calls to **Expand**, starting with $(G_{i,0}, \mathscr{C}_{i,0})$, we will have obtained a clustering with $|V(G_{i,0})|/s_i^j$ clusters in expectation, each of radius $j$ w.r.t. $G_{i,0}$. A simple proof by induction shows that the expected nominal density $d_{i,j} = \mathbb{E}[n/|\mathscr{C}_{i,j}|]$ is $s_1^2 \cdots s_{i-1}^2 (s_i)^{j+1}$. (The algorithm does not use the actual density $n/|\mathscr{C}_{i,j}|$, only its expectation $d_{i,j}$, which can be computed locally.) We now turn to bounding the cluster radius. Let $(v_0, \ldots, v_j)$ be a path in $G_{i,0}$ from the center of a cluster $C \in \mathscr{C}_{i,j}$ to the most distant vertex in the cluster. Let $v'_0, \ldots, v'_j$ be the centers of $\circledcirc^{-1}(v_0), \ldots, \circledcirc^{-1}(v_j)$ in the original graph and let $v''_j$ be the farthest vertex from $v'_j$ in $\circledcirc^{-1}(v_j)$. It follows that $\delta_S(v'_k, v'_{k+1}) \le 2r_i + 1$, for $k \in [0, j)$, and that $\delta_S(v'_j, v''_j) \le r_i$; hence, the radius of $\circledcirc^{-1}(C)$ is never more than $j(2r_i + 1) + r_i$. □

**Lemma 3** *For $i \in [1, L-1]$ and $j \in [0, s_i + 1]$:*

1. $r_i \le 3 \cdot 2^{i-1} s_1 \cdots s_{i-1} - 2$
2. $r_{i,j} < (j + 1/2) 3 \cdot 2^i s_1 \cdots s_{i-1}$
3. $r_{i,j} < 3 \cdot 2^i \log_D d_{i,j}$

*Proof* The proof of part 1 is by induction. For $i = 1$ the $i$th round consists of 1 call to **Expand**, which creates a radius-1 clustering; $1 \le 3 \cdot 2^0 - 2$. Assuming the bound holds for $r_{i-1}$ we have, by Lemma 2(2):

$$\begin{aligned} r_i &= (s_{i-1} + 1)(2r_{i-1} + 1) + r_{i-1} && \{\text{Lemma 2(2)}\} \\ &< (s_{i-1} + \tfrac{3}{2})(2(3 \cdot 2^{i-2} s_1 \cdots s_{i-2} - 2) + 1) && \{\text{Ind.}\} \\ &< 3 \cdot 2^{i-1} s_1 \cdots s_{i-1} - 3s_{i-1} + 9 \cdot 2^{i-2} s_1 \ldots s_{i-2} \\ &< 3 \cdot 2^{i-1} s_1 \cdots s_{i-1} - 2 \end{aligned}$$

The last inequality follows from Lemma 1(3), which states that $3s_{i-1} \ge 3 \cdot 2^i s_1 \ldots s_{i-2}$. Part 2 follows directly from part 1 and Lemma 2(2). For part 3, one may verify that it holds for $i \in \{0, 1\}$. For $i > 1$:

$$\begin{aligned} &3 \cdot 2^i \log_D d_{i,j} \\ &\quad = 3 \cdot 2^i \log_D(s_1^2 \cdots s_{i-1}^2 s_i^{j+1}) && \{\text{Lemma 2(4)}\} \\ &\quad \ge 3 \cdot 2^i (j+1) s_1 \cdots s_{i-1} && \{\text{Lemma 1(2)}\} \\ &\quad > r_{i,j} && \{\text{Part 2}\} \end{aligned}$$

This concludes the proof. □

**Lemma 4** *Let $(u', v')$ be an edge from the original graph removed from consideration in one of the following ways:*

1. *A vertex $u$ in $G_{i,j}$ was marked dead in the call* $\textbf{Expand}(G_{i,j}, \mathscr{C}_{i,j}, s_i^{-1})$ *and* $u' \in \circledcirc^{-1}(u)$.
2. *Both $u'$ and $v'$ lie in $\circledcirc^{-1}(C)$ for some cluster $C \in \mathscr{C}_{i-1,s_{i-1}+1}$ contracted in the formation of $\mathscr{C}_{i,0}$.*

*In the first case, $\delta_S(u', v') \le (2j+2)(2r_i+1) - 1$ and in the second $\delta_S(u', v') \le 2r_i$, where $S$ is the set of spanner edges.*

*Proof* Figure 3 depicts the situation of part 1. Here $u$ is marked dead, $v, w$ lie in an unsampled cluster $C \in \mathscr{C}_{i,j}$, $u', u'' \in \circledcirc^{-1}(u)$, $v' \in \circledcirc^{-1}(v)$, $w' \in \circledcirc^{-1}(w)$ and, in line 7 of **Expand**, the edge $(u'', w')$ was included in favor of $(u', v')$.

Then $\delta_S(u', v') \le \delta_S(u', w') + \delta_S(w', v') \le (2r_i + 1) + 2r_{i,j}$, which is exactly $(2j + 2)(2r_i + 1) - 1$ since $r_{i,j} = j(2r_i + 1) + r_i$. Turning to part 2, the cluster $C$ containing $u'$ and $v'$ has radius $r_i$; thus $\delta_S(u', v') \le 2r_i$. □

**Lemma 5** *The edges identified in lines 4 and 7 in all calls to* **Expand** *form an $O(2^{\log^* n - \log^* D} \log_D n)$-spanner.*

*Proof* Recall that in the last call to **Expand**, namely $\textbf{Expand}(G_{L-1,s_{L-1}}, \mathscr{C}_{L-1,s_{L-1}}, 0)$, we impose a sampling probability of zero, which has the effect of killing every remaining vertex in the graph. Thus, the maximum radius of any cluster is $r_{L-1,s_{L-1}}$ and, by Lemma 4, the maximum

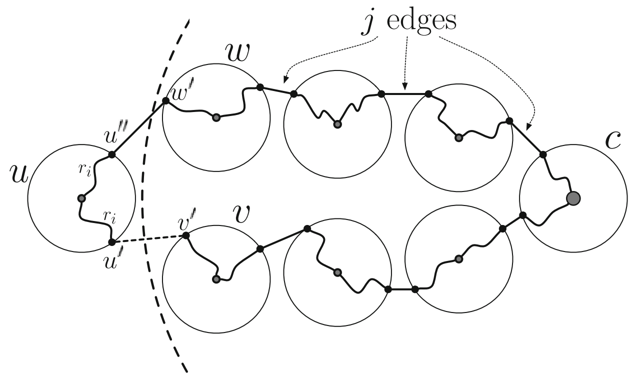


**Fig. 3** The *hollow circles* represent vertices in $G_{i,j}$, or, equivalently, sets of vertices in $G$ spanned by trees with radius $r_i$. The vertex $c \in V(G_{i,j})$ is the center of a cluster in $\mathscr{C}_{i,j}$ (with its boundary indicated by an *arc*) and $u \in V(G_{i,j})$ is a vertex that died in the call to **Expand**$(G_{i,j}, \mathscr{C}_{i,j}, s_i^{-1})$. The edge $(u'', w')$ was included in the spanner (in line 7 of **Expand**) and $(u', v')$ was not. There is a path from $u'$ to $v'$ in the spanner with length at most $(2j + 2)(2r_i + 1) - 1$

distortion is:

$$\begin{aligned}
&(2s_{L-1} + 2)(2r_{L-1} + 1) - 1 \\
&\quad < (2s_{L-1} + 2)(3 \cdot 2^{L-1} s_1 \cdots s_{L-2} - 3) && \{\text{Lemma 3(1)}\} \\
&\quad = 3 \cdot 2^L s_1 \cdots s_{L-1} + 3 \cdot 2^L s_1 \cdots s_{L-2} \\
&\qquad -6s_{L-1} - 6 \\
&\quad < 3 \cdot 2^L s_1 \cdots s_{L-1} && \{\text{Lemma 1(3)}\} \\
&\quad = 3 \cdot 2^L \log_D s_L && \{\text{Lemma 1(2)}\} \\
&\quad \le 3 \cdot 2^{\log^* n - \log^* D + 1} \log_D n && \{\text{Lemma 1(1)}\}
\end{aligned}$$

□

In the proof of Lemma 6, we analyze the expected number of edges contributed by successive calls to the **Expand** procedure. Our analysis corrects an error made by Baswana and Sen [10, p. 547], that given an unweighted graph, their algorithm computes a $(2k-1)$-spanner with size $O(kn+n^{1+1/k})$. This claim may in fact be true, but the argument made in [10, Lemma 4.1] only shows that it is $O(kn + \log kn^{1+1/k})$.

**Lemma 6** *Let* $D \geq 4$. *The expected size of the spanner (i.e., the expected number of edges selected in lines 4 and 7 of* **Expand**) *is* $\frac{Dn}{e} + O(n \log D)$.

*Proof* Consider a single vertex $v \in V(G')$ in a series of $t$ calls to **Expand**, each with sampling probability $p$ and suppose the first call is **Expand**$(G', \mathscr{C}', p)$. The expected number of edges contributed by $v$ in the first call depends on the number of clusters in $\mathscr{C}'$ incident to $v$ and whether $v$ is still alive, i.e., in a cluster in $\mathscr{C}'$. Suppose that $v$ is in $C_0 \in \mathscr{C}'$ and adjacent to $C_1, \ldots, C_q \in \mathscr{C}'$. If $C_0$ is sampled then $v$ contributes zero edges and remains alive; if any of $C_1, \ldots, C_q$ are sampled $v$ contributes 1 edge (in Line 4) and remains alive; otherwise $v$ contributes $q$ edges (in line 7) and dies. It is important to note that the expected contribution of $v$ in this call to **Expand** depends solely on $q$ and $p$ and *not* on the graph topology, the radii of clusters in $\mathscr{C}'$ or any other factors.

Let $Y_p(q_1, \ldots, q_t)$ be the number of edges contributed by a specific live vertex $v$, where, if $v$ is live at the $i$th call to **Expand**, it is incident to $q_i$ live clusters. (In other words, we imagine that an adversary arranges for $v$ to be adjacent to $q_i$ clusters in the $i$th call to **Expand** in order to elicit worst case behavior. There need not be any graph for which the sequence $q_1, \ldots, q_t$ is especially likely.) Thus:

$$\begin{aligned}
\mathbb{E}[Y_p(q_1, \ldots, q_t)] &= (1 - (1-p)^{q_1+1})\mathbb{E}[Y_p(q_2, \ldots, q_t)] \\
&\quad + q_1(1-p)^{q_1+1} + (1-p)(1-(1-p)^{q_1}) \qquad (1)
\end{aligned}$$

i.e., $v$ lives with probability $1-(1-p)^{q_1+1}$, if not all clusters are unsampled, and contributes $Y_p(q_2, \ldots, q_t)$ thereafter; if all clusters are unsampled $v$ contributes $q_1$ edges; if $C_0$ is unsampled but one of $C_1, \ldots, C_{q_1}$ is sampled then $v$ contributes 1 edge. We let $X_p^t$ be the expected worst case scenario:

$$X_p^t = \max_{q_1,\ldots,q_t} \mathbb{E}[Y_p(q_1, \ldots, q_t)]$$

i.e., $X_p^t$ is the maximum expected number of edges contributed by a vertex if an adversary were to select $q_1, \ldots, q_t$. Note that $X_p^t$ is *not* a random variable. Clearly $X_p^0 = 0$. For $t \geq 1$, $X_p^t$ is, by the observations above, precisely:

$$\begin{aligned}
X_p^t &= \max_{q \geq 0} \Big\{(1 - (1-p)^{q+1})X_p^{t-1} + q(1-p)^{q+1} \\
&\qquad + (1-p)(1-(1-p)^q)\Big\} \qquad \{\text{From (1)}\} \\
&= \max_{q \geq 0} \Big\{X_p^{t-1} + (1-p) \\
&\qquad + (q - 1 - X_p^{t-1})(1-p)^{q+1}\Big\} \qquad (2)
\end{aligned}$$

Let us consider the base case, when $t = 1$ and $X_p^1 = (1 - p) + (q-1)(1-p)^{q+1}$. This quantity is maximized at $q = -1/\ln(1-p) + 1$ (take the derivative), which is barely more than $p^{-1} + 1$, say $p^{-1} + 1 + \epsilon$. Thus, we can bound $X_p^1$ as:

$$\begin{aligned}
X_p^1 &= (1-p) + (p^{-1} + \epsilon)(1-p)^{1/p+2+\epsilon} \\
&< (1-p) + e^{-1}(p^{-1} + \epsilon)(1-p)^{2+\epsilon}
\end{aligned}$$

The previous inequality holds since $1+x < e^x$ for all $x \neq 0$. The following inequality holds,[4] since $(p^{-1} + \epsilon)(1-p)^\epsilon < p^{-1}$ for any $p$ and $\epsilon > 0$.

$$\begin{aligned}
&< (1-p) + e^{-1}p^{-1}(1-p)^2 \\
&= (1-p) + (ep)^{-1}(1 - 2p + p^2) \\
&< (1 - 2/e) + (ep)^{-1} \qquad (3)
\end{aligned}$$

We will now prove by induction that:

$$X_p^t \leq p^{-1}(\ln(t+1) - \gamma) + t \qquad (4)$$

[4] Let $f(x) = (C+x)(1-1/C)^x$. (Here $C, x$ play the roles of $p^{-1}, \epsilon$.) Clearly $f(0) = C$. The first derivative $f'(x) = (1-1/C)^x(1 + (C + x)\ln(1-1/C)) < (1-1/C)^x(1-(C+x)/C)$ is strictly less than 0 for $x > 0$, implying that $f(x) < C$ for $x > 0$.

where $\gamma = \ln 2 - 1/e \geq 0.325$. Equation (4) clearly holds for the base case $t = 1$. Substituting the inductive hypothesis for $t-1$ into (2) we have:

$$X_p^t \leq \max_{q \geq 0} \Big\{ p^{-1}(\ln t - \gamma) + t - 1 + (1-p) + (q - p^{-1}(\ln t - \gamma) - t)(1-p)^{q+1} \Big\} \quad (5)$$

In general, the function $(q-x)(1-p)^{q+y}$ is maximized at $q = -1/\ln(1-p) + x > p^{-1} + x$, for any $x$, $y$, and $p \neq 0$. In this case $x = p^{-1}(\ln t - \gamma) + t$ and $y = 1$, that is, $q = t + p^{-1}(\ln t - \gamma + 1) + \epsilon$ for some small $\epsilon > 0$.

$$\begin{aligned}(5) &\leq p^{-1}(\ln t - \gamma) + t \\ &\quad + (p^{-1} + \epsilon)(1-p)^{p^{-1}(\ln t - \gamma + 1) + t + 1 + \epsilon} \\ &< p^{-1}(\ln t - \gamma) + t \\ &\quad + (p^{-1} + \epsilon)(1-p)^{t+1+\epsilon} e^{-(\ln t - \gamma + 1)} \\ &< p^{-1}(\ln t - \gamma + e^{-(\ln t - \gamma + 1)}) + t\end{aligned}$$

The previous two lines follow from the fact that $1 + x < e^x$, for $x \neq 0$, and $(p^{-1} + \epsilon)(1-p)^\epsilon < p^{-1}$, for $\epsilon > 0$. See footnote 4.

$$\begin{aligned}&= p^{-1}(\ln t - \gamma + \tfrac{2}{e^{1+1/e}} t^{-1}) + t && \{\gamma = \ln 2 - 1/e\} \\ &< p^{-1}(\ln t - \gamma + 0.51/t) + t && \{2/e^{1+1/e} < 0.51\} \\ &< p^{-1}(\ln(t+1) - \gamma) + t && \{\ln t + \tfrac{0.51}{t} < \ln(t+1)\}\end{aligned}$$

This completes the inductive proof of (4). Note that our upper bound on $X_p^t$ depends on $p$ and $t$ but is explicitly independent of the size of the graph. Thus, the expected number of edges contributed in the $i$th round is at most $\mathbb{E}[|V(G_{i-1,0})|] \cdot X_p^t = \frac{nX_p^t}{d_{i-1}}$, where $p$ and $t$ are the sampling probability and number of iterations in the $i$th round. It follows that the expected number of spanner edges contributed in the 1st and 2nd rounds are:

$$nX_{1/s_0}^1 \leq n(D/e + 1 - 2/e) \qquad \{\text{From (3)}\}$$

$$\frac{n}{s_1} X_{1/s_1}^{s_1+1} \leq \frac{n}{D}(D+1)(\ln(D+2) - \gamma + 1) \qquad \{\text{From (4)}\}$$

We bound the number of spanner edges contributed by all subsequent rounds as follows. From (4) and the definition of $d_{i-1}$ we have:

$$\sum_{i \geq 3} \frac{n}{d_{i-1}} X_{1/s_{i-1}}^{s_i+1} \leq n \cdot \sum_{i \geq 3} \frac{(s_{i-1}+1)[\ln(s_{i-1}+2) - \gamma + 1]}{s_1^2 s_2^2 \cdots s_{i-2}^2 s_{i-1}}$$

Since $\ln(x+2) < \ln x + 3/x$ for all $x \geq 1$ we can bound this expression as:

$$< n \cdot \sum_{i \geq 3} \frac{(1 + 1/s_{i-1})[\ln s_{i-1} + 3/s_{i-1} - \gamma + 1]}{s_1^2 s_2^2 \cdots s_{i-2}^2}$$

$$< n \cdot \sum_{i \geq 3} \frac{\ln s_{i-1} + \big[(5 + \ln s_{i-1} - \gamma)/s_{i-1} - \gamma + 1\big]}{s_1^2 s_2^2 \cdots s_{i-2}^2}$$

The bracketed expression is maximized at $i = 3$ (so $s_{i-1} \geq s_2 \geq 4^4$) and is always less than 0.75.

$$< n \cdot \sum_{i \geq 3} \frac{\ln s_{i-1} + 0.75}{s_1^2 s_2^2 \cdots s_{i-2}^2}$$

From Lemma 1(2) we have $\ln s_{i-1} = s_1 \cdots s_{i-2} \ln D$, implying:

$$< n \cdot \sum_{i \geq 3} \frac{\ln D + 0.75/D}{s_1 s_2 \cdots s_{i-2}}$$

Since $s_1 = D \geq 4$ and $s_j \geq (D^D)^{j-1}$, for $j \geq 2$, the second term in this sum dominates all subsequent terms:

$$< n \cdot \frac{(1 + 2/D^D)(\ln D + 0.75/D)}{D}$$

$$< \frac{n(\ln(D) + 0.2)}{D}$$

Thus, the expected total size of the spanner is $n(D/e + 1 - 2/e + (1 + 1/D)(\ln(D+2) - \gamma + 1) + (\ln D + 0.2)/D) = \frac{Dn}{e} + O(n \log D)$. □

It is very simple to construct our spanner sequentially in $O(m \log n / \log\log n)$ time, where $m$ is the number of edges in the graph. Each graph contraction or call to **Expand** takes linear time and there are $\log_{s_{L-1}} s_L + 1 + \log_{s_{L-2}} s_{L-1} + 1 + \cdots = L + s_{L-1} + s_{L-2} + \cdots$ such calls. The $s_{L-1}$ term dominates the sum and, since $s_{L-1}^{s_{L-1}+1} < n$ it follows that $s_{L-1} < 2\log n / \log\log n$. Recall that we assumed $n = s_1^2 \cdots s_{L-1}^2 s_L$.

There are several obstacles to implementing this algorithm in a distributed network, the most difficult of which is minimizing the maximum message length. Theorem 2 shows how the algorithm can be implemented efficiently in a distributed network for any $n$.

**Theorem 2** *Consider a synchronized distributed network on $n$-nodes in which the maximum message length is $O(\log^\epsilon n)$, for some fixed $\epsilon$. Let $D < \log^\epsilon n$ and $t = \epsilon^{-1} 2^{\log^* n - \log^* D} \log_D n$. Then in time $O(t + \log n)$ an $O(t)$-spanner can be computed that, with high probability spanner has size $\frac{Dn}{e} + O(n \log D)$. This distortion is within a constant factor of optimal for any fixed $i$ and $D \geq \log^{(i)} n$.*

*Proof* Before the first round of communication every vertex performs the sampling steps (line 1) in all calls to **Expand**. The behavior of a vertex $u$ in a contracted graph is controlled by the center $c$ of $\circledcirc^{-1}(u)$, i.e., at the beginning of the algorithm $c$ selects the round and iteration when its cluster is first left unsampled. Furthermore, every vertex in $\circledcirc^{-1}(u)$ is aware of $c$'s sampling decisions and will propagate this information to vertices that join clusters centered at $c$. Consider the contracted graph and clustering $(G_{i,j}, \mathscr{C}_{i,j})$. Let $u \in C \in \mathscr{C}_{i,j}$ and let $c \in \circledcirc^{-1}(u)$ and $c' \in \circledcirc^{-1}(C)$ be the centers of their respective sets. For every vertex $w \in \circledcirc^{-1}(u)$,

$w$ maintains two spanner edges $(w, p_1(w))$ and $(w, p_2(w))$ that lead to the centers $c$ and $c'$. In other words, it is possible for any node $w$ to send a message to $c$ (the center of the node representing $w$ in $G_{i,j}$) by following $p_1$ pointers; similarly, following $p_2$ pointers leads to the center $c'$ of the $\mathscr{C}_{i,j}$ cluster representing $w$.

We first show that with $O(\log^\epsilon n)$-length messages, each call to **Expand**$(G_{i,j}, \mathscr{C}_{i,j}, s_i^{-1})$ can, with high probability, be executed in time $O(r_{i,j} + s_i \log^{1-\epsilon} n)$. Let $u \in V(G_{i,j})$ be a vertex in $C_0 \in \mathscr{C}_{i,j}$ and adjacent to $C_1, \ldots, C_q \in \mathscr{C}_{i,j}$. If one of $C_0, \ldots, C_q$ is sampled for inclusion in $\mathscr{C}_{i,j+1}$ then $u$ will remain alive; otherwise it and, effectively, all vertices in $\circledcirc^{-1}(u)$ will die. We consider these two cases below.

If $u$ survives this call to **Expand** then all vertices in $\circledcirc^{-1}(u)$ will be able to complete their tasks in exactly $2r_i + 1$ time steps using unit length messages. In the first step, each vertex $u' \in \circledcirc^{-1}(u)$ selects an arbitrary edge connecting it to a sampled cluster, i.e., some edge $(u', v')$ where $v' \in \circledcirc^{-1}(C_k)$ for a sampled $C_k \in \mathscr{C}_{i,j}$. Each vertex in $\circledcirc^{-1}(u)$ might have selected such an edge. It is up to the center $c$ of $\circledcirc^{-1}(u)$ to decide which edge to include in the spanner (and therefore which sampled cluster $u$ is to join). In the next $r_i$ time steps each vertex $u' \in \circledcirc^{-1}(u)$ transmits to $p_1(u')$ at most one candidate edge connecting $u$ to a sampled cluster. Suppose, after $r_i$ steps the center $c \in \circledcirc^{-1}(u)$ selects the edge $(u', v')$ for inclusion in the spanner. In the next $r_i$ steps $c$ sends a message to $u'$ notifying it of this fact. For each vertex $w$ on the path from $c$ to $u'$ inclusive, we set $p_2(w)$ to be the neighbor on the path from $w$ to $v'$. Every other vertex $x \in \circledcirc^{-1}(u)$ sets $p_2(x)$ equal to its $p_1(x)$, i.e., also pointing in the direction of $v'$. This restores the invariant that $p_1$ and $p_2$ point in the direction of the vertex and cluster centers in the now-current contracted graph $G_{i,j+1}$. See Fig. 4. After a round is completed, when all clusters are contracted, each vertex $w$ will simply set $p_1(w)$ equal to $p_2(w)$.

When $u$ dies the implementation of **Expand** can be more time consuming. On the other hand, when $u$ dies the vertices in $\circledcirc^{-1}(u)$ do not participate further in the construction of the spanner, so future calls to **Expand** may safely proceed on schedule. When $u$ dies the center $c \in \circledcirc^{-1}(u)$ must choose exactly one edge from $u$ to each of $C_1, \ldots, C_q$ in order to execute line 7 of **Expand**. This is also easy to accomplish in $2r_i + 1$ time steps if the maximum message length is $q$ but $q$ could, in principle, be very large. (Rather than have each vertex send at most one candidate edge to its parent it would send up to $q$ candidates that connect to some subset of $C_1, \ldots, C_q$.) Note that the probability that $u$ dies given that $q \geq 4s_i . \ln n$ is less than $(1 - s_i^{-1})^{4s_i \ln n} < n^{-4}$. Thus, if any vertex in $\circledcirc^{-1}(u)$ detects that $q > 4s_i \ln n$ it can abort the normal execution of line 7 of **Expand** and tell *every* vertex in $\circledcirc^{-1}(u)$ to include *all* adjacent edges in the spanner. This modification to the

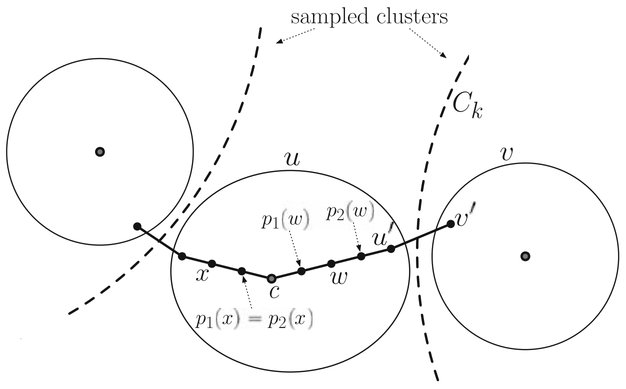


**Fig. 4** In this example $u \in V(G_{i,j})$ is incident to multiple sampled clusters. The center $c$ of $\circledcirc^{-1}(u)$ chooses to join the sampled cluster $C_k$ and selects the edge $(u', v')$ for inclusion in the spanner. All vertices in $\circledcirc^{-1}(u)$ now update their $p_2$ pointers to point in the direction of the center of $\circledcirc^{-1}(C_k)$. If $w$ lies on the path from $c$ to $u'$ then it sets $p_2(w)$ to point in the direction of $v'$. For any $x$ not on the path from $c$ to $u'$ we set $p_2(x)$ equal to $p_1(x)$, i.e., the centers of $\circledcirc^{-1}(u)$ and $\circledcirc^{-1}(C_k)$ are in the same direction, starting at $x$

algorithm increases the expected size of the spanner only negligibly.[5] Given that $q = O(s_i \log n)$, line 7 of **Expand** can be implemented in $O(r_{i,j} + s_i \log^{1-\epsilon} n)$ time with $O(\log^\epsilon n)$-length messages. Each vertex $u' \in \circledcirc^{-1}(u)$ sends to $p_1(u')$ the set of candidate edges received from descendants in $O(s_i \log^{1-\epsilon} n)$ consecutive steps. This process can be pipelined (a vertex can start sending candidates the moment after it receives one) so the total number of steps needed to kill $u$ is $O(r_i + s_i \log^{1-\epsilon} n)$.

Our spanner algorithm begins by executing successive rounds using the implementation of **Expand** described above. When the expected nominal density exceeds $\log^\epsilon n \log(\log^\epsilon n)$ we prematurely end the current round and complete the spanner construction in two more rounds, the first of which amplifies the nominal density to at least $\log n$. Let $i^*, j^*$ be the first round and iteration for which $d_{i^*,j^*} > \log^\epsilon n \cdot \log(\log^\epsilon n)$, i.e., $d_{i^*,j^*-1} = s_1^2 \cdots s_{i^*-1}^2 s_{i^*}^{j^*} \leq \log^\epsilon n \cdot \log(\log^\epsilon n)$. It follows that $s_{i^*} \leq \log^\epsilon n$ and that $d_{i^*,j^*} \leq \log^{2\epsilon} n \log(\log^\epsilon n)$; see footnote.[6] Thus, the execution time up to round $i^*$, iteration $j^*$ is $O(r_{i^*,j^*} + s_{i^*} \log^{1-\epsilon} n) = O(\log n)$. We complete the construction of the spanner with

[5] The probability that this event ever happens is certainly at most $2n \cdot n^{-4}$, and, if it happens, at most $\binom{n}{2}$ edges may be included in the spanner, i.e., the expected increase in the size of the spanner is less than $1/n$.

[6] Due to the upper bound $D \leq \log^\epsilon n$ placed on $D$, $i^*$ and $j^*$ are at least 1. If $i^* = 1$ then $s_{i^*} = D \leq \log^\epsilon n$ by assumption. If $i^* \geq 2$ and $s_{i^*}$ were greater than $\log^\epsilon n$ then $d_{i^*,j^*-1} = s_1^2 \ldots s_{i^*-1}^2 s_{i^*}^{j^*} > s_{i^*}^{j^*} (\log s_{i^*} / \log\log s_{i^*})^2 > \log^\epsilon n \log(\log^\epsilon n)$, contradicting our choice of $i^*, j^*$. The first inequality follows from the fact that $s_{i^*-1} > \log s_{i^*} / \log\log s_{i^*}$, the second from our assumption that $s_{i^*} \geq \log^\epsilon n$.

two more rounds. The sampling probability in both rounds in held at $(\log n)^{-\epsilon}$.

- The $(i^*+2)$th round begins with a nominal density $d_{i^*,j^*}$. Specifically the pair $(G_{i^*+1,0}, \mathscr{C}_{i^*+1,0})$ is equal to:

  $(G_{i^*,j^*} \circledcirc \mathscr{C}_{i^*,j^*}, \{\{v\} \mid v \in V(G_{i^*+1,0})\})$

- The $(i^*+2)$th round continues until the nominal density is at least $\log n$. For $j \in [1, j^{**}]$ where $j^{**} = \log_{\log^\epsilon n} \frac{\log n}{d_{i^*,j^*}}$, the pair $(G_{i^*+1,j}, \mathscr{C}_{i^*+1,j})$ is:

  **Expand**$(G_{i^*+1,j-1}, \mathscr{C}_{i^*+1,j-1}, (\log n)^{-\epsilon})$

- The first pair in round $i^*+3$, $(G_{i^*+2,0}, \mathscr{C}_{i^*+2,0})$, is equal to:

  $(G_{i^*+1,j^{**}} \circledcirc \mathscr{C}_{i^*+1,j^{**}}, \{\{v\} \mid v \in V(G_{i^*+2,0}))$

- In the final round, for $j \in [1, j^{***})$, where $j^{***} = \log_{\log^\epsilon n} \frac{n}{\log n}$, the pair $(G_{i^*+2,j}, \mathscr{C}_{i^*+2,j})$ is equal to:

  **Expand**$(G_{i^*+2,j-1}, \mathscr{C}_{i^*+2,j-1}, (\log n)^{-\epsilon})$

- In the last iteration and round the sampling probability is forced to be zero, i.e., $(G_{i^*+2,j^{***}}, \mathscr{C}_{i^*+2,j^{***}})$ is:

  **Expand**$(G_{i^*+2,j^{***}-1}, \mathscr{C}_{i^*+2,j^{***}-1}, 0)$

Let $r$, $r'$, and $r''$ be the cluster radii after rounds $i^*+1$, $i^*+2$, and $i^*+3$, respectively. It follows from Lemma 3 that $r = r_{i^*,j^*} < 3 \cdot 2^{i^*} \log_D d_{i^*,j^*}$. Since $d_{i^*,j^*} \le \log^{2\epsilon} n \log(\log^\epsilon n)$ we have:

$$\begin{aligned} r &= < 3 \cdot 2^{i^*} \log_D d_{i^*,j^*} \\ &\le \epsilon 2^{i^*+3} \log_D \log n \qquad \{d_{i^*,j^*} \le \log^{2\epsilon} n \log(\log^\epsilon n)\} \end{aligned}$$

From Lemma 2(2) and the bound on $r$, we can obtain bounds on $r'$ and $r''$:

$$\begin{aligned} r' &\le (\epsilon^{-1} - 1)(2r+1) + r \\ &< (\epsilon^{-1} - \frac{1}{2})(2 \cdot \epsilon 2^{i^*+3} \log_D \log n) \\ &< 2^{i^*+4} \log_D \log n \\ r'' &\le (\log_{\log^\epsilon n} \tfrac{n}{\log n})(2r'+1) + r' \\ &< (\tfrac{\log n}{\epsilon \log\log n} - \epsilon^{-1} + 1/2) 2^{i^*+5} \log_D \log n \\ &< \epsilon^{-1} 2^{i^*+5} \log_D n \end{aligned}$$

Since all vertices have died by the end of the $(i^*+3)$th round it follows from Lemma 4 that the distortion of the spanner is at most $2r'' < \epsilon^{-1} 2^{i^*+6} \log_D n < \epsilon^{-1} 2^{\log^* n - \log^* D + 7} \log_D n$. The smallest sampling probability passed to **Expand** is $p = \log^{-\epsilon} n$. Thus, the last two rounds of the spanner construction take time

$$\begin{aligned} &O(r'' + p^{-1} \log^{1-\epsilon} n) \\ &\quad = O(\epsilon^{-1} 2^{\log^* n - \log^* D} \log_D n + \log n). \end{aligned}$$

The analysis of Lemma 6 shows that the expected number of spanner edges contributed by a single vertex in $t$ calls to **Expand** with sampling probability $p$ is $p^{-1}(\ln(t+1) - \gamma) + t$.

Thus, when $p = \log^{-\epsilon} n$ and $t \le \epsilon^{-1} - 1$, the number of spanner edges contributed in the second to last round is at most:

$$\begin{aligned} &\frac{n}{d_{i^*,j^*}} \cdot [\log^\epsilon n(\ln \epsilon^{-1} - \gamma) + \epsilon^{-1} - 1] \\ &\quad = (1 + o(1)) \frac{n(\ln \epsilon^{-1} - \gamma)}{\log(\log^\epsilon n)} \\ &\quad = o(n) \qquad \{d_{i^*,j^*} \ge \log^\epsilon n \log(\log^\epsilon n)\} \end{aligned}$$

and when $p = \log^{-\epsilon} n$ and $t = \log_{1/p} n / \log n$, the number of edges contributed in the last round is:

$$\begin{aligned} &\frac{n}{\log n} \left[\log^\epsilon n (\ln(\log_{\log^\epsilon n} \tfrac{n}{\log n} + 1) - \gamma) + \log_{\log^\epsilon n} \tfrac{n}{\log n}\right] \\ &\quad < \frac{n}{\log n} \left[\log^\epsilon n \ln(\tfrac{\log n}{\epsilon \log\log n}) + \tfrac{\log n}{\epsilon \log\log n}\right] \\ &\quad = (1 + o(1)) n / (\epsilon \log\log n) \ = \ o(n) \end{aligned}$$

Thus, by Lemma 6, the expected size of the spanner is still dominated by the edges contributed in the first two rounds: $\frac{Dn}{e} + O(n \log D)$.

□

## 3 Lower bounds

We provide lower bounds on distributed algorithms for constructing additive, sublinear additive, and $(1+\epsilon, \beta)$-spanners. Our conclusion is that all additive and sublinear additive spanners require a polynomial number of rounds of communication (which depends on the exact distortion and spanner size) and all $(1+\epsilon, \beta)$-spanners require $\Omega(\epsilon^{-1})$ rounds unless $\beta$ is quite large. All of our time/distortion lower bounds are robust inasmuch as they hold in expectation and on the average.[7]

**Theorem 3** *Consider a randomized distributed algorithm that, given a graph with $n$ vertices, after $\tau$ rounds of communication, returns a spanner $H$ such that $|H| \le n^{1+\rho}$ in expectation, where $\tau^2 = o(n^{1-\rho})$. Then there exists a graph with diameter greater than $n^{1-\rho}/(c(\tau+6))$ such that for all vertices $u, v$: $\mathbb{E}[\delta_H(u,v)] \ge \delta(u,v) + \frac{2(1-1/c)}{\tau+2}(\delta(u,v) - (3\tau+11)) - 1$.*

[7] For example, one could define an average-additive $\beta$-spanner to be one whose average (over all pairs of vertices) additive distortion is $\beta$.

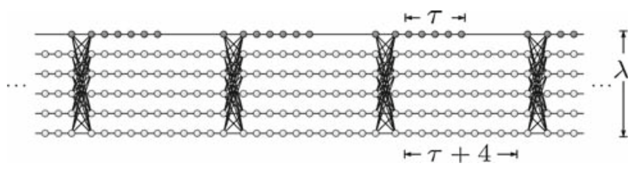


**Fig. 5** One section of our input graph

The proof of Theorem 3 and its corollaries in Theorems 4, 5, and 6 refer to the graph $G(\tau, \lambda, \omega)$, which we now describe. The graph consists of a series of $\omega$ complete $\lambda \times \lambda$ bipartite subgraphs (henceforth *subgraphs*), where vertices on the right side of one subgraph are connected by chains with length $\tau + 5$ to corresponding vertices on the left side of the next subgraph, with the exception of one chain, which is shorter. See Fig. 5.

Specifically, let $v_{L,i,j}$ (and $v_{R,i,j}$) be the $j$th vertex on the left side (right side) of the $i$th subgraph. For $i \in [1, \omega - 1]$ we connect $v_{R,i,1}$ to $v_{L,i+1,1}$ by a path with length $\tau + 1$; this path contains $\tau$ new vertices. For $j \in [2, \lambda]$, we connect $v_{R,i,j}$ to $v_{L,i+1,j}$ by a path with length $\tau + 5$ containing $\tau + 4$ new vertices. In order to make the $\tau$-neighborhood of each vertex in each subgraph look the same (viewing vertices as unlabeled) we attach chains containing $\tau + 1$ new vertices to $\{v_{L,1,j}\}_{j \in [1,\lambda]} \cup \{v_{R,\omega,j}\}_{j \in [1,\lambda]}$. The total numbers of vertices and edges in $G(\tau, \lambda, \omega)$ are

$$\begin{aligned} n' &= \omega(\lambda(\tau + 6) - 4) + \lambda(\tau + 1) - 3(\lambda - 1) + 1 \\ &< (\omega + 1)\lambda(\tau + 6) \\ m' &= \omega\lambda^2 + (\omega + 1)(\lambda(\tau + 5) - 4) - 6(\lambda - 1) + 2 \\ &> \omega\lambda^2 \end{aligned}$$

For parameters $n > 1$, $\rho \in (0, 1)$, and $c \geq 2$, our input to the spanner algorithm is $G(\tau, \lambda, \omega)$, where $\lambda = c(\tau + 6)n^{\rho}$ and $\omega = n^{1-\rho}c^{-1}(\tau + 6)^{-2} - 1$. (We assume that $n^{\rho}$ is an integer. This simplification will clearly have no effect on our argument.) If the algorithm assumes that the vertices have unique labels we assign them a random permutation of $\{1, \ldots, n'\}$. The density of $G(\tau, \lambda, \omega)$ is $m'/n' > \frac{\omega}{\omega+1}\lambda/(\tau + 6)$, which is $\frac{\omega}{\omega+1}cn^{\rho}$. In other words, our spanner algorithm must discard at least a $p = 1 - \frac{1}{c} - \frac{1}{c\omega}$ fraction of the edges. We can make two claims about which edges get discarded and with what probability: (1) only edges in subgraphs may be discarded, and (2) every edge in every subgraph is discarded with exactly the same probability, where the probability is over the random permutation of vertex labels and the random choices made by the algorithm. The first claim follows from the fact that in *any* correct spanner algorithm, an edge $(u, v)$ can only be discarded if either $u$ or $v$ has detected that $(u, v)$ lies on a cycle. (If the algorithm is prepared to discard $(u, v)$ then it is prepared to disconnect the graph.) In our input graph most chain edges *do* lie on cycles; however, no algorithm running in $\tau$ time steps can detect this and, if the algorithm is correct, it must include all of them in the spanner. The second claim follows from the fact that, if we restrict our attention to vertices in subgraphs, all $\tau$-neighborhoods are topologically identical; furthermore, the distributions over labeled $\tau$-neighborhoods are identical.[8]

*Proof* (Theorem 3) A *critical edge* is one of the form $(v_{L,i,1}, v_{R,i,1})$. As observed above, the probability that any edge in such a subgraph is discarded is at least $p = 1 - \frac{1}{c} - \frac{1}{c\omega}$. In particular, the expected number of critical edges discarded is greater than $\omega(1 - \frac{1}{c}) - 1$. We generously assume that these are the *only* edges discarded. Before considering arbitrary vertices $u, v$ consider the special case of $u = v_{L,i_1,1}$ and $v = v_{L,i_2,1}$, for $i_1 < i_2$. Thus, there is a unique shortest path with length $\delta(u, v) = (i_2 - i_1)(\tau + 2)$ containing the critical edges $(v_{L,k,1}, v_{R,k,1})$ for $k \in [i_1, i_2)$. We claim that after discarding any subset of the critical edges, a shortest path (perhaps not *the* shortest path) from $u$ to $v$ in $H$ still passes through the vertices:

$$v_{L,i_1,1}, v_{R,i_1,1}, \ldots, v_{L,i_2-1,1}, v_{R,i_2-1,1}, v_{L,i_2,1}.$$

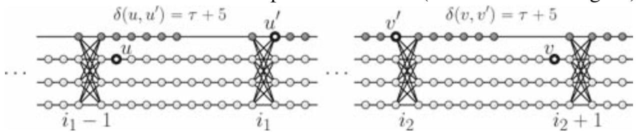
If a shortest path contained the length $\tau + 5$ chain from $v_{R,i,j}$ to $v_{L,i+1,j}$, for $j > 1$, it could be replaced by the equally short path:

$$(v_{R,i,j}, v_{L,i,j}, v_{R,i,1}, \ldots, v_{L,i+1,1}, v_{R,i+1,j}, v_{L,i+1,j}).$$

Thus, a shortest path from $u$ to $v$ in $H$ looks just like the shortest path in $G$, where each missing critical edge is replaced by a length 3 path. Thus, $\mathbb{E}[\delta_H(u, v)] \geq \delta(u, v) + 2p\delta(u, v)/(\tau + 2)$. For two arbitrary vertices $u$ and $v$, the minimum number of critical edges on a path with length $\delta(u, v)$ is $\lceil(\delta(u, v) - 3\tau - 11)/(\tau + 2)\rceil$; see footnote.[9] It

[8] To be more specific, suppose $(u, v)$ is an edge in some subgraph and $\Gamma$ is the union of the $\tau$-neighborhoods of $u$ and $v$. Given the vertex labeling for $\Gamma$ and the random bits used by $\Gamma$ vertices in the execution of the algorithm, we can determine with certainty whether $(u, v)$ is included in the spanner or not. Thus, we can calculate the precise probability with which $(u, v)$ is included in the spanner. This calculation depends only on the topology of $\Gamma$, which is the same for all subgraph edges.

[9] Suppose that $u$ lies on a chain between subgraphs $i_1 - 1$ and $i_1$ and $v$ lies on a chain between subgraphs $i_2$ and $i_2 + 1$, where $i_1 \leq i_2$ and let $u' = v_{R,i_1,1}, v' = v_{L,i_2,1}$. Then $\delta(u, u'), \delta(v, v') \leq \tau + 5$. This is achieved when $u, v$ are the second vertices on their respective chains. (See the below figure)



If $i_2 \leq i_1 + 1$ then $\delta(u, v) \leq 3\tau + 11$ and the claim is vacuously true. Otherwise the *unique* shortest path from $u$ to $v$ passes through $u', v'$, and every critical edge in subgraphs $i_1 + 1, \ldots, i_2 - 1$. There are precisely $(\delta(u', v') - (\tau + 1))/(\tau + 2) \leq (\delta(u, v) - 3\tau - 11)/(\tau + 2)$ such critical edges.

follows from the analysis above that for any $u, v$:

$$\begin{aligned}\mathbb{E}[\delta_H(u,v)] &\geq \delta(u,v) + \tfrac{2p(\delta(u,v)-3\tau-11)}{\tau+2}\\ &= \delta(u,v) + 2\left(1-\tfrac{1}{c}-\tfrac{1}{c\omega}\right)\left(\tfrac{\delta(u,v)-3\tau-11}{\tau+2}\right)\\ &> \delta(u,v) + 2\left(1-\tfrac{1}{c}\right)\left(\tfrac{\delta(u,v)-3\tau-11}{\tau+2}\right) - 1\end{aligned}$$

The last inequality follows from the inequalities $c \geq 2$ and $\frac{\delta(u,v)-3\tau-11}{\tau+2} < \omega$. □

Theorem 4 shows that any algorithm constructing a $(1+\epsilon, \beta)$-spanner with size $n^{1+\rho}$ either needs to take $\Omega(\epsilon^{-1})$ time or have $\beta = \Omega(\epsilon^2 n^{1-\rho})$, even if the distortion guarantees only need to hold in an average or expected sense.

**Theorem 4** *Consider a randomized distributed algorithm that after $\tau$ rounds of communication, returns a spanner $H$ with expected size $n^{1+\rho}$ such that $\delta_H(u,v) \leq (1+\frac{2(1-\gamma)}{\tau+2})\delta(u,v)+\beta$ for any vertices $u, v$. Then $\mathbb{E}[\beta] > \frac{\gamma^2 n^{1-\rho}}{4(\tau+6)^2}$ - O(1). Furthermore:*

$$\mathbb{E}_{u,v}[\delta_H(u,v) - \left(1+\tfrac{2(1-\gamma)}{\tau+2}\right)\delta(u,v)] = \Omega(\gamma^2\tau^{-2}n^{1-\rho}).$$

*Proof* The input graph is as in Theorem 3, where $c = 2/\gamma$. Thus, the number of bipartite subgraphs is $\omega = \frac{n^{1-\rho}}{c(\tau+6)^2} = \frac{n^{1-\rho}\gamma}{2(\tau+6)^2}$. The expected number of critical edges removed is $(1-1/c-1/(c\omega))\omega \geq (1-\gamma/2)\omega - 1$. Let $u$ and $v$ be vertices at distance $\omega(\tau+2)$ whose unique shortest path contains all critical edges. Then $\mathbb{E}[\beta] \geq \mathbb{E}[\delta_H(u,v) - (1+\frac{2(1-\gamma)}{\tau+2})\delta(u,v)] \geq 2(1-\gamma/2)\omega - 1 - 2(1-\gamma)\omega \geq \gamma\omega/2 - 1 = \frac{n^{1-\rho}\gamma^2}{4(\tau+6)^2} - 2$. This large $\beta$ is not an anomaly but holds for the vast majority of vertex pairs. For vertices $u, v$ selected at random, the shortest $u$ to $v$ path will be expected to contain $\mathbb{E}[(\delta(u,v) - O(\tau))/(\tau+2)] = \Omega(\omega)$ critical edges. The argument above then shows that $\mathbb{E}[\delta_H(u,v) - (1+\frac{2(1-\gamma)}{\tau+2})\delta(u,v)] = \Omega(\gamma\omega) = \Omega(\gamma^2\tau^{-2}n^{1-\rho})$. □

Theorem 5 shows that there are no efficient algorithms for constructing additive $\beta$ spanners unless the spanner size and/or $\beta$ are unreasonably large. In particular, any distributed versions of the existing additive 2- and 6-spanners [3,7,17,23,39] would require $\Omega(n^{1/4})$ and $\Omega(n^{1/3})$ time, respectively.

**Theorem 5** *Any distributed algorithm computing an additive $\beta$-spanner with size $n^{1+\rho}$ (in expectation) requires $\Omega(\sqrt{n^{1-\rho}/\beta})$ rounds of communication.*

*Proof* The input graph is $G(\tau, \lambda, \omega)$. Given parameters $\beta \geq 2$, $\rho > 0$, and $n$ sufficiently large, we choose $\tau$, $\lambda$, and $\omega$ as follows: $\tau = \sqrt{n^{1-\rho}/(4\beta)} - 6$, $\lambda = 2(\tau+6)n^\rho$, and $\omega = \frac{n^{1-\rho}}{2(\tau+6)^2}$. Note that $\omega = 2\beta$. As argued in the proof of Theorem 3, if the spanner algorithm took $\tau$ time steps it would discard each critical edge with probability at least $p = \frac{1}{2} - \frac{1}{2\omega}$. The additive distortion between a vertex in the first subgraph and one in the last is twice the number of discarded critical edges. Thus the expected additive distortion is at least $2p\omega = \omega - 1 > \beta$, a contradiction. Hence, any algorithm that produces a spanner with expected additive distortion $\beta$ takes more than $\tau = \Omega(\sqrt{n^{1-\rho}/\beta})$ time. □

Theorem 6 shows that the existing sublinear additive spanners [33,39] have no corresponding efficient distributed implementation. For example, the spanners from [33] have size $o(n^{1+(3/4)^{k+3}})$, with the property that for any two vertices at distance $d$, their distance in the spanner is at most $d + O(d^{1-1/k})$. Theorem 6 implies that any distributed algorithm for computing such a spanner would require $\Omega(n^{(1-(3/4)^{k+3})/(k+1)})$ time.

**Theorem 6** *Suppose a distributed algorithm computes a spanner $H$ with expected size $n^{1+\gamma}$ such that for any $u, v$ with $\delta(u,v) = d$ we have $\delta_H(u,v) = d + O(d^{1-\epsilon})$. Then this algorithm requires $\Omega(n^{\frac{\epsilon(1-\gamma)}{1+\epsilon}})$ rounds of communication to construct $H$.*

*Proof* To be specific, suppose that the algorithm guarantees that for $d = \delta(u,v)$, $\delta_H(u,v) \leq d + cd^{1-\epsilon}$. The input graph is $G(\tau, \lambda, \omega)$ where:

$$\tau + 6 = \tfrac{1}{c}n^{\epsilon(1-\gamma)/(1+\epsilon)}$$
$$\lambda = 4(\tau+6)n^\gamma = \tfrac{4}{c}n^{(\epsilon+\gamma)/(1+\epsilon)}$$
$$\omega = \tfrac{n^{1-\gamma}}{4(\tau+6)^2} = \tfrac{c^2}{4}n^{(1-\gamma)(1-\epsilon)/(1+\epsilon)}.$$

The algorithm discards at least $3\omega/4 - 1$ critical edges in expectation. Thus, for vertices $u, v$ at distance $d = \omega(\tau+2)$ whose shortest path contains all critical edges, the expected additive distortion is $\mathbb{E}[\delta_H(u,v) - \delta(u,v)] \geq 2(3\omega/4 - 1)$. On the other hand the guaranteed additive distortion of $H$ is $cd^{1-\epsilon} < c[(\tau+6)\omega]^{1-\epsilon} < c[\frac{c}{4}n^{(1-\gamma)/(1+\epsilon)}]^{1-\epsilon} < \omega$. Thus, any spanner algorithm producing an $n^{1+\rho}$-size spanner with distortion $d + cd^{1-\epsilon}$ must take time exceeding $\tau = \Omega(n^{\epsilon(1-\gamma)/(1+\epsilon)})$. □

Derbel et al. [15] give lower bounds on distributed spanner algorithms by constructing graphs that look very similar to $G(\tau, \lambda, \omega)$. Rather than complete $\lambda \times \lambda$ bipartite subgraphs they substitute a maximum size (bipartite) graph with girth $2k$, where $k$ is a parameter of the construction. Using an argument similar to ours they show that any algorithm for constructing an $O(n^{1+\rho+O(\rho^2)})$-size spanner in time $\tau = n^{O(\rho^2)}$ has multiplicative stretch $1 + O(1/(\tau\rho))$. (All the constants are given explicitly in the statement of their lower bound.) While their lower bound on multiplicative distortion is better than that of Theorem 3 ($1+O(1/\tau\rho)$ vs. $1+O(1/\tau)$) the additive distortion depends on the *diameter* of the graph, which is significantly smaller in [15]. For time $\tau = n^{O(\rho^2)}$ their lower bound on additive distortion is also $n^{O(\rho^2)}$, which is significantly smaller than the $\tau^{-2}n^{1-\rho} = n^{1-\rho-O(\rho^2)}$ lower bound from Theorem 4.

## 4 Fibonacci spanners

At a very high level the structure and analysis of Fibonacci spanners is similar to Pettie's modular spanners [33] and Thorup and Zwick's emulators [39]. However, their analysis is much more nuanced. Although Fibonacci spanners are not of the sublinear additive variety they actually have much better distortion for polylogarithmic distances than the sublinear additive spanners [33,39] and existing $(1+\epsilon, \beta)$-spanners [19,23,24].

### 4.1 The setup

The algorithm begins by generating vertex sets $V = V_0 \supseteq V_1 \supseteq \cdots \supseteq V_o \supseteq V_{o+1} = \emptyset$, where $V_i$ is sampled from $V_{i-1}$ with probability $q_i/q_{i-1}$. Thus, $\Pr[v \in V_i] = q_i$ and $\mathbb{E}[|V_i|] = q_i n$. We think of $V_0$ and $V_{o+1}$ being sampled with probabilities 1 and $1/n$, respectively. The parameter $o$, lying in $[1, \log_\phi \log n]$, is the *order* of the spanner, which governs the sparseness-distortion tradeoff. Here $\phi = \frac{1+\sqrt{5}}{2}$. Let $p_i(u)$ be the vertex nearest to $u$ in $V_i$. For the sake of specificity, if there are multiple such vertices let $p_i(u)$ be the one whose unique identifier is minimum. The size and distortion of our spanner is a function of $o$, the sampling probabilities, and an integer $\lambda$. If we desire a $(1+\epsilon, \beta)$-spanner $\lambda$ must be on the order of $o/\epsilon$, which makes $\beta = O(o/\epsilon)^o$.

After choosing $o$ and $\lambda$ we define the set $\mathscr{B}_{i+1,\lambda}(v)$, for any $v \in V$, as:

$$\mathscr{B}_{i+1,\lambda}(v) = \left\{ u \in V_i \;\middle|\; \begin{array}{c} \delta(v,u) \le \lambda^i \\ \text{and } \delta(v,u) < \delta(v, V_{i+1}) \end{array} \right\}$$

That is, $\mathscr{B}_{i+1,\lambda}(v)$ is the set of $V_i$-nodes lying in the ball of radius $\min\{\delta(v, V_{i+1}) - 1, \lambda^i\}$ around $v$; see Fig. 6. Notice that since $V_{o+1} = \emptyset$, $\delta(v, V_{o+1}) = \infty$ and $\mathscr{B}_{o+1,\lambda}(v)$ consists of all $V_o$ nodes at distance at most $\lambda^o$. Let $P(v,u)$ be a shortest path from $v$ to $u$. We can express the spanner $S$ very succinctly as the set $S_0 \cup S_1 \cup \cdots \cup S_o$, defined as:

$$S_0 = \bigcup_{\substack{v \in V \\ u \in \mathscr{B}_{1,\lambda}(v)}} P(v,u)$$

$$S_i = \bigcup_{\substack{v \in V_{i-1} \\ u \in \mathscr{B}_{i+1,\lambda}(v)}} P(v,u) \;\cup \bigcup_{\substack{v \in V: \\ \delta(v, p_i(v)) \le \lambda^{i-1}}} P(v, p_i(v))$$

That is, for $i > 0$, we connect every $v \in V_{i-1}$ to every $u \in \mathscr{B}_{i+1,\lambda}(v)$ and every $v \in V$ to every $u \in \mathscr{B}_{1,\lambda}(v)$. Furthermore, we connect each $v \in V$ to its "parent" $p_i(v) \in V_i$ if $\delta(v, p_i(v)) \le \lambda^{i-1}$.

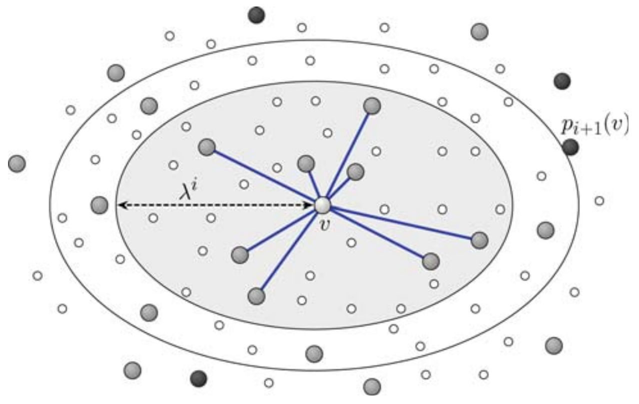


**Fig. 6** The large *light gray vertex* ($v$) is in $V_{i-1}$, *medium gray vertices* are in $V_i$, and *dark gray vertices* in $V_{i+1}$. In the spanner $S$, $v$ is connected by a shortest path to all $V_i$ vertices at distance at most $\lambda^i$ and closer than $p_{i+1}(v)$

### 4.2 Roadmap

Our analysis proceeds as follows. In Lemma 7, we bound the expected size of the spanner as a function of the sampling probabilities. In Lemma 8, we select the optimal probabilities by solving some Fibonacci-like recurrences. In Lemma 9, we show how the distortion of the spanner can be related to two recursively defined functions. In Lemma 10, we find closed form upper bounds on these functions. Theorem 7 and Corollary 1 state the size-distortion tradeoffs of Fibonacci spanners. In Sect. 4.4, we explain how Fibonacci spanners can be constructed distributively using a maximum message length of $\tilde{O}(n^{1/t})$ and analyze how $t$ (negatively) influences the distortion of the spanner. Theorem 8 and Corollary 2 summarize the construction time, size, and distortion of Fibonacci spanners.

**Lemma 7** (Expected size) $\mathbb{E}[|S|] \le on + nq_1^{-1} + \sum_{0 \le i < o} nq_i q_{i+1} q_{i+2}^{-1} \lambda^{i+1}$.

*Proof* We claim the edges in $\bigcup_i \bigcup_{v \in V} P(v, p_i(v))$ form $o$ forests with less than $on$ edges. Recall that $p_i(v) \in V_i$ has the minimum unique identifier among nodes at distance $\delta(v, V_i)$ from $v$. Thus, for any $u \in P(v, p_i(v))$, $p_i(u) = p_i(v)$, implying that $\bigcup_{v \in V} P(v, p_i(v))$ is a forest of $|V_i|$ trees. We focus on the other edges contributed by $S_0, \ldots, S_o$. Fix an $i$. Consider enumerating the vertices in order of distance from a fixed vertex $v \in V$; let them be $v_1, v_2, \ldots, v_{n-1}$. Assuming that $\delta(v, v_j) \le \lambda^i$ the path $P(v, v_j)$ could only appear in $S_i$ if $v \in V_{i-1}$, $v_j \in V_i$ and no vertex in $v, v_1, \ldots, v_j$ appears in $V_{i+1}$. The expected number of edges contributed to $S_i$ on behalf of $v$ is, by the linearity of expectation, less than $\sum_{j=1}^{n-1} \lambda^i q_{i-1} q_i (1 - q_{i+1})^{j+1} < \lambda^i q_{i-1} q_i q_{i+1}^{-1}$. The same argument shows that $\mathbb{E}[|S_0|] \le nq_1^{-1}$. □

Recall that the Fibonacci numbers are defined inductively as: $F_0 = 0$, $F_1 = 1$, and for $k > 1$, $F_k = F_{k-1} + F_{k-2}$. It is well known that $F_k = \frac{1}{\sqrt{5}}(\phi^k - (1-\phi)^k)$ for all $k$, where $\phi = \frac{1+\sqrt{5}}{2}$. The only other property of Fibonacci numbers that we use is that $\phi \cdot F_k + 1 > F_{k+1}$.

**Lemma 8** (Sampling probabilities) *For any* $1 \le o \le \lceil \log_\phi \log n \rceil$ *and* $\lambda > 1$ *the sampling probabilities* $q_1, \ldots, q_o$ *can be chosen such that* $\mathbb{E}[|S|] \le on + O(n^{1+\frac{1}{F_{o+3}-1}}\lambda^\phi)$.

*Proof* By Lemma 7 $\mathbb{E}[|S_0|] = nq_1^{-1}$. We will set $q_1 = n^{-\alpha}\lambda^{-\beta}$, where $\alpha$ and $\beta$ will be fixed later, and attempt to ensure that $\mathbb{E}[|S_i|] = n + \mathbb{E}[|S_0|] = n + n^{1+\alpha}\lambda^\beta$. (The "$n$" term reflects the number of edges in the forest $\bigcup_{v\in V, \delta(v,p_i(v))\le\lambda^{i-1}} P(v, p_i(v))$.) By Lemma 7, $\mathbb{E}[|S_1|] = n + nq_1q_2^{-1}\lambda = n + n^{1-\alpha}\lambda^{1-\beta}q_2^{-1}$, which implies $q_2 = n^{-2\alpha}\lambda^{-2\beta+1}$. We write $q_i$ as $n^{-f_i\alpha}\lambda^{-g_i\beta+h_i}$. Thus, the expected size of $S_i$ is:

$$\begin{aligned}\mathbb{E}[|S_i|] &= n + nq_{i-1}q_iq_{i+1}^{-1}\lambda^i\\ &= n + n^{1+\alpha(f_{i+1}-f_i-f_{i-1})}\\ &\quad \cdot\lambda^{\beta(g_{i+1}-g_i-g_{i-1})-(h_{i+1}-h_i-h_{i-1}-i)}\end{aligned}$$

If $\mathbb{E}[|S_i|]$ is to equal $n + n^{1+\alpha}\lambda^\beta$ it follows that the three sequences $(f_i)_i$, $(g_i)_i$, and $(h_i)_i$ obey the following Fibonacci-like equalities:

$$\begin{array}{lll} f_0 = 0, & f_1 = 1, & f_i = f_{i-1} + f_{i-2} + 1\\ g_0 = 0, & g_1 = 1, & g_i = g_{i-1} + g_{i-2} + 1\\ h_0 = 0, & h_1 = 0, & h_i = h_{i-1} + h_{i-2} + (i-1)\end{array}$$

One can easily prove by induction that $f_i = g_i = F_{i+2} - 1$ and $h_i = F_{i+3} - (i+2)$. Since $q_{o+1} = 1/n$ the expected size of $S_o$, $\mathbb{E}[|S_o|]$, is:

$$\begin{aligned}&\le n + nq_{o-1}q_oq_{o+1}^{-1}\lambda^o\\ &= n + n^{2-\alpha(F_{o+1}-1+F_{o+2}-1)}\\ &\quad\cdot\lambda^{-\beta(F_{o+1}-1+F_{o+2}-1)+(F_{o+2}-(o+1)+F_{o+3}-(o+2)+o)}\\ &= n + n^{2-\alpha(F_{o+3}-2)}\lambda^{-\beta(F_{o+3}-2)+F_{o+4}-(o+3)}\end{aligned}$$

At this point we let $\alpha = 1/(F_{o+3}-1)$ and $\beta = \phi = \frac{1+\sqrt{5}}{2}$:

$$\begin{aligned}&= n + n^{2-\frac{F_{o+3}-2}{F_{o+3}-1}}\lambda^{-\phi F_{o+3}+F_{o+4}+2\phi-(o+3)}\\ &< n + n^{1+\frac{1}{F_{o+3}-1}} \quad \{\phi F_{o+3} + 1 > F_{o+4}\}\end{aligned}$$

Since $S_0, \ldots, S_o$ all have roughly the same expected size, $\mathbb{E}[|S|] \le (o+1)(n + n^{1+\frac{1}{F_{o+3}-1}})\lambda^\phi$. We can reduce this to $on + n^{1+\frac{1}{F_{o+3}-1}}\lambda^\phi$ by substituting $\lambda' = 2\lambda$ for $\lambda$ in all the sampling probabilities. This causes the expected sizes of $S_0, S_1, \ldots$ to decrease geometrically. □

We will see later that for $S$ to behave like a $(1+\epsilon)$-spanner for sufficiently distant vertices we must have $\lambda > o/\epsilon$. Thus, $S$ is sparsest when $o = \log_\phi \log n$ and $\lambda = O(o/\epsilon)$. Since $F_{o+3} \ge \frac{1}{\sqrt{5}}\phi^{\log_\phi \log n + 3} - 1 > \log n$ the expected spanner size is $\mathbb{E}[|S|] = O(\lambda^\phi n^{1+1/(F_{o+3}-1)})$ is $O(n(\epsilon^{-1}\log\log n)^\phi)$.

## 4.3 Analysis of the distortion

Like those of [33,39], the analysis of Fibonacci spanners is conceptually very simple but requires attention to many subtle details. Suppose we wanted to find a $(1+\epsilon)$-approximately shortest path from $x$ to a relatively distant vertex $x'$. So long as edges in $P(x, x')$ are in the spanner we can just walk directly towards $x'$. However, at some point we need to depart from $P(x, x')$, potentially taking us in the opposite direction of $x'$. Since we're only aiming for a $(1+\epsilon)$-approximation, taking a step backwards is no tragedy if this allows us to take $\Omega(1/\epsilon)$ steps toward $x'$. If this is impossible it turns out we may need to take another $1/\epsilon$ steps backward, which, again, is no problem if we can then take $\Omega(1/\epsilon^2)$ steps in the right direction. Roughly speaking, every time we take some steps backward we move to a vertex at a higher level. One could imagine that vertices in $V_{i-1}$ are hilltops from which one can "see" all hilltops $V_i$ within distance (ideally) $\lambda^i$. Thus, the advantage of moving backward from a hilltop in $V_{i-1}$ to a hilltop in $V_i$ is the ability to potentially see a distant hilltop much closer to $x'$. So long as the backward steps are dominated by the forward steps by an $\Omega(1/\epsilon)$ factor the overall distortion will be tolerable.

The intuition sketched above presents a bottom up view of the analysis, which is exactly the opposite of our top down inductive analysis. We suppose that $x$ and $x'$ are at distance $\Delta^i$ for some integer $\Delta$. Our "algorithm" for walking from $x$ to $x'$ in the spanner is to first divide up the actual shortest path into what we call $(i-1)$- and $(i-2)$-segments with lengths of $\Delta^{i-1}$ and $\Delta^{i-2}$, respectively. Starting from $x$ we call the walking algorithm recursively on each successive $(i-2)$-segment; at the same time we walk from $x'$ towards $x$, calling the walking algorithm recursively on each $(i-1)$-segment. Each recursive call has two potential outcomes: it either successfully finds a path from the beginning of an $(i-1)$-segment to the end with length at most $\mathsf{C}_\Delta^{i-1}$ or it finds a path to a hilltop in $V_i$ at distance at most $\mathsf{I}_\Delta^{i-1}$. Depending on the outcome, we call the $(i-1)$-segment either *complete* or *incomplete*. If the two walks starting at $x$ and $x'$ ever meet (i.e., neither encounters an incomplete segment) we can then declare the path from $x$ to $x'$ complete. If this happy scenario does not materialize, we will have found a path from $x$ to a hilltop $p \in V_{i-1}$ and from $x'$ to a hilltop $p' \in V_i$. If $p$ can "see" $p'$ then we have found a path from $x$ to $x'$ and can declare it complete. If $p$ cannot see $p'$ this implies the existence of a fairly short path from $x$ to a hilltop in $V_{i+1}$. In this way we can bound the value $\mathsf{C}_\Delta^i$ (the maximum length of a complete path from $x$ to $x'$) in terms of $\mathsf{C}_\Delta^{i-1}$, $\mathsf{C}_\Delta^{i-2}$, $\mathsf{I}_\Delta^{i-1}$, and $\mathsf{I}_\Delta^{i-2}$. Similarly, we can bound $\mathsf{I}_\Delta^i$ (the distance from $x$ to a hilltop in $V_{i+1}$) inductively. Thus, the analysis amounts to constructing a set of recursive definitions for $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ (Lemma 9), solving them (Lemma 10), then drawing conclusions about the overall distortion of the spanner (Theorem 7).

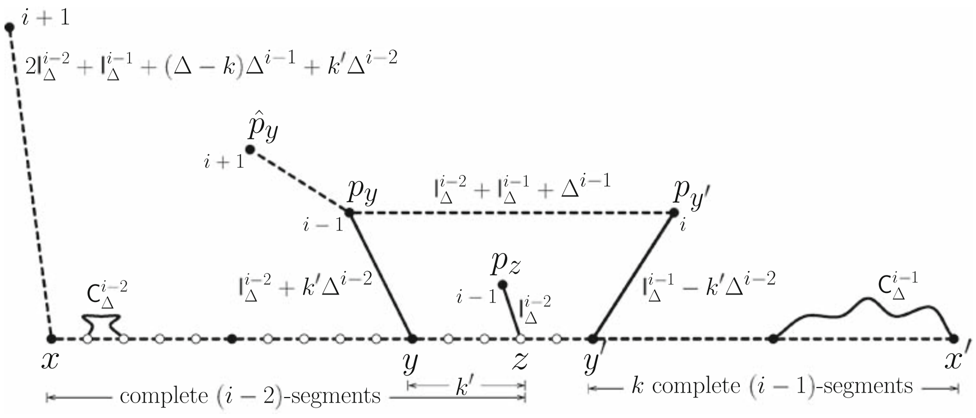

$i+1$
$2\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + (\Delta - k)\Delta^{i-1} + k'\Delta^{i-2}$
$\hat{p}_y$
$i+1$
$p_y$
$\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + \Delta^{i-1}$
$p_{y'}$
$i-1$
$i$
$p_z$
$\mathsf{C}_\Delta^{i-2}$
$\mathsf{I}_\Delta^{i-2} + k'\Delta^{i-2}$
$i-1$
$\mathsf{I}_\Delta^{i-2}$
$\mathsf{I}_\Delta^{i-1} - k'\Delta^{i-2}$
$\mathsf{C}_\Delta^{i-1}$
$x$
$y$
$z$
$y'$
$x'$
complete $(i-2)$-segments
$k'$
$k$ complete $(i-1)$-segments

**Fig. 7** Our attempt to find a path from $x$ to $x'$. The subpath from $x$ to $y$ consists of complete $(i-2)$-segments and the path from $x'$ to $y'$ of complete $(i-1)$-segments. There are paths from $y$ and $y'$ to, respectively, nodes $p_y \in V_{i-1}$ and $p_{y'} \in V_i$. Depending on the distance from $p_y$ to $\hat{p}_y = p_{i+1}(p_y)$, $p_{y'}$ may or may not lie in $\mathscr{B}_{i+1,\lambda}(p_y)$

Before setting up recursive definitions of $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ we need to precisely define some of their properties. Let $S^i = \bigcup_{j=0}^{i} S_j$ be the first $(i+1)$ levels of the spanner $S$.

**Definition 1** (*Complete/Incomplete Segments*) The sets $\{\mathsf{I}_\Delta^i\}$ and $\{\mathsf{C}_\Delta^i\}$, where $1 \le \Delta \le \lambda$ and $i \ge 0$, are *valid* with respect to a spanner $S$ if, for any $i$-segment $(x_0, \ldots, x_1, \ldots, x_2, \ldots, x_\Delta)$, divided into $(i-1)$-segments $\{(x_j, \ldots, x_{j+1})\}$, for $0 \le j < \Delta$, at least one of the following two hold:

1. (The segment is **complete**) $\delta_{S^i}(x_0, x_\Delta) \le \mathsf{C}_\Delta^i$
2. (The segment is **incomplete**) For some $0 \le k < \Delta$ the last $k$ $(i-1)$-segments from $(x_\Delta, \ldots, x_{\Delta-1})$ through $(x_{\Delta-k+1}, \ldots, x_{\Delta-k})$ are complete and:
   (a) $\delta(x_{\Delta-k}, V_i) \le \mathsf{I}_\Delta^{i-1}$
   (b) $\delta(x_0, V_{i+1}) \le \mathsf{I}_\Delta^i - k\Delta^{i-1}$

Figure 8 depicts the situation described in Definition 1(2). Lemma 9 defines a set of valid values for $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$.

**Lemma 9** (Recursive Expression of $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$) *The sets* $\{\mathsf{C}_\Delta^i\}_{\Delta,i}$ *and* $\{\mathsf{I}_\Delta^i\}_{\Delta,i}$ *are valid according to Definition 1, defined as follows. For* $i \in \{0,1\}$ *let* $\mathsf{I}_\Delta^0 = 1, \mathsf{I}_\Delta^1 = \Delta + 1, \mathsf{C}_\Delta^0 = 1,$ *and* $\mathsf{C}_\Delta^1 = \Delta + 2$. *For* $i \ge 2$:

$$\mathsf{I}_\Delta^i = 2\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + \Delta^i + (\Delta - 1)\Delta^{i-2}$$

$$\mathsf{C}_\Delta^i = \max \begin{cases} \Delta \cdot \mathsf{C}_\Delta^{i-1} \\ (\Delta - 1) \cdot \mathsf{C}_\Delta^{i-1} + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) + \Delta^{i-1} \end{cases}$$

*Proof* Given a shortest path from $x$ to $x'$ with length $\Delta^i$, where $\Delta < \lambda$ , we can divide it into $\Delta$ $(i-1)$-segments or $\Delta^2$ $(i-2)$-segments, each of which is either complete or incomplete according to Definition 1. Notice that although the graph is undirected, the definition of an incomplete path depends on the orientation of the path. Let $(x = x_0, \ldots, x_1), (x_1, \ldots, x_2), \ldots, (x_{\Delta-1}, x_\Delta = x')$ be the $(i-1)$-segments and, within the $j$th $(i-1)$-segment, let $(x_{j-1} = x_{j-1,0}, \ldots, x_{j-1,1}),\ (x_{j-1,1}, \ldots, x_{j-1,2}), \ldots, (x_{j-1,\Delta-1}, \ldots, x_{j-1,\Delta} = x_j)$ be the $(i-2)$-segments. We begin our

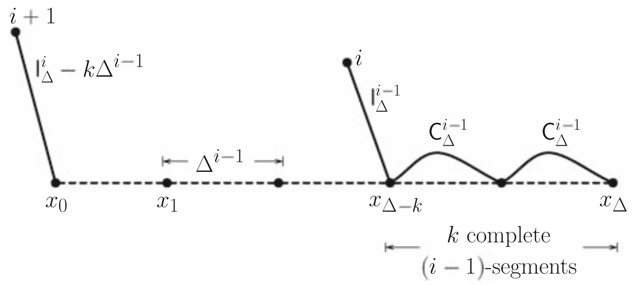

$i+1$
$\mathsf{I}_\Delta^i - k\Delta^{i-1}$
$i$
$\mathsf{I}_\Delta^{i-1}$
$\mathsf{C}_\Delta^{i-1}$
$\mathsf{C}_\Delta^{i-1}$
$\Delta^{i-1}$
$x_0$
$x_1$
$x_{\Delta-k}$
$x_\Delta$
$k$ complete
$(i-1)$-segments

**Fig. 8** A depiction of the guarantees of an incomplete $i$-segment $(x_0, \ldots, x_\Delta)$. Here $(x_{\Delta-k}, \ldots, x_{\Delta-k-1})$ is the first incomplete $(i-1)$-segment starting from $x_\Delta$

attempt to find a complete path from $x$ to $x'$ by identifying the last point $z = x_{j,k'}$ such that all $(i-2)$-segments from $x$ to $z$ are complete, i.e., the distance between their endpoints in $S^{i-2}$ is at most $\mathsf{C}_\Delta^{i-2}$. In a similar fashion, we identify the first point $y' = x_{\Delta-k}$ such that the $k$ $(i-1)$-segments from $x'$ to $y'$ are all complete. Let $y = x_{\Delta-k-1}$. We divide the rest of the proof into three cases depending on the location of $z = x_{j,k'}$ relative to $y$ and $y'$: either (1) $j \ge \Delta - k$ ($z$ is equal to or after $y'$), (2) $j < \Delta - k - 1$ or $j = \Delta - k - 1, k' = 0$ ($z$ is equal to or before $y$) or (3) $j = \Delta - k - 1$ and $k' > 0$ ($z$ is between $y'$ and $y$, as depicted in Fig. 7). We first deal with cases (1–3), assuming $i \ge 2$, then treat the base cases when $i \in \{0, 1\}$.

*Case 1.* In the first case, the whole path is covered by $k$ complete $(i-1)$-segments and $\Delta(\Delta - k)$ complete $(i-2)$-segments. In the worst case $k = \Delta$ and we have:

$$\delta_{S^{i-1}}(x, x') \le \Delta\mathsf{C}_\Delta^{i-1} \tag{6}$$

*Case 2.* Let $p_z = p_{i-1}(z)$, $p_{y'} = p_i(y')$, and $\hat{p}_z = p_{i+1}(p_z)$. Whether the $i$-segment from $x$ to $x'$ is successful hinges on whether $p_{y'} \in \mathscr{B}_{i+1,\lambda}(p_z)$. If so we obtain the following bound on $\delta_{S^i}(x, x')$ and we declare the path from $x$ to $x'$ *complete*.

$$\begin{aligned}\delta_{S^i}(x,x') &\le \delta_{S^{i-2}}(x,x_{j,k'}) + \delta_{S^{i-1}}(x_{j,k'},p_z) + \delta_{S^i}(p_z,p_{y'})\\ &\quad + \delta_{S^i}(p_{y'},y') + \delta_{S^{i-1}}(y',x')\\ &\le (j\Delta + k')\mathsf{C}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-2} + (\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}\\ &\quad + (\Delta - k - j)\Delta^{i-1} - k'\Delta^{i-2}) + \mathsf{I}_\Delta^{i-1} + k\mathsf{C}_\Delta^{i-1}\end{aligned}$$

The expression above is maximized at $k'=0$, and the one below at $k=\Delta-1$ and $j=0$:

$$\begin{aligned}&\le j\Delta\mathsf{C}_\Delta^{i-2} + k\mathsf{C}_\Delta^{i-1} + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) + (\Delta-k-j)\Delta^{i-1}\\ &\le (\Delta-1)\mathsf{C}_\Delta^{i-1} + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) + \Delta^{i-1} \qquad (7)\end{aligned}$$

However, if $p_{y'} \notin \mathscr{B}_{i+1,\lambda}(p_z)$ we declare the path from $x$ to $x'$ to be *incomplete*. We can then guarantee that:

$$\begin{aligned}\delta(x,V_{i+1}) &\le \delta(x,z) + \delta(z,p_z) + \delta(p_z,\hat{p}_z)\\ &\le (j\Delta^{i-1} + k'\Delta^{i-2}) + \mathsf{I}_\Delta^{i-2}\\ &\quad + (\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + (\Delta-k-j)\Delta^{i-1} - k'\Delta^{i-2})\\ &\le \mathsf{I}_\Delta^{i-1} + 2\mathsf{I}_\Delta^{i-2} + \Delta^i - k\Delta^{i-1} \qquad (8)\end{aligned}$$

*Case 3.* As in case 2 let $p_z = p_{i-1}(z)$, $p_{y'} = p_i(y')$, $p_y = p_{i-1}(y)$, and $\hat{p}_y = p_{i+1}(p_y)$. By the incompleteness of the $(i-1)$-segment from $y'$ to $y$ and the completeness of the first $k'(i-2)$-segments from $y$ to $z$ it follows from Definition 1(2) that $\delta(z,p_z) \le \mathsf{I}_\Delta^{i-2}$ and $\delta(y',p_{y'}) \le \mathsf{I}_\Delta^{i-1} - k'\Delta^{i-2}$. See Fig. 8 for a depiction of this case.

By the definition of $S^{i-1}$ and the triangle inequality we have $\delta_{S^{i-1}}(y,p_y) = \delta(y,V_{i-1}) \le \delta(y,z) + \delta(z,p_z) = k'\Delta^{i-2} + \mathsf{I}_\Delta^{i-2}$. Whether the whole path from $x$ to $x'$ is complete depends on whether $p_{y'}$ is in $\mathscr{B}_{i+1,\lambda}(p_y)$. If it is we have:

$$\begin{aligned}\delta_{S^i}(x,x') &\le \delta_{S^{i-2}}(x,y) + \delta_{S^{i-1}}(y,p_y) + \delta_{S^i}(p_y,p_{y'})\\ &\quad + \delta_{S^i}(p_{y'},y') + \delta_{S^{i-1}}(y',x')\\ &\le \Delta(\Delta-k-1)\mathsf{C}_\Delta^{i-2} + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1})\\ &\quad + \Delta^{i-1} + k\mathsf{C}_\Delta^{i-1}\\ &\le (\Delta-1)\mathsf{C}_\Delta^{i-1} + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) + \Delta^{i-1} \qquad (9)\end{aligned}$$

On the other hand, if $p_{y'}$ is *not* in the set $\mathscr{B}_{i+1,\lambda}(p_y)$ then we can conclude that $\delta(p_y,\hat{p}_y) \le \mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + \Delta^{i-1}$ and therefore that

$$\begin{aligned}\delta(x,V_{i+1}) &\le \delta(x,y) + \delta(y,p_y) + \delta(p_y,\hat{p}_y)\\ &\le (\Delta-k-1)\Delta^{i-1} + 2\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + \Delta^{i-1} + k'\Delta^{i-2}\\ &\le \mathsf{I}_\Delta^{i-1} + 2\mathsf{I}_\Delta^{i-2} + \Delta^i + (\Delta-1)\Delta^{i-2} - k\Delta^{i-1} \qquad (10)\end{aligned}$$

The last line is maximized at $k'=\Delta-1$. Taking the maximum over Equations (6,7,9) and (8,10) we can easily write inductive expressions for $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ in terms of $\mathsf{C}_\Delta^{i-1}, \mathsf{I}_\Delta^{i-1}$, and $\mathsf{I}_\Delta^{i-2}$, for any $i \in [2,o]$.

$$\mathsf{C}_\Delta^i \le \max\begin{cases}\Delta\mathsf{C}_\Delta^{i-1}\\ (\Delta-1)\mathsf{C}_\Delta^{i-1} + 2(\mathsf{I}_\Delta^{i-1} + \mathsf{I}_\Delta^{i-2}) + \Delta^{i-1}\end{cases}$$

$$\mathsf{I}_\Delta^i \le \mathsf{I}_\Delta^{i-1} + 2\mathsf{I}_\Delta^{i-2} + \Delta^i + (\Delta-1)\Delta^{k-2}$$

We now prove valid bounds on $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ for $i \in \{0,1\}$. Let $x,x' \in V$ be such that $\delta(x,x') = \Delta^0$, i.e., $(x,x')$ is an edge. If $\delta(x,V_1) > 1$ then $x' \in \mathscr{B}_{1,\lambda}(x)$ and therefore $\delta_{S_0}(x,x') = \delta(x,x') = 1$. That is, for any $\Delta \ge 1$, the assignment $\mathsf{C}_\Delta^0 = \mathsf{I}_\Delta^0 = 1$ is valid. Turning to the case $i=1$, let $(x = x_0,\ldots,x_\Delta = x')$ be a path from $x$ to $x'$ with length $\Delta$. Let $(x_{\Delta-k-1},x_{\Delta-k})$ be the *last* edge not present in $S_0$. That is, if we view the path as oriented from $x'$ to $x$, $(x_{\Delta-k},x_{\Delta-k-1})$ is the first incomplete 0-segment. Using the observation above it follows that $\delta(x_{\Delta-k},V_1) \le \mathsf{I}_\Delta^0 = 1$. Now if $p_1(x_{\Delta-k})$ lies inside $\mathscr{B}_{2,\lambda}(x)$ we declare $(x,\ldots,x')$ complete:

$$\begin{aligned}\delta_{S^1}(x,x') &= \delta_{S^1}(x,p_1(x_{\Delta-k})) + \delta_{S^1}(p_1(x_{\Delta-k}),x_{\Delta-k})\\ &\quad + \delta_{S^0}(x_{\Delta-k},x')\\ &= (\Delta-k+1) + 1 + k = \Delta + 2\end{aligned}$$

If $p_1(x_{\Delta-k})$ lies outside $\mathscr{B}_{2,\lambda}(x)$ we can bound $\delta(x,V_2)$ as: $\delta(x,V_2) \le \delta(x,p_1(x_{\Delta-k})) = \Delta + 1 - k$. Thus, the assignment $\mathsf{C}_\Delta^1 = \Delta + 2$ and $\mathsf{I}_\Delta^1 = \Delta + 1$ is valid for all $\Delta \ge 1$.

□

Finding a closed form bound on $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ is tedious but not at all difficult.

**Lemma 10** (Tight Bounds on $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$) *The sets* $\{\mathsf{C}_\Delta^i\}$ *and* $\{\mathsf{I}_\Delta^i\}$ *given below are valid according to Definition 1, where* $c'_\Delta = 1 + \frac{2\Delta+1}{(\Delta+1)(\Delta-2)}$ *and* $c_\Delta = 3 + \frac{6\Delta-2}{\Delta(\Delta-2)}$.

$$\begin{aligned}&\mathsf{I}_1^i \le \tfrac{2^{i+2}}{3} && \mathsf{C}_1^i \le 2^{i+1}\\ &\mathsf{I}_2^i \le (i+\tfrac{2}{3})2^i + \tfrac{1}{3} && \mathsf{C}_2^i \le 3(i+1)2^i\\ &\mathsf{I}_\Delta^i \le c'_\Delta\Delta^i && \mathsf{C}_\Delta^i \le \min\begin{cases}c_\Delta\Delta^i\\ \Delta^i + 2c'_\Delta i\Delta^{i-1}\end{cases}\end{aligned}$$

*Proof* What follows are six terse induction proofs bounding $\{\mathsf{C}_\Delta^i\}$ and $\{\mathsf{I}_\Delta^i\}$ for $\Delta=1$, $\Delta=2$, and $\Delta\ge 3$. Each has two base cases for $i \in \{0,1\}$. The recursive expressions for $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ come from Lemma 9.

*For* $\Delta = 1$. We claim $\mathsf{I}_1^i = (2^{i+2}-1)/3$ for $i$ even and $\mathsf{I}_1^i = (2^{i+2}-2)/3$ for $i$ odd. As base cases we have: $\mathsf{I}_1^0 = 1 = \frac{2^2-1}{3}$ and $\mathsf{I}_1^1 = 2 = \frac{2^3-2}{3}$. For $i > 1$:

$$\begin{aligned}\mathsf{I}_1^i &= 2\mathsf{I}_1^{i-2} + \mathsf{I}_1^{i-1} + 1 && \{\text{Lemma 9}\}\\ &= \begin{cases}2(\frac{2^i-1}{3}) + \frac{2^{i+1}-2}{3} + 1 & i \text{ even}\\ 2(\frac{2^i-2}{3}) + \frac{2^{i+1}-1}{3} + 1 & i \text{ odd}\end{cases} && \{\text{Ind. hyp.}\}\\ &= \begin{cases}\frac{2^{i+2}-1}{3} & i \text{ even}\\ \frac{2^{i+2}-2}{3} & i \text{ odd.}\end{cases}\end{aligned}$$

We claim $\mathsf{C}_1^i = 2^{i+1}-1$. As base cases we have: $\mathsf{C}_1^0 = 1 = 2^1 - 1$ and $\mathsf{C}_1^1 = 3 = 2^2 - 1$. For $i > 1$:

$$\begin{aligned}\mathsf{C}_1^i &= 2(\mathsf{I}_1^{i-2} + \mathsf{I}_1^{i-1}) + 1 && \{\text{Lemma 9}\}\\ &= 2(\tfrac{2^i}{3} + \tfrac{2^{i+1}}{3} - 1) + 1 = 2^{i+1} - 1 && \{\text{Ind. hyp.}\}\end{aligned}$$

**For** $\Delta = 2$. We claim $\mathsf{I}_2^i = (i + \frac{2}{3})2^i + \frac{(-1)^i}{3}$. As base cases we have: $\mathsf{I}_2^0 = 1 = (0 + \frac{2}{3})2^0 + \frac{(-1)^0}{3}$ and $\mathsf{I}_2^1 = 3 = (1 + \frac{2}{3})2^1 + \frac{(-1)^1}{3}$. For $i > 1$:

$$\begin{aligned}
\mathsf{I}_2^i &= 2\mathsf{I}_2^{i-2} + \mathsf{I}_2^{i-1} + \tfrac{3}{2}2^i && \{\text{Lemma 9}\}\\
&= 2(i - \tfrac{4}{3})2^{i-2} + (i - \tfrac{1}{3})2^{i-1} + \tfrac{(-1)^i}{3} + \tfrac{3}{2}2^i && \{\text{Ind.}\}\\
&= (2i - \tfrac{5}{3})2^{i-1} + \tfrac{3}{2}2^i + \tfrac{(-1)^i}{3}\\
&= (i + \tfrac{2}{3})2^i + \tfrac{(-1)^i}{3}
\end{aligned}$$

For the claimed bounds on $\mathsf{C}_2^i$ we have, as base cases: $\mathsf{C}_2^0 = 3 \le 3 \cdot 1 \cdot 2^0$ and $\mathsf{C}_2^1 = 4 \le 3 \cdot 2 \cdot 2^1$. For $i > 1$:

$$\begin{aligned}
\mathsf{C}_2^i &\le \max\begin{cases} 2\mathsf{C}_2^{i-1}, \\ \mathsf{C}_2^{i-1} + 2(\mathsf{I}_2^{i-2} + \mathsf{I}_2^{i-1}) + \tfrac{3}{2}2^i \end{cases} && \{\text{Lemma 9}\}\\
&\le \max\begin{cases} 3i2^i, \\ 3i2^{i-1} + 2((i - \tfrac{4}{3})2^{i-2} \\ + (i - \tfrac{1}{3})2^{i-1}) + 2^{i-1} \end{cases} && \{\text{Ind. hyp.}\}\\
&\le 2^i(3i/2 + i/2 - 2/3 + i - 1/3 + 1/2)\\
&< 3(i + 1)2^i
\end{aligned}$$

**For** $\Delta \ge 3$. For the claimed bounds on $\mathsf{I}_\Delta^i$ we have, as base cases: $\mathsf{I}_\Delta^0 = 1 \le c'_\Delta \Delta^0$ and $\mathsf{I}_\Delta^1 = \Delta + 1 \le c'_\Delta \Delta^1$. For $i > 1$:

$$\begin{aligned}
\mathsf{I}_\Delta^i &\le 2\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1} + \Delta^i + (\Delta - 1)\Delta^{i-2} && \{\text{Lemma 9}\}\\
&\le \Delta^i(c'_\Delta(\tfrac{2}{\Delta^2} + \tfrac{1}{\Delta}) + 1 + \tfrac{1}{\Delta} - \tfrac{1}{\Delta^2}) && \{\text{Ind. hyp.}\}\\
&= \Delta^i((1 + \tfrac{2\Delta+1}{(\Delta+1)(\Delta-2)})(\tfrac{\Delta+2}{\Delta^2}) + 1 + \tfrac{\Delta-1}{\Delta^2}) && \{\text{Def.}c'_\Delta\}\\
&= \Delta^i(1 + \tfrac{2\Delta+1}{\Delta^2} + \tfrac{(2\Delta+1)(\Delta+2)}{(\Delta+1)(\Delta-2)\Delta^2})\\
&= \Delta^i(1 + \tfrac{(2\Delta+1)[(\Delta+1)(\Delta-2)+\Delta+2]}{(\Delta+1)(\Delta-2)\Delta^2})\\
&= \Delta^i(1 + \tfrac{2\Delta+1}{(\Delta+1)(\Delta-2)})\\
&= c'_\Delta \Delta^i && \{\text{Definition of} c'_\Delta\}
\end{aligned}$$

We first show that $\mathsf{C}_\Delta^i \le c_\Delta \Delta^i$, then that $\mathsf{C}_\Delta^i \le \Delta^i + 2c'_\Delta i \Delta^{i-1}$. For the first bound on $\mathsf{C}_\Delta^i$ our base cases are: $\mathsf{C}_\Delta^0 = 1 \le c_\Delta \Delta^0$ and $\mathsf{C}_\Delta^1 = \Delta + 2 \le c_\Delta \Delta$. For $i > 1$:

$$\begin{aligned}
\mathsf{C}_\Delta^i &\le \max\begin{cases} \Delta\mathsf{C}_\Delta^{i-1}, \\ (\Delta - 1)\mathsf{C}_\Delta^{i-1} + \Delta^{i-1} \\ + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) \end{cases} && \{\text{Lemma 9}\}\\
&\le \max\begin{cases} \Delta c_\Delta \Delta^{i-1}, \\ (\Delta - 1)c_\Delta \Delta^{i-1} + \Delta^{i-1} \\ + 2(c'_\Delta \Delta^{i-2} + c'_\Delta \Delta^{i-1}) \end{cases} && \{\text{Ind. hyp.}\}\\
&\le c_\Delta \Delta^i + \Delta^{i-1}(-c_\Delta + \tfrac{2c'_\Delta}{\Delta} + 2c'_\Delta + 1)\\
&\le c_\Delta \Delta^i \quad \{\text{Definition of } c_\Delta = 1 + 2c'_\Delta + \tfrac{2c'_\Delta}{\Delta}\}
\end{aligned}$$

For the second bound on $\mathsf{C}_\Delta^i$ our base cases are: $\mathsf{C}_\Delta^0 = 1 \le \Delta^0 + 2c'_\Delta \cdot 0 \cdot \Delta^{-1}$ and $\mathsf{C}_\Delta^1 = \Delta + 2 \le \Delta + 2c'_\Delta$. For $i > 1$:

$$\begin{aligned}
\mathsf{C}_\Delta^i &\le \max\begin{cases} \Delta\mathsf{C}_\Delta^{i-1}, \\ (\Delta - 1)\mathsf{C}_\Delta^{i-1} + \Delta^{i-1} \\ \quad + 2(\mathsf{I}_\Delta^{i-2} + \mathsf{I}_\Delta^{i-1}) \end{cases} && \{\text{Lemma 9}\}\\
&\le \max\begin{cases} \Delta^i + 2c'_\Delta(i-1)\Delta^{i-1}, \\ (\Delta - 1)(\Delta^{i-1} + 2c'_\Delta(i-1)\Delta^{i-2}) \\ \quad + \Delta^{i-1} + 2(c'_\Delta \Delta^{i-2} + c'_\Delta \Delta^{i-1}) \end{cases} && \{\text{Ind.}\}\\
&\le \Delta^i + \Delta^{i-1}(2c'_\Delta(i-1) + 2c'_\Delta + \tfrac{2c'_\Delta - 2c'_\Delta(i-1)}{\Delta})\\
&\le \Delta^i + 2c'_\Delta i \Delta^{i-1} \quad \{\text{For } i \ge 2\}
\end{aligned}$$

This concludes the proof of Lemma 10. □

Notice that Lemma 10's bound on $\mathsf{I}_\Delta^o$ is $c'_\Delta \Delta^o$, which is vacuous because $V_{o+1}$ is empty. In other words, *every* $o$-segment (a path with length $\Delta^o$) must be complete. The intuitive reason for this is that every hilltop in $V_{o-1}$ sees every hilltop in $V_o$ at distance $\lambda^o$. Thus, if $\delta(u, v) = \Delta^o$, the multiplicative distortion of $S$ for the pair $u, v$ is $\delta_S(u, v)/\delta(u, v) = \mathsf{C}_\Delta^o/\Delta^o$. With this observation in hand, Theorem 7 easily follows from Lemma 10.

**Theorem 7** *Let $S$ be the Fibonacci spanner generated with parameters $o \in [2, \log_\phi \log n]$, $\epsilon \in (0, 1]$, and $\lambda = 3o/\epsilon + 2$. Then, with high probability,*

$$\mathbb{E}[|S|] = O((o/\epsilon)^\phi n^{1+\frac{1}{F_{o+3}-1}})$$

*and for every pair of vertices $u, v$, if $\delta(u, v) = d$ then $\delta_S(u, v)$ is bounded from above by:*

$$\begin{cases} d \cdot 2^{o+1} & d = 1 \\ d \cdot 3(o + 1) & d = 2^o \\ d \cdot (3 + \frac{6\Delta-2}{\Delta(\Delta-2)}) & d = \Delta^o, \textit{ for } \Delta \ge 3 \\ & d = (3o/\epsilon')^o, \\ d \cdot (1 + \epsilon') & \textit{for any } \epsilon' \in [\epsilon, 1] \end{cases}$$

*Proof* The maximum $\Delta$ we wish to use is $3o/\epsilon$. Note that in the proof of Lemma 9 we often make the inference that if $v \in V_{i-1}$, $u \in V_i$, and $\delta(v, u) \le \lambda^i$, either $\delta_{S^i}(v, u) = \delta(v, u)$ or $\delta(v, p_{i+1}(v)) \le \lambda^i$. Here $\lambda^i$ is never more than $\mathsf{C}_\Delta^i$ or $\mathsf{I}_\Delta^i$. One can easily check (see Lemma 10) that $\mathsf{C}_\Delta^i$ and $\mathsf{I}_\Delta^i$ are at most $(\Delta + 2)^i$. Thus, the analysis of Lemma 9 and 10 goes through as long as $\Delta \le \lambda - 2$. This justifies our choice of $\lambda$.

The first three lines in the claimed bound on $\delta_S(u, v)$ follow directly from Lemma 10. We prove the last line; in particular, why the choice of $\Delta = 3o/\epsilon'$ gives a multiplicative distortion of $1 + \epsilon'$. By Lemma 10, if $\delta(v, u) = \Delta^o =$

$(3o/\epsilon')^o$ it follows that:

$$\begin{aligned}\delta_S(v,u) &\le (3o/\epsilon')^o + 2c'_\Delta o(3o/\epsilon')^{o-1}\\ &= \delta(v,u)\left(1+\frac{2c'_\Delta o}{3o/\epsilon'}\right)\\ &= \delta(v,u)\left(1+\left(\frac{2\epsilon'}{3}\right)\left(1+\frac{2\Delta+1}{(\Delta+1)(\Delta-2)}\right)\right)\\ &\qquad\qquad \{\text{Def. of} c'_\Delta\}\\ &\le \delta(v,u)\left(1+\left(\frac{2\epsilon'}{3}\right)\left(1+\frac{13}{28}\right)\right)\\ &\qquad\qquad \{\Delta \ge 3o \ge 6\}\\ &< \delta(v,u)\left(1+\epsilon'\right)\end{aligned}$$

□

We think the most interesting parametrization of Theorem 7 is when the order $o = \log_\phi \log n - O(1)$, yielding a nearly linear size spanner. However, in this case the distortion for very close vertices is $2^{o+1} = 2^{\log_\phi \log n - O(1)} \approx (\log n)^{1.44}$. Theorem 2 will give us an $O(\frac{\log n}{\log\log\log n})$-spanner with size $O(n \log\log n)$. By including such a spanner with a Fibonacci spanner we obtain the distortion bounds stated in Corollary 1.

**Corollary 1** *Any graph on $n$ vertices contains a spanner with size $O(n(\epsilon^{-1}\log\log n)^\phi)$ such that for any vertices $u, v$, if $\delta(u,v) = d$ then $\delta_S(u,v)$ is bounded from above by*:

$$\begin{cases} d\cdot O(\frac{\log n}{\log\log\log n}) & d \ge 1\\ d\cdot 3(\log_\phi \log n) & d \ge (\log n)^{\log_\phi 2} \approx (\log n)^{1.44}\\ d\cdot(3+\frac{6\Delta-2}{\Delta(\Delta-2)}) & d \ge (\log n)^{\log_\phi \Delta},\ \textit{for } \Delta \ge 3\\ & d = (3o/\epsilon')^o,\\ d\cdot(1+\epsilon') & \textit{for any } \epsilon' \in [\epsilon, 1]\end{cases}$$

*Proof* The parameters of the spanner are $o = \log_\phi \log n - 2$ and $\lambda = 3o/\epsilon + 2$. Note that the bounds of Theorem 7 hold for distances that are of the form $\Delta^o$ for some integer $\Delta \le \lambda - 2$. Of course, in general $d^{1/o}$ may not be an integer and $d$ may be much greater than $(\lambda-2)^o$. We choose not to pursue a detailed analysis for such distances. If $d$ is not a nice number we round it up to $\lceil d^{1/o}\rceil^o$ and apply Theorem 7. If $d$ is too large to apply Theorem 7 directly (i.e., if $d > (3o/\epsilon)^o$) we simply chop it up into $\lceil d/(3o/\epsilon)^o\rceil$ pieces with length at most $(3o/\epsilon)^o$ and apply Theorem 7 separately to each piece. □

## 4.4 Distributed construction

Our spanner is composed of a collection of shortest paths that is determined solely by the initial random sampling. In the first stage of the algorithm every node $v$ must, for each $i \in [1, o]$, ensure that a path $P(v, p_i(v))$ is included in the spanner or determine that $\delta(v, V_i) > \lambda^{i-1}$. This stage can be executed in $\lambda^{i-1}$ time steps using unit-length messages. In the first step each vertex in $V_i$ notifies its neighbors that it is in $V_i$. In general, in the $k$th step each vertex $v$ receives a message from each neighbor $w$ indicating the $V_i$-vertex with the minimum unique identifier at distance $k-1$ from $w$. In the $(k+1)$th step $v$ sends the minimum among these $V_i$-vertices to all neighbors that it has yet to receive a message from. Thus, after $\lambda^{i-1}$ steps each $v \in V$ knows the first edge on the path $P(v, p_i(v))$ or knows that $\delta(v, V_i) \ge \lambda^{i-1}$.

In the second stage we must connect each $x \in V = V_0$ with each $y \in \mathscr{B}_{1,\lambda}(x)$ and each $x \in V_{i-1}$ with each $y \in \mathscr{B}_{i+1,\lambda}(x)$. Since the radius of any of these balls is at most $\lambda^o$, this stage is trivial to execute in $O(\lambda^o)$ time with unbounded messages. However, if our budget constrains us to messages with length $\tilde{O}(n^{1/t})$ we need to be more careful.

Consider the following protocol for computing $S_i$. Each node $y \in V_i$ will *attempt* to broadcast its identity to all nodes at distance at most $\lambda^i$. In the $k$th step each $z \in V$ will receive a message from each neighbor $w$ listing all $V_i$-nodes at distance $k-1$ from $w$. (Thus, in the $(k+1)$th step $z$ will send to $w$ all $V_i$-nodes received in the $k$th step, excluding those already known to $w$.) If $z$ ever needs to send a message longer than the maximum $\tilde{O}(n^{1/t})$ it *ceases* participation in this protocol. After $\lambda^i$ time each $x \in V_{i-1}$ will know the set of $V_i$-vertices at distance $\lambda^i$, excluding those that were blocked due to cessation of nodes. In other words, this protocol will work correctly if every "blocked" $y \in V_i$ at distance $\delta(x,y) \le \lambda^i$ does not appear in $\mathscr{B}_{i+1,\lambda}(x)$. We claim that with probability $1 - n^{-\Omega(1)}$ this protocol works correctly if the message length is bounded at $s \ge 4q_{i+1}^{-1} q_i \ln n$. Consider the situation from the point of view of $x \in V_{i-1}$. Let $y \in \mathscr{B}_{i+1,\lambda}(x)$ and suppose that $z$ blocked $x$ from becoming aware of $y$ by ceasing to participate after the $k$th step, because it failed to transmit the set $\{y, y_1, \ldots, y_s\}$ to a neighbor (See Fig. 9). If follows that for any $y' \in \{y_1, \ldots, y_s\}$, $\delta(x,y') < \delta(x, V_{i+1})$ since $\delta(x,y') \le \delta(x,z) + k = \delta(x,y) < \delta(x, V_{i+1})$. Thus, if the protocol fails then $|\mathscr{B}_{i+1,\lambda}(x)| > s$. Since nodes in $V_i$ are promoted to $V_{i+1}$ with probability $q_{i+1}/q_i$ the probability of failure (from $x$'s point of view) is at most $(1 - q_{i+1}/q_i)^{s+1} < e^{-sq_{i+1}/q_i} < n^{-4}$. Since $\sum_i |V_i| < 2n$ the total probability of failure is certainly less than $2n^{-3}$.

This Monte Carlo protocol can easily be converted to a Las Vegas one as follows. Suppose that a vertex $z$ ceased participation in the construction of $S_i$ after step $k$. It will then broadcast the number $k$ to all vertices at distance at most $\lambda^i$. If an $x \in V_{i-1}$ receives such a message and $\delta(x,z) +$

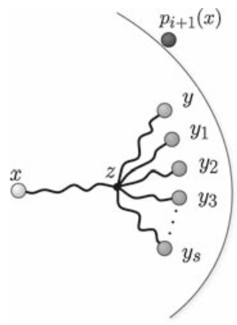


**Fig. 9** If a message *should* be transmitted from $y \in \mathscr{B}_{i+1,\lambda}(x)$ to $x \in V_{i-1}$, only messages from other nodes in $\mathscr{B}_{i+1,\lambda}(x)$ could possibly interfere with it

$k < \delta(x, V_{i+1})$ then $x$ has detected a failure. It broadcasts a message commanding *all* vertices at distance at most $\lambda^i$ to include *all* adjacent edges in the spanner. This error detection mechanism can be implemented with unit-length messages and increases the expected size of the spanner by at most $\binom{n}{2} \cdot 2n^{-3} < 1/n$.

Under the usual choice of sampling probabilities the most expensive of $S_0, \ldots, S_o$ to compute is $S_o$, which requires $O(\lambda^o)$ time and messages with length:

$$\tilde{O}(nq_o) = \tilde{O}(n^{1-\frac{F_{o+2}-1}{F_{o+3}-1}} \lambda^{-\phi(F_{o+2}-1)+F_{o+3}-(o+2)}) \approx n^{1-1/\phi} \approx n^{.38}.$$

If $n^{.38}$ is too much we can simply increase the order $o$ of the spanner, which gives us the freedom to tighten the gap between consecutive sampling probabilities $q_i$ and $q_{i+1}$. In particular, if our desired spanner size is $O(n^{1+\alpha}\lambda^{\phi})$ and our desired message size is $\tilde{O}(n^{1/t}) > n^{\alpha}$, we find the maximum $i$ such that $q_i/q_{i+1} \le n^{1/t}$. The sampling probabilities $q_1, \ldots, q_{i+1}$ are kept as is and the remaining sampling probabilities are replaced with $\{q'_j\}_{j>i+1}$, where $q'_j = q_{i+1}n^{(-j+i+1)/t}$. The overall effect of limiting the message size to $\tilde{O}(n^{1/t})$ is to increase the order $o$ by at most $t$. Theorem 8 follows immediately from the above discussion, Theorem 7, and the construction of sparse multiplicative spanners from Section 2.

**Theorem 8** *Consider a synchronized distributed network in which messages of length $\tilde{O}(n^{1/t})$ can be communicated at each time step. Let $o$ be an integer such that $2 \le o \le \log_{\phi}\log n$ and $F_{o+3} > t$, and let $\lambda = 3(o+t)/\epsilon + 2$. A spanner $S$ with expected size $O(\lambda^{\phi} n^{1+\frac{1}{F_{o+3}-1}})$ can be constructed in $O(\lambda^{o+t})$ time such that for any $u, v$ with $\delta(u,v) = d$, $\delta_S(u,v)$ is bounded from above by:*

$$\begin{cases} d \cdot O(\min\{\frac{\log n}{\log\log\log n}, F_{o+3}\}) & d \ge 1 \\ d \cdot 3(o+t+1) & d \ge 2^{o+t} \\ d \cdot (3 + \frac{6\Delta-2}{\Delta(\Delta-2)}) & d \ge \Delta^{o+t}, \Delta \ge 3 \\ & d \ge \left(\frac{3(o+t)}{\epsilon'}\right)^{o+t}, \\ d \cdot (1+\epsilon') & \text{for any } \epsilon' \in [\epsilon, 1] \end{cases}$$

*Note that in the first bound, $2^{\log^* n - \log^* |S|/n} \log_{|S|/n} n = O(F_{o+3})$.*

The introduction stated the results of Theorem 8 for the special case of $o = \log_{\phi}\log n$, using the $(\alpha, \beta)$-spanner terminology. We restate these claims:

**Corollary 2** *Consider a synchronized distributed network in which messages of length $\tilde{O}(n^{1/t})$ can be communicated at each time step. For $\lambda = 3(\log_{\phi}\log n + t)/\epsilon + 2$, a spanner $S$ with expected size $O(n\lambda^{\phi})$ can be constructed in $O(\lambda^{\log_{\phi}\log n + t})$ time such that $S$ is simultaneously:*

- *an $O(\frac{\log n}{\log\log\log n})$-spanner (see Theorem 2),*
- *a $(3(\log_{\phi}\log n + t), \beta_1)$-spanner,*
- *a $(3+\rho, \beta_2)$-spanner,*
- *and a $(1+\epsilon', \beta_3)$-spanner, for any $\epsilon' \ge \epsilon$.*

*Here $\beta_1 = 2^t(\log n)^{\log_{\phi} 2}$, $\beta_3 = (\frac{3(\log_{\phi}\log n + t)}{\epsilon'})^{\log_{\phi}\log n + t}$, and for any parameter $\Delta \ge 3$, $\rho = \frac{6\Delta-2}{\Delta(\Delta-2)}$ and $\beta_2 = \Delta^{\log_{\phi}\log n + t}$.*

## 5 Conclusion

In this paper, we developed new randomized algorithms for constructing spanners with linear or near-linear size that use relatively short messages. Furthermore, we have shown that in any reasonable amount of time the best spanners one could hope to find are of the $(1+\epsilon, \beta)$ variety: there are no efficient constructions of (sublinear) additive spanners unless the density or additive distortion is very large.

Our results leave open several difficult problems. Can a linear size $O(\log n)$-spanner (or linear size skeleton) by constructed in $O(\log n)$ time with unit-length messages, and more generally, is it possible to find $(2k-1)$-spanners distributively (for arbitrary, not necessary constant $k$) with size $O(n^{1+1/k})$? Baswana and Sen claimed that their randomized distributed algorithm finds $(2k-1)$-spanners whose expected size is $O(kn + n^{1+1/k})$, a bound that would be optimal (assuming the girth conjecture) for $k < \log n/\log\log n$. However, in Sect. 2 we found and corrected a flaw in their analysis, which yields the slightly weaker bound of $O(kn + (\log k)n^{1+1/k})$ on the expected size of the spanner. The best deterministic distributed $(2k-1)$-spanner algorithm [15] returns slightly denser spanners, with size $O(kn^{1+1/k})$.

Perhaps the most interesting applications of spanners are in constructing distance labeling schemes [26,38], approximate distance oracles [6,9,27,35,38], and compact routing tables [1,11,12,31,37,34] that guarantee approximately shortest routes. Nearly all results in these areas deal exclusively with purely multiplicative distortion, i.e., distortion not of the $(\alpha, \beta)$ variety. It is not clear whether one could find approximate distance oracles (and labelings, and routing tables) whose space-distortion tradeoffs match those of the best spanners. As a first step one might look for *any* nontrivial tradeoff that beats the standard *girth bound* for purely multiplicative distortion. For example, is it possible to stock the nodes of an unweighted graph with $O(n^{1-\epsilon})$-size routing tables such that if $\delta(u,v) = d$, the route taken from $u$ to $v$ has length $(3-\epsilon)d + \text{polylog}(n)$?

**Acknowledgments** We would like to thank the anonymous referees for spotting several errors and greatly improving the readability of the paper.

## References

1. Abraham, I., Gavoille, C., Malkhi, D.: On space-stretch trade-offs: upper bounds. In: Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 217–224 (2006)
2. Abraham, I., Gavoille, C., Malkhi, D., Nisan, N., Thorup, M.: Compact name-independent routing with minimum stretch. In: Proceedings of the 16th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA) pp. 20–24 (2004)
3. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast estimation of diameter and shortest paths (without matrix multiplication). SIAM J. Comput. **28**(4), 1167–1181 (1999)
4. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete Comput. Geom. **9**, 81–100 (1993)
5. Baswana, S.: Streaming algorithm for graph spanners—single pass and constant processing time per edge. Inf. Proc. Lett. **106**(3), 110–114 (2008)
6. Baswana, S., Kavitha, T.: Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In: Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 591–602. (2006)
7. Baswana, S., Kavitha, T., Mehlhorn, K., Pettie, S.: Additive spanners and $(\alpha, \beta)$-spanners. ACM Trans. on Algorithms (2009)
8. Baswana, S., Sarkar, S.: Fully dynamic algorithms for graph spanners with poly-logaritmic update time. In: Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1125–1134 (2008)
9. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in expected $O(n^2 \log n)$ time. ACM Trans. Algorithms **2**(4), 557–577 (2006)
10. Baswana, S., Sen, S.: A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. J. Random Struct. Algs. **30**(4), 532–563 (2007)
11. Cowen, L.J.: Compact routing with minimum stretch. J. Algor. **28**, 170–183 (2001)
12. Cowen, L.J., Wagner, C.G.: Compact roundtrip routing in directed networks. J. Algor. **50**(1), 79–95 (2004)
13. Derbel, B., Gavoille, C.: Fast deterministic distributed algorithms for sparse spanners. Theor. Comput. Sci. **399**(1–2), 83–100 (2008)
14. Derbel, B., Gavoille, C., Peleg D.: Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In: Proceedings of the 21st International Symposium on Distributed Computing (DISC) pp. 179–192. (2007)
15. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: On the locality of distributed sparse spanner construction. In: Proceedings 27th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 273–282. (2008)
16. Derbel, B., Gavoille, C., Peleg, D., Viennot, L.: Local computation of nearly additive spanners. In: Proceedings 23rd International Symposium on Distributed Computing (DISC), LNCS volume 5805, pp. 176–190. (2009)
17. Dor, D., Halperin, S., Zwick, U.: All-pairs almost shortest paths. SIAM J. Comput. **29**(5), 1740–1759 (2000)
18. Dubhashi, D., Mei, A., Panconesi, A., Radhakrishnan, J., Srinivasan, A.: Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. J. Comput. Syst. Sci. **71**(4), 467–479 (2005)
19. Elkin, M.: Computing almost shortest paths. ACM Trans. Algorithms **1**(2), 283–323 (2005)
20. Elkin, M.: A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. In: Proceedings of the 26th International Symposium on Principles of Distributed Computing (PODC), pp. 185–194 (2007)
21. Elkin, M.: Streaming and fully dynamic centralized algorithms for maintaining sparse spanners. In: Proceedings of the 34th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 716–727 (2007)
22. Elkin, M., Peleg, D.: $(1 + \epsilon, \beta)$-spanner constructions for general graphs. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 173–182 (2001)
23. Elkin, M., Peleg, D.: $(1 + \epsilon, \beta)$-spanner constructions for general graphs. SIAM J. Comput. **33**(3), 608–631 (2004)
24. Elkin, M., Zhang, J.: Efficient algorithms for constructing $(1 + \epsilon, \beta)$-spanners in the distributed and streaming models. Distrib. Comput. **18**(5), 375–385 (2006)
25. Erdős, P.: Extremal problems in graph theory. In Theory of Graphs and its Applications (Proceedings of Symposium Smolenice, 1963), pp. 29–36. Publ. House Czechoslovak Acad. Sci., Prague (1963)
26. Gavoille, C., Peleg, D., Pérennes, S., Raz, R.: Distance labeling in graphs. J. Algor. **53**, 85–112 (2004)
27. Mendel, M., Naor, A.: Ramsey partitions and proximity data structures. J. Eur. Math. Soc **9**(2), 253–275 (2007)
28. Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press, Cambridge, England (2007)
29. Panconesi, A.: Personal communication (2008)
30. Peleg, D.: Distributed computing: a locality-sensitive approach. SIAM (2000)
31. Peleg, D., Upfal, E.: A trade-off between space amd efficiency for routing tables. J. ACM **36**(3), 510–530 (1989)
32. Pettie, S.: Low distortion spanners. ACM Trans. on Algorithms (to appear)
33. Pettie, S.: Low distortion spanners. In: Proceedings of the 34th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 78–89 (2007)
34. Roditty, L., Thorup, M., Zwick, U.: Roundtrip spanners and roundtrip routing in directed graphs. In: Proceedings of the 13th ACM-SIAM Symposium On Discrete Algorithms (SODA), pp. 844–851 (2002)
35. Roditty, L., Thorup, M., Zwick, U.: Deterministic constructions of approximate distance oracles and spanners. In: Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP), pp. 261–272 (2005)
36. Roditty, L., Zwick, U.: On dynamic shortest paths problems. In: Proceedings of the 12th Annual European Symposium on Algorithms (ESA), pp. 580–591 (2004)
37. Thorup, M. Zwick, U.: Compact routing schemes. In: Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 1–10 (2001)
38. Thorup, M., Zwick, U.: Approximate distance oracles. J. ACM **52**(1), 1–24 (2005)
39. Thorup, M., Zwick, U.: Spanners and emulators with sublinear distance errors. In: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 802–809 (2006)
40. Wenger, R.: Extremal graphs with no $C^4$s, $C^6$s, or $C^{10}$s. J. Comb. Theory Ser. B **52**(1), 113–116 (1991)
41. Woodruff, D.: Lower bounds for additive spanners, emulators, and more. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 389–398 (2006)