Very Sparse Additive Spanners and Emulators

Greg Bodwin
Department of Computer Science
Stanford University
Stanford, CA
gbodwin@cs.stanford.edu

Virginia Vassilevska Williams
Department of Computer Science
Stanford University
Stanford, CA
virgi@cs.stanford.edu

ABSTRACT

We obtain new upper bounds on the additive distortion for graph emulators and spanners on relatively few edges. We introduce a new subroutine called "strip creation," and we combine this subroutine with several other ideas to obtain the following results:

1. Every graph has a spanner on $O(n^{1+\epsilon})$ edges with $\tilde{O}(n^{1/2-\epsilon/2})$ additive distortion.
2. Every graph has an emulator on $\tilde{O}(n^{1+\epsilon})$ edges with $\tilde{O}(n^{1/3-2\epsilon/3})$ additive distortion whenever $\epsilon \in [0, \frac{1}{5}]$.
3. Every graph has a spanner on $\tilde{O}(n^{1+\epsilon})$ edges with $\tilde{O}(n^{2/3-5\epsilon/3})$ additive distortion whenever $\epsilon \in [0, \frac{1}{4}]$.

Our first spanner has the new best known asymptotic edge-error tradeoff for additive spanners whenever $\epsilon \in [0, \frac{1}{7}]$. Our second spanner has the new best tradeoff whenever $\epsilon \in [\frac{1}{7}, \frac{3}{17}]$. Our emulator has the new best asymptotic edge-error tradeoff whenever $\epsilon \in [0, \frac{1}{5}]$.

1. INTRODUCTION

A spanner of a graph G is a sparser subgraph of G over the same vertex set that approximately preserves all pairwise distances between nodes. An emulator of a graph G is a possibly weighted graph H on the same vertex set as G such that the pairwise distances in H approximate the pairwise distances in G. Researchers try to improve the tradeoff between the sparsity of the spanner/emulator and the accuracy with which it preserves the distances of the original graph. Spanners were first introduced in the 1980s, where they were used to speed up protocols run over unsynchronized networks [3, 20]. Emulators were introduced by Dor, Halperin and Zwick [13]. Spanners and emulators have since found a wide variety of applications, including compact routing schemes [10, 11, 21, 23, 24], almost-shortest path algorithms [16, 14, 15, 13], distance oracles [25, 7, 4, 23], broadcasting [18], and many others.

ITCS'15, January 11–13, 2015, Rehovot, Israel.

http://dx.doi.org/10.1145/2688073.2688103.

Much of the initial theoretical work on spanners studied *multiplicative* stretch spanners, that is, all pairwise distances are preserved up to a small multiplicative factor. Althofer et al. [2] discovered that there exist spanners on $O(n^{1+1/k})$ edges with multiplicative stretch $2k-1$ for all integers $k \geq 1$. This upper bound is tight for multiplicative spanners assuming an unproven conjecture of Erdös [17]. Recent work has focused more on mixed and additive spanners. An additive spanner preserves the distances within a small additive error, and mixed spanners allow both a multiplicative and an additive error.

There are three known constructions that produce spanners of constant additive error. Aingworth et al. [1] gave a construction for spanners on $O(n^{3/2})$ edges with additive error of 2, Chechik [9] showed how to construct spanners on $\tilde{O}(n^{7/5})$[1] edges with additive distortion of 4, and Baswana et al. [6, 5] gave a construction for spanners on $\tilde{O}(n^{4/3})$ edges with additive distortion of 6 (the runtime of this construction was improved by Woodruff [27]). Knudsen [19] later simplified the constructions of $+2$ and $+6$ spanners while obtaining the same edge bounds. Dor, Halperin and Zwick [13] obtain an additive emulator on $\tilde{O}(n^{4/3})$ edges and $+4$ distortion. It remains a major open problem whether or not all graphs admit spanners *or* emulators of constant additive distortion with $\tilde{O}(n^{4/3-\epsilon})$ edges for any $\epsilon > 0$. Notably, Woodruff [26] has proven the existence of a graph family for which any spanner of $2k-1$ additive distortion has at least $O(k^{-1}n^{1+1/k})$ edges, so we cannot hope for $\tilde{O}(n)$ edge spanners with constant additive distortion.

In light of this, some attempts have been made to produce relatively efficient spanners below the $n^{4/3}$ threshold. Bollobás et al. [8] showed a construction for spanners on $O(2^{1/\epsilon}n^{1+\epsilon})$ edges and $n^{1-2\epsilon}$ additive distortion; the distortion was later improved to $O(n^{1-3\epsilon})$ by Baswana et al [6]. Pettie [22] achieved $n^{9/16-7\epsilon/8}$ distortion, and Chechik [9] achieved $n^{1/2-3\epsilon/2}$ distortion whenever $\epsilon \in [\frac{3}{17}, \frac{1}{3}]$. Jointly, these last two spanners form the current state of the art. The construction of Baswana et al [6] can be generalized in a straightforward way to produce emulators with $n^{1/2-3\epsilon/2}$ additive distortion for any $\epsilon \in [0, \frac{1}{3}]$ (they do not discuss this explicitly in their paper), but no other results for emulators are known.

Our Work.

We introduce a new subroutine that is useful for spanner/emulator construction called "strip creation," and we

[1]The notation $\tilde{O}(f(n))$ suppresses poly $\log n$ factors.

apply this subroutine to obtain some significantly improved upper bounds on the accuracy/sparsity tradeoff available for spanners and emulators below the $n^{4/3}$ edge threshold. In Section 3, we use this subroutine in a straightforward way to produce a spanner on $O(n^{1+\epsilon})$ edges with $\tilde{O}(n^{1/2-\epsilon/2})$ additive distortion. In Section 4, we merge this idea with some others to achieve an emulator construction with $\tilde{O}(n^{1/3-2\epsilon/3})$ additive distortion, so long as $\epsilon \in [0, \frac{1}{5}]$, and a spanner of additive distortion $\tilde{O}(n^{2/3-5\epsilon/3})$, so long as $\epsilon \in [0, \frac{1}{4}]$.

Our emulator has the best tradeoff whenever $\epsilon \in [0, \frac{1}{5})$ (it ties the generalization of Baswana et al [6], or Chechik's spanner [9], at $\epsilon = \frac{1}{5}$). The state-of-the-art asymptotics for purely additive spanners are summarized in the following table:[2]

Author	Distortion	Best When
Before this paper		
Pettie [22]	$\tilde{O}(n^{9/16-7\epsilon/8})$	$\epsilon \in [0, \frac{3}{17}]$
Chechik [9]	$\tilde{O}(n^{1/2-3\epsilon/2})$	$\epsilon \in [\frac{3}{17}, \frac{1}{3}]$
After this paper		
New; Section 3	$\tilde{O}(n^{1/2-\epsilon/2})$	$\epsilon \in [0, \frac{1}{7}]$
New; Section 4	$\tilde{O}(n^{2/3-5\epsilon/3})$	$\epsilon \in [\frac{1}{7}, \frac{3}{17}]$
Chechik [9]	$\tilde{O}(n^{1/2-3\epsilon/2})$	$\epsilon \in [\frac{3}{17}, \frac{1}{3}]$

Preliminaries.

All graphs in this paper are undirected and unweighted. The number of nodes in all of our graphs is n, unless otherwise stated. For a graph $G = (V, E)$ and $u, v \in V$, we denote by $\delta_G(u, v)$ the length of the shortest path in G from u to v.

We will often refer to *the* shortest path between two nodes in a graph, when in fact there may be many equally short paths. Any shortest path may be chosen for each pair of nodes, as long as (1) the choice is consistent, and (2) the paths are as nested as possible - that is, any two shortest paths intersect on at most one subpath. We will use this second property in our constructions. We will use the notation $\rho_G(u, v)$ to refer to the chosen shortest path between u and v in G.

Given a graph G, spanners and emulators are sparser versions of G that approximately preserve shortest path distance between every pair of nodes. More formally:

DEFINITION 1. *A weighted, undirected graph $H = (V, E', w)$ is an* (α, β)-emulator *of another graph $G = (V, E)$ (on the same vertex set) if, for all nodes $u, v \in V$, we have*

$$\delta_G(u, v) \leq \delta_H(u, v) \leq \alpha\delta_G(u, v) + \beta.$$

DEFINITION 2. *An* unweighted (α, β)*-emulator of a graph G is called an* (α, β)-spanner *of G if it is a subgraph of G.*

When $\alpha = 1$, we say that the spanner/emulator is *additive* with *distortion* β, and when $\beta = 0$, we say that the spanner/emulator is *multiplicative* with *stretch* α. If neither of these is the case, then the spanner/emulator is *mixed*.

[2]Some authors consider spanners whose error is a function of d, the original distance between the nodes, rather than n; these results are not included in our table.

2. STRIP CREATION

In this section, we will describe our main subroutine, called "strip creation."

Graph Clustering.

This is an important auxiliary subroutine. A very similar subroutine to this one has been used in many different spanner and emulator constructions (see [13, 9] for example). The input is a graph $G = (V, E)$ and an integer e, and the output is a partial partitioning of V into "clusters." The algorithm works as follows:

Algorithm 1: cluster(G, e)

Data: A graph $G = (V, E)$ and an integer e

```
Initialize C = ∅;
Unmark all nodes;
while there is an unmarked node u with at least e − 1 unmarked neighbors do
    Initialize C to be the set u plus any e − 1 of its unmarked neighbors;
    Mark all nodes in C;
    Add C to C
end
return C;
```

Intuitively, we think each $C \in \mathcal{C}$ as a "cluster" of nodes. Each cluster contains a node at distance one from all other nodes; this is called the "cluster center." The subroutine of generating these clusters will be called **cluster**(G, e). The subroutine of picking out the center of a given cluster will be called **center**(C). A node is *clustered* if it belongs to some $C \in \mathcal{C}$ and *unclustered* otherwise.

Our version of graph clustering differs slightly from the typical clustering algorithm used in other spanner constructions. We produce clusters that necessarily have at least e nodes each (this property will be important later). The other important feature is that there are at most ne edges in the graph between unclustered nodes:

CLAIM 1. *Let $\mathcal{C}$ be the output of* **cluster**(G, e)*. There are at most ne edges in G with both endpoints unclustered in $\mathcal{C}$.*

PROOF. First, note that at termination of the **cluster** subroutine, no node can have at least $e-1$ unmarked neighbors: if it did, then we could turn it into a new cluster and add it to $\mathcal{C}$. Therefore, we have at most $e - 1$ edges with both endpoints unclustered incident on any given unclustered node. We have at most n unclustered nodes in total, and the claim follows. □

Strip Creation.

This is our main subroutine. We will add a carefully-chosen set of shortest paths to the spanner that collectively have some convenient coverage properties. The subroutine works as follows:

Algorithm 2: createStrips$(G, \mathcal{C}, d, m)$

Data: A graph $G = (V, E)$, a clustering $\mathcal{C}$ of G, and integers d, m.

1 Initialize a set $\mathcal{S} = \emptyset$;
2 **while** *there exist* $u, v \in V$ *such that the following properties all hold:*

1. $\delta_G(u, v) \leq d$
2. $\rho_G(u, v)$ *intersects at most* m *different paths in* $\mathcal{S}$
3. $\rho_G(u, v)$ *intersects exactly* m *clusters that are disjoint from all paths in* $\mathcal{S}$

3 **do**
4 | Add $\rho_G(u, v)$ to $\mathcal{S}$;
5 **end**
6 **return** $\mathcal{S}$;

The paths in $\mathcal{S}$ are the *strips* of the graph. This subroutine will be called **createStrips**$(G, \mathcal{C}, d, m)$.

One property that makes this subroutine useful is that it is very cheap to add the edges in $\mathcal{S}$ to our spanner. This is our first lemma.

LEMMA 1. *The set* $\mathcal{S}$ *contains only* $O(n)$ *edges that are incident on a clustered node.*

PROOF. When we add a new path $\rho_G(u, v)$ to $\mathcal{S}$, we sort its edges (a, b) into the following cases:

1. The nodes a and b are both unclustered: this edge is not incident on a clustered node, so we can ignore it.
2. There is no edge already in $\mathcal{S}$ that touches a (or the same condition holds for b): there are $O(n)$ of this type of edge in total.
3. There is an edge in $\mathcal{S}$ that touches a and another edge in $\mathcal{S}$ that touches b, but no edge (a, b): this type of edge must go between two strips that are already in $\mathcal{S}$. Since two shortest paths intersect on at most one subpath, there can be at most two of these edges per path in $\mathcal{S}$ that intersects $\rho_G(u, v)$. Therefore, the number of this type of edge in $\rho_G(u, v)$ is at most $2m$. There are at most $\frac{n}{m}$ paths added to $\mathcal{S}$ in total (since each one must be the first to touch m new clusters), so the total number of this type of edge in the graph is $O(n)$.
4. The edge (a, b) is already in $\mathcal{S}$: this does not add a new edge to $\mathcal{S}$, so we can ignore it.

□

Adding the edges of a strip set to our graph gives it some convenient connectivity properties. This is the subject of our next two lemmas.

DEFINITION 3. *A cluster is* clean *if it does not share a node with any strip* $S \in \mathcal{S}$.

LEMMA 2. *Let* H *be a subgraph of* G *over the same vertex set, and suppose* H *contains exactly the edges of* **multspan**(G) *and* **createStrips**$(G, \mathcal{C}, d, m)$. *Let* u, v *be nodes in* G *with the property that* $\rho_G(u, v)$ *intersects at most* k *strips and at most* k *clean clusters. Then* $\rho_H(u, v) \leq \rho_G(u, v) + \tilde{O}(k)$.

PROOF. Construct a new path P as follows:

1. Start at u.
2. Repeat until you reach v:
   (a) Walk down $\rho_G(u, v)$ until you encounter a node x that belongs to a non-clean cluster C. Let S be a strip that intersects C. Let $s \in S \cap C$.
   (b) Travel the shortest path from x to s.
   (c) Walk along S until the last cluster C' intersected by both S and $\rho_G(u, v)$. Let $s' \in S \cap C'$. Let $x' \in \rho_G(u, v) \cap C'$.
   (d) Travel the shortest path from s' to x'.
   (e) Walk along $\rho_G(u, v)$ until you exit C'.

First, we will argue that P is only $O(k)$ longer than $\rho_G(u, v)$. Each time P departs from $\rho_G(u, v)$, we travel distance at most two (from x to s, which belong to the same cluster), and when P rejoins $\rho_G(u, v)$ we again travel distance at most two (from s' to x', which again belong to the same cluster). Let $a \in \rho_G(u, v)$ be a node immediately before one of these departures, and let $b \in \rho_G(u, v)$ be a node immediately after one of these departures. It follows from the triangle equality that $|P[a \leftrightarrow b]| \leq \delta_G(a, b) + 8$. Since we assume there are at most k departures in total, this implies that $|P| \leq \delta_G(a, b) + 8k$.

Second, we will argue that only $O(k)$ edges in P are missing from H. To see this, each edge $(a, b) \in P$ falls into one of the following cases:

1. The nodes a and b are both unclustered: this edge is present in H.
2. At least one of the nodes belongs to a clean cluster: this edge might not be present in H. We have assumed that $\rho_G(u, v)$ intersects at most k clean clusters, and therefore P intersects at most k clean clusters as well. We can verify that P intersects a clean cluster on at most four edges. Therefore, there are at most $4k$ of this type of edge in total in P.
3. At least one of the nodes belongs to a non-clean cluster, but (a, b) is not contained in any strip: this must be one of the edges immediately preceding or immediately following a departure of P from $\rho_G(u, v)$. As discussed above, there are at most four of these edges per departure. There are at most k departures, so there are at most $4k$ of this type of edge in total in P.
4. The edge (a, b) is contained in a strip: this edge is present in H.

Therefore, at most $8k$ edges in P are missing from H. For each missing edge (a, b), we know from edges added in the **multspan** subroutine that $\rho_H(a, b) = \tilde{O}(1)$. From this, we can conclude $|P| \leq \rho_G(u, v) + O(k)$, so $\rho_H(u, v) \leq \rho_G(u, v) + \tilde{O}(k)$. □

LEMMA 3. *Let* H *be a subgraph of* G *over the same vertex set, and suppose* H *contains exactly the edges of* **multspan**(G) *and* **createStrips**$(G, \mathcal{C}, d, m)$. *Let* u, v *be nodes in* G *with the property that* $\rho_G(u, v)$ *intersects exactly* k *clean clusters and fewer than* $\frac{k}{2}$ *strips. Then* $\delta_G(u, v) \geq \Omega(\frac{kd}{m})$.

PROOF. Partition $\rho_G(u,v)$ into $\frac{k}{m}$ sections, where each section contains exactly m clean clusters (the last section might contain fewer). For each of these sections, we evidently chose not to add this subpath P of $\rho_G(u,v)$ as a new strip. That means that either (1) P intersects at least m clusters, or (2) P has length at least d. The former can only be the case at most half the time, otherwise $\rho_G(u,v)$ intersects at least $\frac{k}{2}$ strips. Therefore, at least half of these sections have length at least d, so the total length of $\rho_G(u,v)$ is at least $\frac{kd}{2m}$. □

3. FIRST SPANNER CONSTRUCTION

Here we will prove the following theorem:

THEOREM 1. *For any parameter $\epsilon \in [0,1]$, there exists a spanner on $O(n^{1+\epsilon})$ edges with additive distortion $\tilde{O}(n^{1/2-\epsilon/2})$.*

Before we begin our construction, we require one new subroutine.

Very Sparse Multiplicative Spanners.

In [2], the authors describe an efficient algorithm that generates spanners on $O(n^{1+1/k})$ edges with multiplicative stretch $2k-1$. We will employ this algorithm as a subroutine several times throughout this paper; for simplicity, we will always set $k = \log n$. This gives us a spanner on $O(n)$ edges with $\tilde{O}(1)$ multiplicative stretch.

The subroutine of generating such a spanner will be called **multspan**(G).

Main Construction.

Our first spanner construction is as follows:

Algorithm 3: $\tilde{O}(n^{1/2-\epsilon/2})$ additive distortion spanners on $O(n^{1+\epsilon})$ edges

Data: An unweighted, undirected graph $G = (V, E)$ and a number ϵ

1 Initialize $H =$ **multspan**(G);
2 Initialize $\mathcal{C} =$ **cluster**(G, n^{ϵ});
3 Initialize $\mathcal{S} \leftarrow$ **createStrips**$(G, \mathcal{C}, \infty, n^{1/2-\epsilon/2})$ // $(G, \mathcal{C}, d, m)$
4 Add all edges in $\mathcal{S}$ to H;
5 Add all edges to H whose endpoints are both unclustered in $\mathcal{C}$;
6 **return** H;

Our edge bound for this construction is very straightforward. The **multspan** subroutine adds $O(n)$ edges, the **createStrips** subroutine adds $O(n)$ edges, and since every unclustered node has at most n^{ϵ} unclustered neighbors (otherwise we could have turned these into a new cluster), there are at most $n^{1+\epsilon}$ edges with both endpoints unclustered.

We will now prove our error bound.

CLAIM 2. *Any shortest path $\rho_G(u,v)$ intersects at most $n^{1/2-\epsilon/2}$ clean clusters.*

PROOF. If $\rho_G(u,v)$ intersected more than $n^{1/2-\epsilon/2}$ clean clusters, then we would add one of its subpaths as a new strip before terminating the **createStrips** subroutine. □

CLAIM 3. *The graph H returned by Algorithm 3 spans G with additive distortion of $\tilde{O}(n^{1/2-\epsilon/2})$.*

PROOF. First, note that there are at most $n^{1/2-\epsilon/2}$ strips in $\mathcal{S}$, since each strip intersects $n^{1/2-\epsilon/2}$ previously clean clusters, and there are at most $n^{1-\epsilon}$ clusters in total. The error bound is now immediate from Lemma 2. □

4. SECOND SPANNER CONSTRUCTION

We will prove the following results:

THEOREM 2. *Every graph has a spanner on $\tilde{O}(n^{1+\epsilon})$ edges with $\tilde{O}(n^{2/3-5\epsilon/3})$ additive distortion, whenever $\epsilon \in [0, \frac{1}{4}]$.*

THEOREM 3. *Every graph has an emulator on $\tilde{O}(n^{1+\epsilon})$ edges with $\tilde{O}(n^{1/3-2\epsilon/3})$ additive distortion, whenever $\epsilon \in [0, \frac{1}{5}]$.*

Hitting Sets.

We will need one last subroutine before we proceed with the construction. Let $V = \{1, \ldots, n\}$ be a set of nodes, and let $\mathcal{R}$ be a set of subsets of V. We say that $H \subset V$ is a *hitting set* of $\mathcal{R}$ if for every $R \in \mathcal{R}$, there is a node $h \in H \cap R$. The following result is well known:

LEMMA 4. *Let m be the minimum size of any element $R \in \mathcal{R}$. Then $\mathcal{R}$ has a (polynomial-time constructible) hitting set H with $|H| = O(\frac{n}{m}\log(|\mathcal{R}|))$.*

It can be shown that the greedy algorithm, which repeatedly selects the node h that hits the most un-hit sets in $\mathcal{R}$, will suffice to implement this lemma. It can also be shown that a sufficiently large random sample of nodes will implement this lemma with high probability. We will not prove this here; instead, we will simply use the notation **hittingSet**$(\mathcal{R})$ as a subroutine that constructs a hitting set of $\mathcal{R}$ of the asymptotic size bound given in the above lemma.

Main Construction.

This time, we will have two parameters in our construction: Δ and μ. For the emulator construction, we set $\mu = \frac{1}{6} - \frac{5}{6}\epsilon$ and $\Delta = \frac{1}{3} - \frac{2}{3}\epsilon$. For the spanner construction, we set $\mu = \frac{1}{3} - \frac{4}{3}\epsilon$ and $\Delta = \frac{2}{3} - \frac{5}{3}\epsilon$. Note that we have the constraint $\mu \geq 0$. This gives rise to the restrictions $\epsilon \in [0, \frac{1}{5}]$ for the emulator and $\epsilon \in [0, \frac{1}{4}]$ for the spanner.

Algorithm 4: Algorithm that implements Theorems 3 and 2

Data: A graph $G = (V, E)$ and a number ϵ

1 Initialize $H \leftarrow$ **multspan**(G);
2 Initialize $\mathcal{C} \leftarrow$ **cluster**(G, n^{ϵ});
3 Initialize $\mathcal{S} \leftarrow$ **createStrips**$(G, \mathcal{C}, n^{\Delta}, n^{\mu})$ // $(G, \mathcal{C}, d, m)$
4 Add all edges in $\mathcal{S}$ to H;
5 Add all edges between unclustered nodes to H;
6 Initialize $\mathcal{Q} \leftarrow \emptyset$;
7 **for** *every ordered pair of nodes* u, v *such that* $\rho_G(u, v)$ *intersects at least* $\frac{n^{\Delta}}{2}$ *strips or at least* n^{Δ} *clean clusters* **do**
8 Let x be the first node on $\rho_G(u, v)$ such that $\rho_G(u, x)$ intersects at least $\frac{n^{\Delta}}{2}$ strips or at least n^{Δ} clean clusters;
9 **if** $\rho_G(u, x)$ *intersects at least* $\frac{n^{\Delta}}{2}$ *strips* **then**
10 Initialize Q to be the set of all nodes within distance two of any of these strips;
11 Add Q to $\mathcal{Q}$;
12 **else**
13 Initialize Q to be the set of nodes in $\rho_G(u, x)$;
14 Add Q to $\mathcal{Q}$;
15 **end**
16 **end**
17 Initialize $\mathcal{T} \leftarrow$ **hittingSet**$(\mathcal{Q})$;

Algorithm 4: Spanner Continuation (Theorem 2)

18 Add to H all edges in **subsetspan**$(G, \mathcal{T})$;
19 **return** H;

Algorithm 4: Emulator Continuation (Theorem 3)

18 **for** *every pair* $t_1, t_2 \in \mathcal{T}$ **do**
19 Add an edge to H between t_1 and t_2 of weight $\delta_G(t_1, t_2)$;
20 **end**
21 **return** H;

The main **for** loop in this algorithm omits pairs u, v with the property that $\rho_G(u, v)$ intersects less than $\frac{n^{\Delta}}{2}$ strips and less than n^{Δ} clean clusters. We can ignore these paths because we already have enough edges in place to span these paths with $\tilde{O}(n^{\Delta})$ (this is immediate from Lemma 2).

We will now prove our certificates for this construction.

Claim 4 (Edge Bound). *In either case, the returned graph H has $\tilde{O}(n^{1+\epsilon})$ edges.*

Proof. Our first goal is to place a lower bound on the minimum size of any element $Q \in \mathcal{Q}$. We create these elements Q in lines 10 and13. In the first case, we are adding the nodes within distance two of $\Omega(n^{\Delta})$ different strips. Each of these strips was the first to intersect n^{μ} distinct clusters, each of which contains at least n^{ϵ} nodes. Therefore, the size of each of these elements is $\Omega(n^{\Delta+\mu+\epsilon})$. In line 13, we add all of $\delta_G(u, x)$ to Q. We know that this path intersects at least n^{Δ} clean clusters and at most $\frac{n^{\Delta}}{2}$ strips. By Lemma 3, we conclude that this path has length at least $\Omega(n^{2\Delta-\mu})$.

By Lemma 4, the size of $\mathcal{T}$ is at most $\tilde{O}(n^{\max(1-\Delta-\mu-\epsilon, 1-2\Delta+\mu)})$. For the emulator parameter settings, this gives $|\mathcal{T}| = \tilde{O}(n^{1/2+\epsilon/2})$, so we add $|\mathcal{T}|^2 = \tilde{O}(n^{1+\epsilon})$ edges to H in the emulator continuation. For the spanner parameter settings, this gives $|\mathcal{T}| = \tilde{O}(n^{2\epsilon})$. Then **subspan**$(G, \mathcal{T})$ has $n|\mathcal{T}|^{1/2} = \tilde{O}(n^{1+\epsilon})$ edges. □

Claim 5 (Error Bound). *The returned graph H spans/emulates G with $\tilde{O}(n^{\Delta})$ additive distortion.*

Proof. Let u, v be arbitrary nodes in V. First, suppose that $\rho_G(u, v)$ intersects fewer than $\frac{n^{\Delta}}{2}$ strips and fewer than n^{Δ} clean clusters. Then by Lemma 2, we already have $\rho_H(u, v) \leq \rho_G(u, v) + \tilde{O}(n^{\Delta})$.

Otherwise, let x_u be the first node on $\rho_G(u, v)$ such that $\rho_G(u, x_u)$ does *not* have this property, and let x_v be the same for $\rho_G(v, u)$. First, suppose $\rho_G(u, x_u)$ intersects at least $\frac{n^{\Delta}}{2}$ strips. Then there is some node $t_u \in \mathcal{T}$ on one of these strips (Line 10). Otherwise, there is some node $t_u \in \mathcal{T}$ that sits directly on the path $\rho_G(u, x_u)$ (Line 13). Let w_u be the closest node on $\rho_G(u, x_u)$ to t_u. Define t_v, w_v similarly with respect to v.

We now proceed by the triangle inequality. We have

$$\delta_G(t_u, t_v) \leq \delta_G(t_u, w_u) + \delta_G(w_u, w_v) + \delta_G(w_v, t_v)$$

We know that $\delta_G(t_u, w_u)$ and $\delta_G(w_v, t_v)$ are both $O(n^{\Delta})$, since t_u (t_v) is within distance two of a strip that intersects $\rho_G(u, x_u)$ ($\rho_G(x_v, v)$). We can then write

$$\delta_G(w_u, t_u) + \delta_G(t_u, t_v) + \delta_G(t_v, w_v) \leq \delta_G(w_u, w_v) + O(n^{\Delta})$$

Some algebra yields

$$\begin{aligned}\delta_G(u, w_u) + \delta_G(w_u, t_u) + \delta_G(t_u, t_v) + \delta_G(t_v, w_v) + \delta_G(w_v, v) \\ \leq \delta_G(u, w_u) + \delta_G(w_u, w_v) + \delta_G(w_v, v) + \tilde{O}(n^{\Delta})\end{aligned}$$

Since $w_u, w_v \in \rho_G(u, v)$, the right-hand side simplifies.

$$\begin{aligned}\delta_G(u, w_u) + \delta_G(w_u, t_u) + \delta_G(t_u, t_v) + \delta_G(t_v, w_v) + \delta_G(w_v, v) \\ \leq \delta_G(u, v) + \tilde{O}(n^{\Delta})\end{aligned}$$

It follows from Lemma 2 that $\delta_H(u, w_u) \leq \delta_G(u, w_u) + \tilde{O}(n^{\Delta})$ and $\delta_H(w_v, v) \leq \delta_G(w_v, v) + \tilde{O}(n^{\Delta})$. We also know that $\delta_G(w_u, t_u) \leq n^{\Delta}$ and $\delta_G(t_v, w_v) \leq n^{\Delta}$. Additionally, we know that $\delta_H(t_u, t_v) \leq \delta_G(t_u, t_v) + 2$, because we have added a subset spanner (or a direct emulator edge) that enforces this property on every pair of nodes in $\mathcal{T}$. This gives us

$$\begin{aligned}\delta_H(u, w_u) + \delta_H(w_u, t_u) + \delta_H(t_u, t_v) + \delta_H(t_v, w_v) + \delta_H(w_v, v) \\ \leq \delta_G(u, v) + \tilde{O}(n^{\Delta})\end{aligned}$$

We finish by applying the triangle inequality to the left-hand side.

$$\delta_H(u, v) \leq \delta_G(u, v) + \tilde{O}(n^{\Delta})$$

□

It is worth noting that better spanners would follow quickly from better subset spanners. It is not yet known how to generalize the subset spanner result in [12] to fewer edges and a

non-constant error bound; for our purposes, any error of up to n^{Δ} would be acceptable. This seems to be an attainable and relevant problem, which we leave open.

5. CONCLUSION

We have improved the additive distortion bounds for spanners and emulators for many values of ϵ below the $n^{4/3}$ edge threshold. The main open question of spanner research still remains: do there exist spanners or emulators on $O(n^{4/3-\delta})$ edges with additive distortion $n^{o(1)}$ for some constant $\delta > 0$?

Our work also suggests that further improvements in spanner construction may follow from generalizations of the recent work in subset spanners. More formally: can we construct a subset spanner on an asymptotically smaller number of edges, given a subset size of $|S|$ and a budget of n^{Δ} additive distortion?

6. REFERENCES

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28:1167–1181, 1999.

[2] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[3] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM*, pages 32, 804–823, 1985.

[4] S. Baswana and T. Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. *Proc. 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 591–602, 2006.

[5] S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. Additive spanners and (α, β)-spanners. *ACM Trans. Algo.*, 7:A.5, 2005.

[6] S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. New constructions of (α, β)-spanners and purely additive spanners. *Proc. 16th SODA*, pages 672–681, 2005.

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

[8] B. Bollobás, D. Coppersmith, and M. Elkin. Sparse distance preservers and additive spanners. *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 414–423, 2003.

[9] Shiri Chechik. New additive spanners. *SODA*, pages 498–512, 2013.

[10] L. J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, pages 28, 170–183, 2001.

[11] L. J. Cowen and C. G. Wagner. Compact roundtrip routing in directed networks. *Journal of Algorithms*, pages 50, 1, 79–95, 2004.

[12] M. Cygan, F. Grandoni, and T. Kavitha. On pairwise spanners. *Symposium on Theoretical Aspects of Computer Science (STACS)*, 2013.

[13] Dorit Dor, Shay Halperin, and Uri Zwick. All pairs almost shortest paths. *Proc. 37th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 452–461, 1996.

[14] M. Elkin. Computing almost shortest paths. *ACM Trans. Algorithms*, pages 1(2):283–323, 2005.

[15] M. Elkin. A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners. *Proc. 26th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 185–194, 2007.

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

[17] P. Erdös. Extremal problems in graph theory. *Theory of graphs and its applications*, pages 29–36, 1964.

[18] A. M. Farley, A. Proskurowski, D. Zappala, and K. Windisch. Spanners and message distribution in networks. *Discrete Applied Mathematics*, pages 137(2):159–171, 2004.

[19] Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. *Symposium and Workshop on Algorithm Theory (SWAT)*, pages 277–281, 2014.

[20] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal of Computing*, pages 18, 740–747, 1989.

[21] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, pages 36(3):510–530, 1989.

[22] Seth Pettie. Low distortion spanners. *34th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 78–89, 2007.

[23] L. Roditty, M. Thorup, and U. Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Trans. Algorithms*, page 3(4): Article 29, 2008.

[24] M. Thorup and U. Zwick. Compact routing schemes. *Proc. 13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

[25] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM 52*, pages 1, 1–24, 2005.

[26] D. P. Woodruff. Lower bounds for additive spanners, emulators, and more. *Proc. 47th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 389–398, 2006.

[27] D. P. Woodruff. Additive spanners in nearly quadratic time. *37th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 463–474, 2010.