

New Methods for Approximating Shortest Paths

by

Kevin Lu

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
May 24, 2019

Certified by.....
Virginia V. Williams
Associate Professor
Thesis Supervisor

Accepted by
Katrina LaCurts
Chair, Department Committee on Graduate Theses

New Methods for Approximating Shortest Paths

by

Kevin Lu

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

A spanner H of a graph G is a sparse subgraph that approximates all pairwise distances. Of particular interest are additive spanners on unweighted graphs, which satisfy the following for any two vertices u, v : $\text{dist}(H, u, v) \leq \text{dist}(G, u, v) + f(n)$, where dist is the distance with respect to the graph H or G and $f(n)$ is a function of the number of vertices of the graph. We study a variety of problems related to additive spanners, and have two new results of significance.

For additive spanners with $O(n)$ edges, it is well known that $f(n)$ must be a polynomial function $O(n^\alpha)$ for some $0 < \alpha < 1$. Previously, it was known that the optimal value of α was between $1/13$ and $3/7$; by combining two previously known methods, our first significant result improves the lower bound from $1/13$ to $1/11$.

The all pairs approximate shortest paths problem takes as input an unweighted graph, and outputs a distance matrix that approximates all pairwise distances. We present a new improvement to the algorithm of Dor, Halperin, and Zwick for the $+4$ and $+6$ approximation algorithms. In the $+4$ approximation algorithm, our new algorithm runs in $O(n^{15/7})$ time, an improvement from the previous $O(n^{11/5})$, and in the $+6$ approximation algorithm, our new algorithm runs in $O(n^{19/9})$ time, an improvement from the previous $O(n^{17/8})$.

Thesis Supervisor: Virginia V. Williams
Title: Associate Professor

Acknowledgments

I would like to thank professor Virginia V. Williams, for being an incredible thesis supervisor. Her support and many insights helped make this thesis possible. I would also like to thank Greg Bodwin for his help and expertise, especially for some of the ideas in Chapter 2.

Finally, I would like to give a special thank you to my friends and family for helping me through this journey.

Contents

1	Introduction	13
1.1	Preliminaries and Notation	13
1.2	Related Work	14
1.3	Our Contributions and Outline	17
2	Sparse spanner lower bounds	19
2.1	Combining two ideas	19
2.1.1	The base graph	20
2.1.2	A graph product	22
2.1.3	A recursive construction	23
2.1.4	Analysis of size and bounds	25
2.2	Future Directions	28
2.2.1	Avoiding Layered Structure	28
2.2.2	k-Average free sets	30
2.2.3	Other approaches	31
3	Efficient Algorithms	33
3.1	Matrix Multiplication	33
3.2	All Pairs Approximate Shortest Paths	35
3.2.1	Preliminaries	36
3.2.2	Faster algorithms for +4 and +6 stretch	37
3.2.3	Further attempts to improve APASP	41

4	Conclusions and Future Work	43
4.1	Knudsen's clustering method and algorithm performance	44
4.2	Unexplored questions	44

List of Figures

- 2-1 A diagram taken from [2] showing the structure of the recurrence. Each L_i represents one layer in G_0 . Any path in G_0 will have edges of alternating colors, which we call a and b . In H_k , the corresponding path travels between the $\pi(a)$ th and $\pi(b)$ th input/output in H_{k-1} . The paths between consecutive input/output ports has length $(2l - 1)^{k-1}$. 24

List of Tables

1.1	A variety of algorithms for spanner construction in the centralized computation model.	17
2.1	Known bounds and constructions for a variety of spanners. Entries above the doubled lines are upper bounds, while entries below them are lower bounds	20

Chapter 1

Introduction

All known solutions to the all pairs shortest paths (APSP) problem require $\tilde{O}(n^3)$ time, a runtime that has not been improved on since Dijkstra's algorithm was created. In this thesis, we consider ways to approximate the APSP problem. In particular, we focus on additive spanners in unweighted graphs.

Consider an unweighted graph G . A subgraph H is an (α, β) spanner of G if for every pair of vertices $(s, t) \in V$, we have $\text{dist}(s, t, H) \leq \alpha \cdot \text{dist}(s, t, G) + \beta$. Spanners with $\alpha = 1$ are additive spanners, and spanners with $\beta = 0$ are multiplicative spanners. The study of spanners generally analyzes the trade-offs between the size of $|H|$ and the values of α and β .

There are a few specific problems in spanners that we have focused on. First, we will consider the sparsest possible spanners - those of size $O(n)$. Next, we consider the problem of fault-tolerant spanners, which maintain the distance guarantees even with the possibility of a fixed number of edge or vertex failures. Finally, we study efficient algorithms for creating spanners.

1.1 Preliminaries and Notation

In a graph G , let V_G to be the set of vertices and E_G the set of edges, with sizes $|V_G|$ and $|E_G|$ respectively. Let $\text{dist}(u, v, G)$ to be the shortest-path distance between vertices u and v in G ; we will write just $\text{dist}(u, v)$ when it is clear from the context.

Thus formally, $H \subseteq G$ an (α, β) spanner of G if

$$\text{dist}(u, v, H) \leq \alpha \cdot \text{dist}(u, v, G) + \beta$$

for all $u, v \in V$. For a spanner H , we call $|E_H|$ the *size* of the spanner. Furthermore, for $(1, \beta)$ spanners, we will call β the *stretch* or *additive stretch*.

Fault tolerant spanners consider graphs where up to f components may fail. In the vertex fault tolerant (VFT) model, a subgraph $H \subseteq G$ is an f -vertex fault tolerant spanner if

$$\text{dist}(u, v, HF) \leq \alpha \cdot \text{dist}(u, v, GF) + \beta$$

for all $u, v \in V$, where $F \subset V$ and $|F| \leq f$.

The *diameter* of a graph G is the maximum value of $\text{dist}(u, v)$ over all pairs $u, v \in V$.

Given an arbitrary weighted graph G , we define (β, ϵ) *hopset* H as follows. H is a set of edges (u, v) with the edge weight equal to $\text{dist}(u, v, G)$. The graph G' is the weighted graph $(V, E \cup H, w)$, the β weighted distance $\text{dist}_{G'}^{(\beta)}(u, v)$ is the length of the shortest path from u to v in G' that uses at most β edges. H is a (β, ϵ) hopset, with $\beta \geq 1, \epsilon > 0$, if for any $u, v \in V$, we have

$$\text{dist}_{G'}^{(\beta)}(u, v) \leq (1 + \epsilon)\text{dist}_G(u, v)$$

1.2 Related Work

The study of graph spanners first arose from the study of network synchronization [5]. Most early work focused on multiplicative spanners. One of the first significant theoretical results was given by Althofer et. al in 1993 [4], which showed, using a simple greedy algorithm, that any graph has a $2k + 1$ multiplicative spanner of size $O(n^{1+1/k})$. Furthermore, under Erdős' girth conjecture, this bound is tight.

More recent work has generally been focused on either additive or mixed spanners.

There are several different constructions for $(1, \alpha)$ spanners where $\alpha = O(1)$, with some of the most notable being a $+2$ spanner with $O(n^{3/2})$ size [3], a $+4$ spanner with $O(n^{7/5})$ size [14], and a $+6$ spanner with $O(n^{4/3})$ size [7]. Recently, some papers also provided alternate constructions for the $+2$ and $+6$ versions [36, 24]. Woodruff [35] proved that, regardless of the truth of Erdős’ girth conjecture, there exists a graph on n nodes such that any $(2k - 1)$ spanner of the graph has at least $\Omega(k^{-1}n^{1+1/k})$ nodes. Of the additive spanners with $O(1)$ stretch, these results show that sizes of the $+2$ and $+6$ spanners are optimal with respect to the value of the polynomial exponent, but there is still a gap for the size of the $+4$ spanner. In a notable recent paper, Abboud and Bodwin proved that the $O(n^{4/3})$ size is optimal for all additive spanners with $O(1)$ stretch [1].

While the $4/3$ stretch factor is tight for $O(1)$ additive stretch, there are a variety of known spanner constructions that have fewer edges, with the trade-off of a looser approximation function. Elkin and Peleg [21] showed that fewer than $O(n^{4/3})$ edges is possible for $(1 + \epsilon, \beta)$ spanners, with $\epsilon > 0$; their result was later improved by Thorup and Zwick [33]. Thorup’s results are one of the first constructions of a family of spanners, with size and stretch trade offs.

One of Thorup’s results is a family *sublinear additive spanner*, an additive spanner with sublinear polynomial stretch and fewer than $\tilde{O}(n^{4/3})$ edges. Of particular interest to us are *sparse sublinear additive spanners*, which have just $O(n)$ edges. Recent work has improved the stretch factor for these sparse spanners [2, 23]. For a more comprehensive list of many currently known bounds, please refer to Table 2 in the chapter on spanner lower bounds.

In 2010, Chechik et. al [15] introduced the idea of fault-tolerant spanners, and presented algorithms for constructing a f -fault tolerant $(2k - 1)$ stretch multiplicative spanner with $O(f^2 k^{f+1} n^{1+1/k} \log^{1-1/k} n)$ edges. This result was improved by Dinitz and Krauthgamer [17]. Brauchschvig et. al [13] was the first to demonstrate the existence of additive fault-tolerant spanners, and their result has been improved by a variety of authors since. A few examples are [28] [8] [26]. Still relatively little is known about them; many recent results only consider the case of a small number of

faults, and it seems the only known lower bounds are the trivial ones.

There are also a variety of problems closely related to spanners. For example, pairwise distance preservers are sparse subgraphs that exactly preserve distances for certain pairs of vertices. The idea was first introduced by Bollobás et. al [12], and have seen a decent amount of study [11, 29]. Sparse pairwise distance preservers are used in the construction of many state-of-the-art spanner constructions, such as the ones by Bodwin and Williams [10, 9]. Furthermore, fault tolerant distance preservers and BFS trees, such as Parter and Peleg’s construction [27], are also essential to the construction of certain fault tolerant spanners.

An emulator is similar to a spanner in that it approximates pairwise distances. However, emulators are not restricted to being a subgraph of the original graph; this generally means that emulators can be sparser than spanners, for the same stretch. Emulators were introduced first by Dor, Halperin, and Zwick [18]. Most constructions of graphs proving spanner lower bounds have natural corresponding emulators, including the most recent results [7, 2, 23].

There are two other problems of note that tend to have a corresponding solution for each spanner *lower bound*. The problem of shortcutting digraphs, first introduced by Thorup [32], considers the diameter of a graph G . It asks, given a set of “shortcutting” edges E' , is it possible to reduce the diameter of the new graph, $G' = (V, E \cup E')$, with the minimum possible size of E' . The problem is particularly interesting when $|E'| = n$ or $|E'| = m$. The current best known results come from the graph construction of Huang and Pettie [23], which is very similar to their construction to bound spanners.

The second notable problem is that of hopsets, which were first explicitly studied by Cohen [16]. The current best known results are due to Abboud, Bodwin, and Pettie [2], but it is likely that the construction of Huang and Pettie [23] can also be modified for hopsets. Elkin and Neiman [20] showed that any graph has a (β, ϵ) hopset of size $\tilde{O}(n^{1+1/k})$, where $\beta = (\log k/\epsilon)^{\log k}$.

Furthermore, another area of research involves the construction of efficient algorithms for additive spanners. In the centralized computation model, we consider “efficient” algorithms to be algorithms that run in $\tilde{O}(n^2)$ time. Currently, there are

Table 1.1: A variety of algorithms for spanner construction in the centralized computation model.

Citation	Spanner Size	Additive Stretch	Algorithm Runtime
Knudsen [25]	$O(n^{3/2})$	2	$O(n^2)$
Woodruff [36]	$O(n^{4/3})$	6	$O(n^2 \log^2 n)$
Baswana et al [7]	$O(n^{4/3})$	6	$O(n^{2/3}m)$
Knudsen [25]	$O(n^{4/3})$	8	$O(n^2)$

known efficient algorithms for the +2, +6, and +8 additive spanners. See table 1.2 for a summary of the current best known results.

Dor, Halperin, and Zwick [18] introduced the All Pairs Almost Shortest Paths (APASP) problem, and provide a family of algorithms for calculating it. APASP approximates pairwise distances, but these pairwise distances do not necessarily correspond to any particular subgraph. For +2 stretch the algorithm runs in time $\tilde{O}(\min(n^{3/2}m^{1/2}, n^{7/3}))$, and for all even $k > 2$, the + k stretch algorithm runs in time $\tilde{O}(\min(n^{2-2/(k+2)}m^{2/(k+2)}, n^{2+2/(3k+2)}))$. To the best of our knowledge, there have not been any significant improvements to their method since.

1.3 Our Contributions and Outline

We have two main results, and a variety of interesting but unsuccessful attempts for some open problems. We have divided the remainder of this thesis into four chapters. The first chapter discusses our work with spanner lower bounds, and in particular, sparse additive spanner lower bounds. The second chapter involves a brief study of fault tolerant spanners. The third chapter deals with creating faster algorithms for certain spanner constructions and the All Pairs Almost Shortest Paths problem. Finally, in the fourth chapter we summarize our work and suggest a few promising directions for future continued research.

In our first main result, we improve the lower bound for sparse additive sublinear spanners. We show that,

Theorem 1.3.1. *There exists a family of graphs G such that any additive spanner with $O(n)$ edges requires $O(n^{1/11})$ stretch.*

This is an improvement over the previous lower bound of $O(n^{1/13})$. In our second main result, we improve the runtime for the +4 and +6 versions of the APASP problem. In particular we show that,

Theorem 1.3.2. *Given an unweighted graph G , there exists an algorithm that approximates all pairwise distances to within an additive stretch of 4 in $O(n^{15/7})$ time.*

Theorem 1.3.3. *Given an unweighted graph G , there exists an algorithm that approximates all pairwise distances to within an additive stretch of 6 in $O(n^{19/9})$ time.*

These are improvements over the previous runtimes of $O(n^{11/5})$ and $O(n^{15/7})$, respectively.

Chapter 2

Sparse spanner lower bounds

We define *sparse spanners* as spanners that have $O(n)$ edges. Additive sparse spanners cannot have constant distortion, but can have a polynomial $O(n^\alpha)$ distortion. A natural and major open question asks what is the optimal possible value of α such that every graph has a sparse spanner with $O(n^\alpha)$ distortion. It has been conjectured by Bodwin and Williams [9] that the optimal value for α is $3/7$, based on a variety of current known upper bounds.

In this section we discuss approaches that we have tried to improve the best current known *lower bounds*. All interesting known lower bounds have taken the approach of constructing a graph with a large number of "critical edges." Critical edges of a graph must be included in a spanner to satisfy approximate distances. We will also use this approach.

The current best known upper and lower bounds are outlined in table 2:

2.1 Combining two ideas

Our main result combines two major ideas from [23] and [2]. Here, we provide a construction of a family of graphs that will require at least $O(n^{1/11-\epsilon})$ distortion for any sparse spanner, for any $\epsilon > 0$. This improves upon the previous bound of $O(n^{1/13})$.

Table 2.1: Known bounds and constructions for a variety of spanners. Entries above the doubled lines are upper bounds, while entries below them are lower bounds

Citation	Spanner Size	Additive Stretch	Notes
Aingworth, Chekuri, Indyk, and Mowani [3]	$\tilde{O}(n^{3/2})$	2	
Checknik [14]	$\tilde{O}(n^{7/5})$	4	
Baswana, Kavitha, Mehlhorn, and Pettie [7]	$\tilde{O}(n^{4/3})$	6	
Checknik [14]	$O(n^{20/17+\epsilon})$	$O(n^{4/17-3\epsilon/2})$	$0 \leq \epsilon$
Bowdin and Williams [9]	$\tilde{O}(n^{5/4-5\epsilon/12})$	$O(n^\epsilon)$	$3/13 \leq \epsilon$
Bowdin and Williams [9]	$\tilde{O}(n^{4/3-7\epsilon/9})$	$O(n^\epsilon)$	$0 \leq \epsilon \leq 3/13$
Bodwin and Williams [9] (Conjectured)	$O(n^{4/3-7\epsilon/9})$	$O(n^\epsilon)$	$0 \leq \epsilon$
Abboud and Bodwin [2]	$O(n^{4/3-\epsilon})$	$\Omega(n^\delta)$	$\delta = \delta(\epsilon)$
Abboud and Bodwin [2]	$O(n)$	$\Omega(n^{1/22})$	
Huang and Pettie [23]	$O(n)$	$\Omega(n^{1/13})$	
new	$O(n)$	$\Omega(n^{1/11-\epsilon})$	$0 \leq \epsilon$

2.1.1 The base graph

Consider the points on the integer lattice \mathbf{Z}^d , where d is some positive integer dimension. Define $B_d(\rho)$ to be the set of all lattice points within Euclidean distance ρ of the origin, and define $V_d(\rho)$ to be the convex hull of $B_d(\rho)$. By a result of Barany and Larman [6], $V_d(\rho)$ contains $\Theta\left(\rho^{d\frac{d-1}{d+1}}\right)$ points. This result is used as a building block of graphs in [23], and will also be used in our construction. For convenience, we will write $\delta_\rho(d) = \Theta\left(\rho^{d\frac{d-1}{d+1}}\right)$, and drop the subscript when it is clear what ρ is from the context. We will now create the base graph, G_b .

Vertex set: the base graph G_b will consist of $l + 1$ total layers, starting with layer 0 and ending with layer l , with each layer having n^d vertices representing all of the unique elements of $(\mathbb{Z}/n\mathbb{Z})^d$. We label the vertex representing the d -tuple x in layer i as v_x^i .

Edge set: We only create an edge between two vertices if they are in adjacent layers, and the points they represent are separated by an element of $V_d(\rho)$. We will set $\rho \ll n$ here. To be precise, there is an edge between v_x^i and v_y^{i+1} if and only if $\exists \gamma \in V_d(\rho) : x + \gamma = y$.

Path set: We will keep track of a set of paths P , which will all be of length l . For an arbitrary $\gamma \in V_d(\rho)$ and $s \in (\mathbb{Z}/n\mathbb{Z})^d$, we have paths consisting of the vertices, $v_s^0, v_{s+\gamma}^1, v_{s+2\gamma}^2, \dots, v_{s+l\gamma}^l$.

We will set $l\rho = n$. We can quickly calculate the sizes of the vertex, edge, and path sets as follows:

$$\begin{aligned} |V| &= (l\rho)^d l \\ |E| &= (l\rho)^d l \delta(d) \\ |P| &= (l\rho)^d \delta(d) \end{aligned}$$

Next, we will finish our discussion on the base graph with the following lemma:

Lemma 2.1.1. *Consider a path $p \in P$, with starting vertex s in layer 0 and ending vertex t in layer l . The path p is also the unique shortest path between s and t .*

Proof: The path p has length l because it contains one vertex in each layer. It is clear that any path between s and t must have length at least l , as edges only exist between adjacent layers. We now must show that p is unique. WLOG we set p to be the path containing the vertices, $s = v_x^0, v_{x+\gamma}^1, v_{x+2\gamma}^2, \dots, v_{x+l\gamma}^l = t$ for some $\gamma \in V_d(\rho)$.

For the sake of contradiction, assume there is another path p' . Then, since edges only exist between vertices of two layers if their corresponding points are separated by some $\gamma' \in V_d(\rho)$, we can define the path by the vertices $s = v_x^0, v_{x+\gamma_1}^1, v_{x+\gamma_1+\gamma_2}^2, \dots, v_{x+\sum_i \gamma_i}^l = t$ for some value of $\gamma_1, \gamma_2, \dots, \gamma_l \in V_d\rho$. This implies that,

$$l\gamma \equiv \sum_{i=1}^l \gamma_i \pmod{n}$$

However, note that all terms are in the convex hull $V_d(\rho)$, and that we have chosen $n = l\rho$. This value is sufficiently large that we must have,

$$l\gamma = \sum_{i=1}^l \gamma_i$$

The only solution for this is when $\gamma_1 = \gamma_2 = \dots \gamma_l = \gamma$, which means p and p' correspond to the same path. This finishes our proof of the lemma ■.

As a result of this, we also get the following corollaries.

Corollary 2.1.2. *The edge set of G_b is exactly equal to the union of the paths P .*

Corollary 2.1.3. *Each edge $(u, v) \in G_b$ is part of exactly one path $p \in P$*

Corollary 2.1.4. *Two points $u, v \in V$ can both be in at most one path $p \in P$.*

2.1.2 A graph product

We will create a denser and more complex graph starting from G_b , which we will call G_0 . It has edge and vertex sets defined below:

Vertex set: We label all vertices in the form $v_{x,y}^i$, where $x, y \in (\mathbb{Z}/n\mathbb{Z})^d$ and $i \in \{0, 2, \dots, 2l + 1\}$.

Edge set: For even i , we create an edge between $v_{x,y}^i$ and $v_{x',y}^{i+1}$ if in the base graph G_b , there exists an edge between $v_x^{i/2}$ and $v_{x'}^{i/2+1}$. Similarly, for odd i , we create an edge between $v_{x,y}^i$ and $v_{x,y'}^{i+1}$ if there exists an edge between $v_y^{(i-1)/2}$ and $v_{y'}^{(i+1)/2}$ in G_b .

Paths: For any two (not necessarily distinct) paths $v_{x_0}^0, v_{x_1}^1, \dots, v_{x_l}^l$ and $v_{y_0}^0, v_{y_1}^1, \dots, v_{y_l}^l$ in the base graph G_b , we add the path $v_{x_0, y_0}^0, v_{x_1, y_0}^1, v_{x_1, y_1}^2, \dots, v_{x_l, y_{l-1}}^{2l}, v_{x_l, y_l}^{2l+1}$ to the paths set of G_p .

Let us consider the size of the vertex, edge, and paths set. The number of vertices in each layer has been squared, since they correspond to pairs of points now, but the number of layers has stayed within a constant factor. Furthermore, each vertex has degree equal to $2\delta(d)$ if it is not in layers 0 or $2l + 1$, and has degree $\delta(d)$ otherwise. Finally, each path in G_p corresponds to a pair of paths in G_0 , so the size of the path set has been squared. So for this graph we have,

$$\begin{aligned} |V| &= O((l\rho)^{2d}l) \\ |E| &= O((l\rho)^{2d}l\delta(d)) \\ |P| &= (l\rho)^{2d}\delta(d)^2 \end{aligned}$$

We make one final note before starting a recursive construction.

Claim 2.1.5. *Each pair of paths in G_p can share at most one edge. Furthermore, we can label every edge with one of $(l\rho)^d\delta(d)$ colors such that the edges in each path alternate colors.*

Proof: Assume two paths share more than one edge. More than one shared edge implies that the two pairs of paths share at least 3 distinct points, which means that in each of base graphs G_b , there are two common corresponding points. From corollary 2.1.4, this means that the paths in the base graphs were identical. This contradicts means that the two paths in G_0 must be the same.

For the coloring, we can assign one color to each path in the base graph G_b . Then, since each edge in G_0 corresponds to an edge in a copy of G_b , let the color correspond as well. It is easy to verify this coloring forces every path $p \in P$ to alternate colors.

2.1.3 A recursive construction

Thus far we have generally been following the construction of Huang and Pettie [23], with the primary difference being that all layers contain exactly the same number of vertices. This allows our graph to be used in the recursive approach of Abboud, Bodwin, and Pettie [2]. We will set up some notation first.

We will recursively create successive graphs H_1, H_2, \dots , each with slightly improved guarantees for sparse spanner size. Each of graph H_i will have a path set P_i and edge set E_i . We will let $H_i[x]$ denote the graph as a function of the size of the first layer, and use similar notation $E_i[x]$ and $P_i[x]$. Each graph H_i will be layered; vertices in the first layer are called **input ports** and vertices in the last layer are called **output ports**. Further, we let $\bar{H}_i[x]$ denote the graph with the same structure as $H_i[x]$, but with input and output ports reversed. We describe the process below.

The base case. In the base case, we have the complete bipartite graph on $2x$. Precisely, we can label the vertices 1 through $2x$. Then, both the edge set and path set are the same: $E_1[x] = P_1[x] = \{1, 2, \dots, x\} \times \{x + 1, x + 2, \dots, 2x\}$. Thus we have $|E_1[x]| = |P_1[x]| = x^2$.

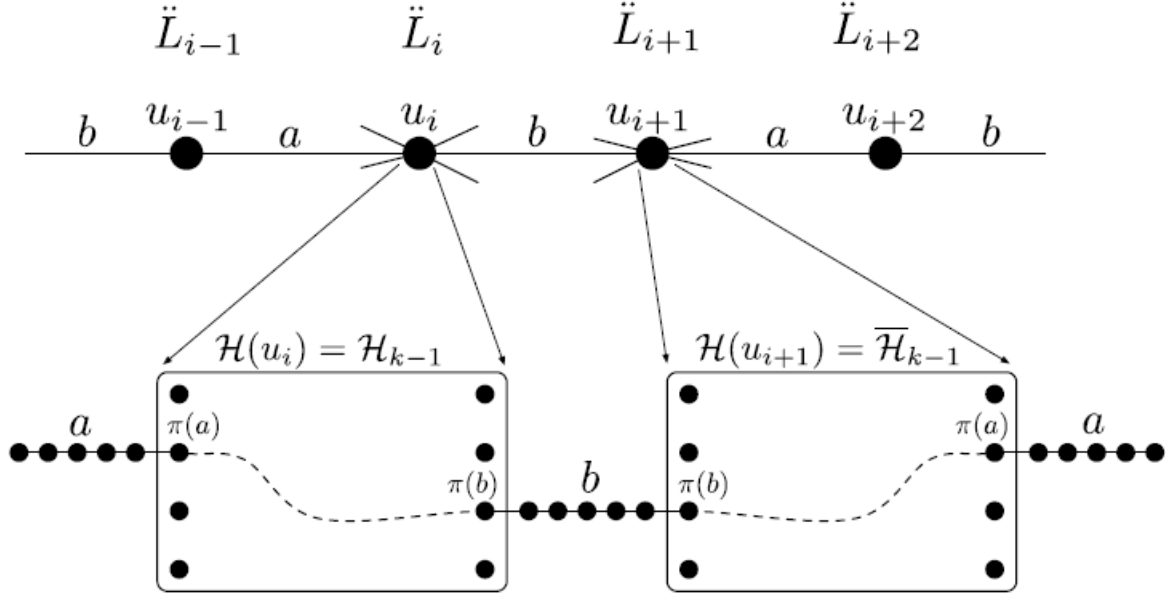


Figure 2-1: A diagram taken from [2] showing the structure of the recurrence. Each L_i represents one layer in G_0 . Any path in G_0 will have edges of alternating colors, which we call a and b . In H_k , the corresponding path travels between the $\pi(a)$ th and $\pi(b)$ th input/output in H_{k-1} . The paths between consecutive input/output ports has length $(2l - 1)^{k-1}$.

The recursive case. When $k \geq 1$ we construct $H_k[x]$ using the two graphs $H_{k-1}[\delta(d)]$ and $G_0[x]$ as follows.

Let $\pi : \{1, 2, \dots, \delta(d)\} \rightarrow \{1, 2, \dots, \delta(d)\}$ be a randomly selected *port assignment permutation*. $G_0[x]$ has $2l + 1$ layers; we will leave the first and last layers as is. For each remaining vertex $v_{x,y}^i$, we replace it with $H_{k-1}[\delta(d)]$ if i is even and $\bar{H}_{k-1}[\delta(d)]$ if i is odd. For each edge from layer i to $i + 1$ with label a , replace it with a path of length $(2l - 1)^{k-1}$ from the $\pi(a)$ th output port of the i th layer to the $\pi(a)$ th input port of the $i + 1$ th layer. See figure 2-1 for a diagram.

The new path set $P_k[x]$ is defined in the natural way. For each $p \in P$, where P is the path set for G_0 , p alternates colors. Call the colors a and b . Then, the path set $P_k[x]$ contains all paths such that the path from $\pi(a)$ to $\pi(b)$ is contained in P_{k-1} . We also have the following lemma:

Lemma 2.1.6. *If $(u_0, u_{2l}) \in P_k$ then there is a unique shortest path from u_0 to u_{2l} in H_k . This path passes through exactly $(2l - 1)^{k-1}$ complete bipartite graphs and has*

length $(2(k-1)l+1)(2l-1)^{k-1}$.

Proof: we will prove this recursively. Let d_k be the length of the paths in P_k . When $k=1$, we have 1 complete bipartite graph, as required; also, we have $d_1=1$, so the base case is satisfied. For d_k , consider the path path in H_k and the corresponding path in G_0 . Note that the path in G_0 passes through $2l+1$ vertices and $2l$ edges. In H_k , each vertex becomes a path through a copy of H_{k-1} , and each edge becomes a path of length $(2l-1)^{k-1}$. Therefore we have the recurrence:

$$\begin{aligned} d_1 &= 1 \\ d_k &= (2l-1)d_{k-1} + 2l(2l-1)^{k-1} \end{aligned}$$

This has the closed form $d_k = (2(k-1)l+1)(2l-1)^{k-1}$ as desired.

Next, we shall proceed to analyze what lower bounds we can construct from this new graph.

2.1.4 Analysis of size and bounds

We will start with an analysis of the graph H_2 . Then we will analyze the recursive graphs H_2, H_3, \dots , each of which will have a slightly better spanner bound. We start with the following observation:

Claim 2.1.7. *Every edge in H_k that is part of some copy of H_1 is part of at most 1 path $p \in P_k$.*

Proof: We proceed by induction. The base case for H_1 is clearly true. For H_k , the corresponding clique edge must be inside some copy of H_{k-1} . Inductively, there's at most one path $p' \in P_{k-1}$ that contains that edge, and suppose that a and b are the input and output ports.

Given the input and output ports, by contracting all copies of H_{k-1} , we get a corresponding edge in the graph G_0 . By lemma 2.1.1, this edge is part of exactly one path in G_0 . This means the original edge can be part of at most one path in H_k .

We will also make use of the following lemma, which is a generalization of lemmas used by previous works:

Lemma 2.1.8. *Let $D = (2l - 1)^{k-1}/2$. Every spanner of H_k with additive stretch $+2D$ must contain at least $D|P_k|$ (bipartite) clique edges.*

Proof: For the sake of contradiction suppose there exists a spanner H with stretch $2D - 1$ containing at most $D|P_k| - 1$ clique edges. Each path $p \in P_k$ traverses through $(2l - 1)^{k-1} = 2D$ clique edges. Thus, by the pigeonhole principal, there exists a path $p \in P_k$ such that at least D clique edges are missing in p .

Let the new shortest path in H be p' . On both graphs H_k and H , we contract all copies of H_{k-1} , and replace all the subdivided paths with single edges. Essentially we are doing the reverse of the matrix product, and we return to the graph G_0 . Let the contracted path be p'' ; there are then two cases:

1. If p'' is the same path as the unique shortest path in G_0 , then for each of the D missing clique edges, there is an additive distortion of at least $+2$. So in this case, $|p'| \geq |p| + 2D$.
2. If p'' is not the unique shortest path, it traverses at least two extra edges than the shortest one. This means p' must have gone through one of the layers of H_{k-1} in reverse. So once again, we have $|p'| \geq |p| + 2D$.

In either case, the distortion is at least $2D$, so H cannot be a $+(2D - 1)$ spanner. Now, the idea is to set $D|P_k| = |V|$. All that remains is to bound the sizes of H_k to get our final result.

As a warm up, we will start with the analysis of H_2 , which is essentially the same graph as the one used by Huang and Pettie [23]. We will express all values in terms of d , l , and ρ .

The number of vertices is determined by two factors: the vertex expansion step, and the edge expansion step. Each vertex in G_0 becomes $2\delta(d)$ vertices, so the number of vertices caused by vertex expansion is $2(l\rho)^{2d}l\delta(d)$. Each edge in G_0 becomes $2l - 1$

different vertices, so the number of vertices from edge expansion is $(l\rho)^{2d}(l)(2l-1)\delta(d)$. Thus the total number of vertices is $O((l\rho)^{2d}l^2\delta(d))$.

As a result of the edge expansion step, H_2 has become a very sparse graph. The number of edges from vertex expansion is $O((l\rho)^{2d}l\delta(d)^2)$, and the number of edges from edge expansion is $O((l\rho)^{2d}l^2\delta(d))$. So the total number of edges is equal to $O((l\rho)^{2d}(l + \delta(d))l\delta(d))$. Finally, the number of paths has not changed, which is still $O((l\rho)^{2d}\delta(d)^2)$.

Then, using Lemma 2.1.8, we need to set $|V| = |P|D$. Here, $D = O(l)$. Therefore we have, $(l\rho)^{2d}l^2\delta(d) = (l\rho)^{2d}l\delta(d)^2$, which results in $l = \delta(d)$. Using the fact that $\delta(d) = \Theta(\rho^{\frac{d-1}{d+1}})$, we get that $\rho = \Theta(l^{\frac{d+1}{d-1}})$. So we can write,

$$|V| = (l\rho)^{2d}l^2\delta(d) = O(l^{2d+3+\frac{2(d+1)}{d-1}})$$

We would like to minimize the value of the exponent; to do this, we can use either 2 or 3 dimensions. Then we get, $|V| = O(l^{13})$. This gives us the same $O(n^{1/13})$ lower bound that was given by Huang and Pettie. We now proceed to our new results.

For H_3, H_4, \dots , we will express all values $|V_i|$ and $|P_i|$ in terms of l . We also define L_i to be the length of each path in $|P_i|$. We make the following claim:

Claim 2.1.9. *In the graph H_k , with $k \geq 2$, we will have the following:*

- $L_k = O\left(l^{\left(\frac{6}{5}(1-6^{1-k})\right)}\right)$
- $|P_k| = O(l^1 2)$
- $|V_k| = O\left(l^{\left(12+\frac{6}{5}(1-6^{1-k})\right)}\right)$

Proof: We prove this by induction. For $k = 2$, we get $|V_2| = O(l^{13})$, $|P_2| = O(l^2)$, and $L_2 = O(l)$. All of these values match our calculations.

For H_k , we must substitute versions of H_{k-1} and \bar{H}_{k-1} for vertices. Note that each point has $\delta(d) = l$ inputs and outputs; this means H_{k-1} must have $O(l^2)$ paths. Using the inductive hypothesis, this implies that the length of the paths in this subgraph is equal to $L_{k-1}^{1/6}$. So, we have $L_k = O\left(l \cdot L_{k-1}^{1/6}\right) = O\left(l^{1+\frac{1-6^{-k}}{5}}\right) = O\left(l^{\left(\frac{6}{5}(1-6^{1-k})\right)}\right)$,

as desired for L_k . $|P_k|$ clearly stays the same size for all k . Then, since we have $|V_k| = O(L_k|P_k|)$, we have $|V_k| = O\left(l^{\left(12 + \frac{6}{5}(1-6^{1-k})\right)}\right)$ as well. So our claim has been proven.

This leads to the proof of **Theorem 1**. As we take $k \rightarrow \infty$, we can see that L_k approaches $l^{6/5}$ and $|V_k|$ approaches $l^{66/5} = L_k^{11}$. So, this construction creates graphs such that any $O(n)$ size spanner has at least $O(n^{1/11-\epsilon})$ additive stretch, for an arbitrarily small $\epsilon > 0$.

2.2 Future Directions

In this section we outline some of the (currently unsuccessful) attempts at improving the bound even further.

2.2.1 Avoiding Layered Structure

It should be noted that the graph from [11] is significantly denser than the graph used in [23], because the layered structure of Huang and Pettie's graph effectively reduces the dimension by one. It is natural to consider whether it is possible directly use a modified version of the original Coppersmith/Elkin graph, avoiding the layered structure.

We have tried this approach, but have had difficulty bounding the number of edges in the first graph product. A naive bound does not improve on the current best results, giving a lower bound of $O(n^{1/17})$ distortion, while a more detailed analysis has been difficult. We will show a progress sketch here.

Using a modification of a graph from the work of Coppersmith and Elkin [11], it is possible to create a "base graph" G with the following properties:

- There exists a path set P_G of size $O(l^{d-1}\rho^d\delta(d))$, where each path has length l
- $|V_G| = (l\rho)^d$
- The degree of each vertex is $\delta(d)$, and $|E_G| = O(l^d\rho^d\delta(d))$

Then, after applying a transformation similar to the graph product described in the previous sections, we have a graph G' with the following size constraints:

- $|P_{G'}| = |P_G|^2 = O(l^{2d-2}\rho^{2d}\delta(d)^2)$
- $|V_{G'}| = |V_G|^2 = O(l^{2d}\rho^{2d})$
- $|E_{G'}| = O(l^{2d}\rho^{2d}\delta(d))$

Notice that we have $|V_{G'}| = |V_G|^2$, while in the corresponding step of the layered graph we have $|V_{G_0}| = |V_{G_b}|^2/l$. This difference of a factor of l is ultimately what prevents us from getting a better bound with this method.

Finally we can apply an obstacle product to G' to get the graph G'' . The obstacle product is a matrix product very similar to the creation of the graph of H_2 . For the full details, please refer to Abboud and Bodwin [1]. The resulting graph will obey lemma 2.1.8 and has the following size constraints:

- $|P_{G''}| = |P_G|^2 = O(l^{2d-2}\rho^{2d}\delta(d)^2)$
- $|V_{G''}| = |V_G|^2 = O(l^{2d+1}\rho^{2d}\delta(d))$
- $|E_{G''}| = O(l^{2d+1}\rho^{2d}\delta(d))$

Furthermore the length of each path remains $O(l)$. Using lemma 2.1.8, we have that $l^{2d-1}\rho^{2d}\delta(d)^2 = l^{2d+1}\rho^{2d}\delta(d)$; from this we get that $\delta(d) = l^2$. Then we can simplify $|P_{G''}|$ to get $|P_{G''}| = l^{2d+3+4\frac{d+1}{d-1}}$. We seek to minimize the exponent, and this is achieved when we have $d = 3$ and $|P_{G''}| = l^{17}$.

This method gives $O(n^{1/17})$ stretch using our simple analysis, but we suspect that some of the sizes of vertex/edge sets are not tight. In particular, in the first graph product, I suspect it is possible to do better than $|V_{G'}| = |V_G|^2$.

Avoiding layered structure is a very promising idea because it saves the use of an entire dimension. If some of the bounds for the graph products can be tightened, there could very well be some improvements. Furthermore, we have briefly explored the idea of using a recursive method in this non-layered graph, but have not found any useful constructions.

2.2.2 k-Average free sets

Consider a set of positive integers S , with elements $\{a_i\}$. S is a k -average free set if the equation:

$$ka_i = \sum_{j=1}^k a_j$$

has only the trivial solution, $a_i = a_1 = a_2 = \dots a_k$. The "k-Average free sets" problem asks, if all integers are less than some number R , what is the maximum size of S with respect to R and k . This problem and related ones have been the subject a consider amount of research, and results have been used in many spanner constructions [1] [2]. We propose some extensions of the problem which, to the best of my knowledge, have been much less studied.

Multidimensional k-Average free sets: a d -dimensional k -average free set S is defined similarly to k -average free sets, but elements are tuples of length d , all bounded by some norm R . We would like to find the largest possible size of S in terms of R , d , and k .

Multidimensional Modular k-Average free sets: We define a d -dimensional k -average set S similarly to d -dimensional k -average free sets, however instead of addition, we use addition mod P for some positive integer P . We ask what is the maximum size of S in terms of k , d , and P .

Note that a convex hull of lattice points, as used in [11], is a natural way to find such sets. However I suspect it may not be optimal for several reasons:

- For Multidimensional k -average free sets, the size of the convex hull will have dependence on R but not k . There are be better solutions with $k = O(1)$, and I suspect there should be better solutions for $k = O(poly(R))$ as well.
- For Multidimensional Modular k -Average free sets, the convex hull used has radius $r = R/k$, and essentially doesn't try to take advantage of any features of modular addition.

In particular we are interested in improvements for $d = 2$ and $d = 3$, which

would immediately lead to better lower bounds. We note that the Multidimensional k -Average free sets problem may be quite difficult, as it seems several approaches to making k -average free sets do consider “flattening” higher dimensional tuples [1], with a blowup that is polynomial in d .

2.2.3 Other approaches

In our constructions we have determined that either 2 or 3 dimensions is optimal; however, note that the “ideal” value for d is actually a non-integral value. If there were some kind of lattice and convex hull on a space with dimension between 2 and 3, with the same bounds of $V_d(\rho)$, we could get an improved solution. We have experimented with using the Menger sponge as a lattice, but using a naive version of a convex hull doesn’t give the same properties of Claim 2.1.5.

Furthermore, note that our construction of G_0 was essentially a product of two base graphs. We have experimented with graph products utilizing more than two graphs. It seems like the naive graph products with, for example, 3 graphs gives the same result as the graph product with just 2 graphs. However, it may be possible to come up with some better bounds by using a logarithmic or polynomial number of base graphs in the graph product.

Chapter 3

Efficient Algorithms

A relatively unexplored area for spanners lies in the construction of efficient spanners. Most work done thus far has been on multiplicative spanners, from both centralized and distributed perspectives (see [34], [19], [30], [22] for selected examples).

There has been moderate success in creating efficient algorithms for the $O(1)$ gap additive spanners. Woodruff [36] showed that a +6 spanner with $O(n^{4/3})$ edges can be constructed in $O(n^2 \log n)$ time, although the spanner construction is slightly different than the one given in [18]. More recently, Knudsen [25] showed that a +2 spanner with $O(n^{3/2})$ edges, as well as a +8 spanner with $O(n^{4/3})$ edges, can both be constructed in $O(n^2)$ time. However, there is still no efficient construction for some version of Chechik's [14] $O(n^{7/5})$ +4 spanner, or any of the variety of sublinear spanners, to the best of my knowledge. Furthermore, there are almost no known results for fault tolerant spanners.

3.1 Matrix Multiplication

One natural way to reduce the runtime of the +2 spanner is with methods relating to utilize matrix multiplication. In fact, using the Seidel's [31] method, the all-pairs shortest path problem can be solved in $O(n^\omega)$ time, where ω is the best known exponent for matrix multiply. Given the distance matrix it is straightforward to generate an additive +2 spanner.

Since it is known that $\omega < 2.373$, the matrix multiply algorithm is already an improvement over the algorithm in Aingworth et. al [3], although it is not the most exciting result. We have some ideas for working towards a faster matrix-multiply algorithm, based on an algorithm that attempts to create BFS trees from a subset of all vertices, which we will present here.

Algorithm 1 Vertex Set Shortest Paths - a variation on Seidel [31]

function VSSP(A: An $n \times n$ matrix, k : a positive interger $\leq n$)
let $Z = A \cdot A$
let B be an $n \times n$ 0-1 matrix, where
 $b_{ij} = 1$ iff $i \neq j$ **and** ($a_{ij} = 1$ **or** $z_{ij} > 0$)
if $b_{ij} = 1$ for all $i \neq j$ **then**
return first k rows of of the $n \times n$ matrix $D = 2B - A$
let $T = \text{VSSP}(B, k)$
let $X = T \cdot A$
return $k \times n$ matrix D , where

$$d_{ij} = \begin{cases} 2t_{ij}, & \text{if } x_{ij} \geq t_{ij} \cdot \text{degree}(j) \\ 2t_{ij} - 1, & \text{if } x_{ij} < t_{ij} \cdot \text{degree}(j) \end{cases}$$

For the details on the correctness of this algorithm, see [31], the analysis is very similar to the analysis of his APD algorithm. If we let $k = O(\sqrt{n} \log(n))$, corresponding to a hitting set of vertices with degree greater than \sqrt{n} , VSSP can be used to create a +2 spanner using the method of [3].

We now consider the runtime of the algorithm. The number of recursive calls of VSSP is bounded by $\log(n)$, because each iteration uses a matrix corresponding to the square graph of the previous iteration. The time per iteration is dominated by two matrix products: the calculation of Z , which involves the product of two $n \times n$ matrices, and the calculation of X , which involves the product of a $k \times n$ matrix and a $n \times n$ matrix.

If $k = o(n)$, the calculation of X is faster than $O(n^\omega)$, but we have not found a way to circumvent the calculation of Z , which is necessary to generate the graph B . Note that the fast calculation of B is equivalent to a fast algorithm for boolean matrix multiply, so this problem may be difficult to solve directly. However, we suspect it

may be possible if the diameter of the graph is restricted to some value $O(n^\epsilon)$ for some $0 < \epsilon < 1$.

3.2 All Pairs Approximate Shortest Paths

Given an input graph G , an algorithm for All Pairs Approximate Shortest Paths (APASP) problem outputs a distance matrix D such that d_{ij} is close to $\text{dist}(i, j, G)$. Specifically, we are interested in the case where $\text{dist}(i, j, G) \leq d_{ij} \leq \text{dist}(i, j, G) + 2$. This problem is closely related to the +2 additive spanner problem. In [18], it is shown that the +2 APASP problem can be solved in $O(n^{7/3})$ time.

It should be noted that even if a spanner is known, it is not simple to extract an APASP distance matrix in a time faster than matrix multiply. In fact, in the algorithm given in [18], the APASP distance matrix does not correspond to any particular spanner.

Based on the results of [25], we have a new approach to solving the +2 version of the APASP problem; however, it currently also requires $O(n^{7/3})$ time. This removes a log factor from the result in [18], but we are hopeful that parts of it could be optimized for a polynomially faster runtime. The idea is to use a clustering algorithm in places where a dominating set was previously used. We first briefly describe the clustering algorithm used by Knudsen [25].

Let t be a parameter that can depend on G . For a sequence u_1, u_2, \dots, u_l of nodes we define the *clusters* C_1, C_2, \dots, C_l as follows:

$$C_i = (\Gamma_G(u_i) \cup \{u_i\})(C_1 \cup \dots \cup C_{i-1})$$

Next, we define a series of graphs G_0, G_1, \dots, G_l as follows. We have $G_0 = G$, and for each $i \geq 1$, we define G_i to be the subgraph of G that contains edge (u, v) if not both u and v are contained in $(C_1 \cup \dots \cup C_i)$.

Definition 3.2.1. A sequence u_1, u_2, \dots, u_l is called a t -clustering if the following requirements are satisfied:

- The node u_i maximizes the size of the cluster C_i
- Every cluster contains at least t nodes
- For every node v we have $|(\Gamma_G(v) \cup \{v\})(C_1 \cup \dots \cup C_l)| < t$

In summary, a t -clustering is created by recursively greedily selecting clusters, until there are no more vertices that can be clustered. This clustering has the following properties - see Knudsen's paper for the details.

Properties of t -clustering

- The number of edges in G_l is at most nt .
- Let $u, v \in V$ be a pair of nodes. If the shortest path from u to v is not contained in G_l , then there exists an index i such that

$$d_{T_i}(u_i, u) + d_{T_i}(u_i, v) \leq d_G(u, v) + 2$$

- Given a graph G and a parameter $t > 0$, we can construct a t -clustering u_1, \dots, u_l and the corresponding BFS trees T_1, \dots, T_l in $O(n^2)$ time.

We can create a $+2$ spanner with $O(n^{3/2})$ edges using t -clustering as follows:

Knudsen's $+2$ spanner. Set $t = \sqrt{n}$, and perform a t -clustering. The edge set containing every edge that contains an unclustered vertex, as well as the edges in the t -clustering, is a $+2$ spanner.

By the properties of t -clustering, it is straightforward to verify that Knudsen's $+2$ spanner has $O(n^{3/2})$ edges.

3.2.1 Preliminaries

In this section we will describe some of the tools used in our new APASP algorithm. We will start with the following lemma:

Lemma 3.2.1. *Suppose that v is part of cluster i and we have a t -clustering u_1, u_2, \dots, u_l . Let $D(v, u_i)$ be equal to the distance between v and u_j in the tree T_i . Then, $\text{dist}(v, u_j) \leq \min_{k \leq j} (D(v, u_k) + D(u_k, u_j)) + 2$*

Proof: By the properties of t -clustering, we can obtain the BFS trees T_1, \dots, T_l in $O(n^2)$ time. Then, let $D(u_i, v)$ denote the distance between u_i and v in the tree T_i . We let $D(u_i, v) = \infty$ if v is not contained in T_i .

Consider the shortest path P between some cluster center u_i and some vertex v . Consider the vertex that was clustered earliest; in other words, pick a vertex $w \in P$ with corresponding cluster center u_j such that j is minimized. Then, when u_j was chosen as a cluster center, all edges in P were still available in the graph. Thus we have,

$$D(v, u_j) + D(u_j, u_i) \leq \text{dist}(v, w) + 1 + \text{dist}(w, u_i) + 1 = \text{dist}(v, u_i) + 2$$

which proves the lemma. ■

The following lemma then immediately applies:

Lemma 3.2.2. *Suppose u_1, u_2, \dots, u_l is a set of cluster centers obtained from t -clustering. We can find approximate distances between the cluster centers and all other vertices to within a $+2$ additive stretch, in $O(nl^2)$ time.*

Proof: To approximate $\text{dist}(v, u_i)$ to within a $+2$ stretch we can take $\min_j (D(v, u_j) + D(u_j, u_i))$. As j can range from 1 through l , it takes $O(nl^2)$ time to compute all approximate distances, as desired. ■

3.2.2 Faster algorithms for $+4$ and $+6$ stretch

We can create faster algorithms by allowing a higher stretch factor, and in a few of these cases Knudsen's algorithm provides polynomial improvements. Our algorithm, which is based on the classic Dor, Halperin, and Zwick algorithm. The full algorithm is shown in Algorithm 3.2.2.

Algorithm 2 *APASP_k*: a variation on Dor, Halperin, and Zwick

Input: An unweighted undirected graph $G = (V, E)$
Output: A $n \times n$ matrix $\{\hat{\delta}(u, v)\}_{u, v}$ of estimated distances
For $i \leftarrow 1$ to $k - 1$ let $s_i \leftarrow n^{1-i/k}$
 $(\langle E_1, E_2, \dots, E_k, E^* \rangle, \langle D_1, D_2, \dots, D_k \rangle) \leftarrow \mathbf{decompose}(G, \langle s_1, s_2, \dots, s_{k-1} \rangle)$
Let D be an $\lceil k/2 \rceil$ matrix storing the approximation of lemma 3.2.2.
for every $u, v \in V$ **do**
 if $(u, v) \in E$ **then**
 $\hat{\delta}(u, v) \leftarrow 1$
 else if $(u, v) \in D$ **then**
 $\hat{\delta}(u, v) \leftarrow D[u, v]$
 else
 $\hat{\delta}(u, v) \leftarrow \infty$
for $i \leftarrow 1$ to k **do**
 for every $u \in D_i$ **do**
 run **Dijkstra** $((V, E_i \cup E^* \cup (\{u\} \times V) \cup (\cup_{i+j_1+j_2 \leq 2k+1} D_{j_1} \times D_{j_2})), \hat{\delta}, u)$

Analysis

Before we begin our analysis, we introduce the idea of the *type* of a vertex or edge.

Definition 3.2.2. Type A vertex is of type x if it has degree between $n^{(k-x)/k}$ and $n^{(k-x+1)/k}$. The type of an edge (u, v) is equal to the minimum of the type of u and the type of v . Let all vertices of type x or lower be denoted V_x . Furthermore, let all edges of type x or higher be denoted E_x .

First we consider the runtime of *APASP_k*. The algorithm differs from the one in [18] in that we have an extra step of finding approximate distances using our lemma. Our lemma requires $O(n(n^{\lceil k/2 \rceil/k})^2)$ time, which is at most $O(n^{2+1/k})$ time. The other steps take the same amount of time as in the original algorithm, $O(n^{2+1/k})$. So the total runtime of our algorithm is $O(n^{2+1/k})$.

Next we consider the amount of stretch. Our improvements are based off the following claim:

Claim 3.2.3. *Let P be the shortest path between two vertices, $u, v \in V$. If P contains a vertex of type $\lceil k/2 \rceil - 1$ or smaller, we will have $\hat{\delta}(u, v) \leq \text{dist}(u, v) + 4$.*

Proof: Consider the iteration of the algorithm where $i = \lceil k/2 \rceil$. Let w be the point on P closest to v of type $\lceil k/2 \rceil$ or smaller, and let w' be the cluster center of

w . Consider iteration $\lceil k/2 \rceil$ of the outer for loop, with Dijkstra's starting at u . Note that the edge (w, w') is in E_* , and all of the edges in P from w to v have type at least $\lceil k/2 \rceil$, which are in $E_{\lceil k/2 \rceil}$. Also, the edge (u, w') exists because the edge set includes $\{u\} \times V$. Therefore we have,

$$\begin{aligned}
\text{dist}(u, w', (V, E_{\lceil k/2 \rceil})) + \text{dist}(w', v, (V, E_{\lceil k/2 \rceil})) &= D[u, w'] + \text{dist}(w', v, (V, E_{\lceil k/2 \rceil})) \\
&\leq \text{dist}(u, w', G) + 2 + \text{dist}(w', v, G) \\
&\leq \text{dist}(u, w, G) + \text{dist}(w, v, G) + 4 \\
&= \text{dist}(u, v, G) + 4
\end{aligned}$$

Thus, the additive stretch is at most 4 as desired. The remainder of the analysis is very similar to the analysis of Dor, Halperin, and Zwick. We now define recursively the following sequence:

$$e_{i_1, j, i_2} = \begin{cases} 0 & \text{if } i_1 \leq j \\ 2 & \text{if } i_1 + j + i_2 \leq 2k + 1 \\ 4 & \text{if } j \geq \lceil k/2 \rceil \\ e_{i_1-1, j, i_1} & \text{otherwise} \end{cases}$$

Now we will prove the following claim inductively:

Claim 3.2.4. *If $u \in D_{i_1}$ and $v \in D_{i_2}$ are connected by a path P of length l in which the vertex of highest degree belongs to V_j , then $\delta_{i_1}(u, v) \leq l + e_{(i_1, j, i_2)}$.*

If $i_1 \leq j$, then all vertices in the path are present in $E_{i_1}(u)$, and the stretch is zero. This covers the first case.

If $i_1 + j + i_2 \leq 2k + 1$, with $j < i_1$, then $D_j \times D_{i_2} \in E_{i_1}(u)$. Let w be a vertex of type j on the path P , and w' be the corresponding cluster center. Note that both the edges (u, w') and (w', v) are in the edges of $E_{i_1}(u)$. Both of these distances must have been calculated in previous iterations. Thus we have, $\text{dist}_i(u, v) \leq$

$\text{dist}_i(u, w') + \text{dist}_i(w', v) = \text{dist}(u, w') + \text{dist}(w', v) \leq \text{dist}(u, v) + 2$, as desired. This covers the second case.

The third case is covered by Claim 3.2.3.

Finally, consider the case when $i_1 + j + i_2 > 2k + 1$, $j < i_1$, and $j > \lceil k/2 \rceil$. Since $j < i_1$, $V_j \in V_{i_1-1}$. Let w be the last vertex on P that belongs to V_{i_1-1} , and let w' be its corresponding cluster center. Then, all edges on P from w to v are contained in $E_{i_1}(u)$, so we have $\text{dist}_{i_1}(w', v) \leq \text{dist}_{i_1}(w, v) + 2 = \text{dist}(w, v) + 1$. Next, consider the path P' starting with the edge (w', w) , and then continuing through the edges of P until it reaches u . By the inductive hypothesis, we have $\text{dist}_{i_1-1}(w', u) \leq \text{dist}(w, u) + 1 + e_{i_1-1, j, i}$. Combining these results, we get that $\text{dist}_{i_1}(u, v) \leq e_{i_1-1, j, i} + 2$. This covers the last claim.

It remains to expand and bound the recursion for e_{i_1, j, i_2} . In fact, it will suffice to check $e_{k, j, k}$ for $1 \leq j \leq k$. It is not difficult to unroll the loop to get the following closed form result:

$$e_{k, j, k} = \begin{cases} 0 & \text{if } j = k \\ 2 & \text{if } j = 1 \\ 4 & \text{if } j \leq \lceil k/2 \rceil \\ \min\{2(k - j), 2\lceil j/2 \rceil + 2\} & \text{otherwise} \end{cases}$$

From this closed form we get **Theorem 2** and **Theorem 3**, by plugging in $k = 7$ and $k = 9$. The maximum stretch of 4 for $k = 7$ occurs when $j = 4$ or $j = 5$, and the maximum stretch of 6 for $k = 9$ occurs when $j = 6$. Our work has essentially resulted in the third case of the function being added. Unfortunately, with large k , the optimal value of j is equal to $2k/3 + O(1)$, which is larger than $\lceil k/2 \rceil$. This means our algorithm $APASP_k$ does not, based on our analysis, provide any speedup for any algorithms with stretch greater than 6.

3.2.3 Further attempts to improve APASP

We have (unsuccessfully) made efforts to improve APASP for larger values of stretch, and we detail some of our attempts in this section.

First, note that in Lemma 3.2.2, we are essentially performing a min-plus matrix multiplication. It is natural to try to use results from min-plus matrix multiplication to make improvements to the algorithm, but the current best known results for min-plus matrix multiplication are only better than brute force when all elements are $O(n^\epsilon)$ for some $\epsilon < 1$; the exact run time depends on the ϵ . This suggests that if the diameter is limited, there may be further improvements.

The above idea is especially hopeful because it works the best for small pairwise distances, while the algorithm generally, in some sense, works best for longer pairwise distances. This approach may lead to some small incremental improvements.

We note that Dor, Halperin, and Zwick had two algorithms for APASP - one for dense graphs, which we have modified for **Theorem 2** and **Theorem 3**, and one for sparse graphs. The case for dense graphs is more straightforward to modify using Knudsen's clustering algorithms, but we strongly suspect there may be a way to use the clustering algorithms for sparse graphs as well.

Chapter 4

Conclusions and Future Work

In this thesis we have detailed our work in a wide variety of problems related to spanners. It is our hope that our efforts and results will prove useful to future studies on spanners. We have two significant new findings, which we will summarize.

The first is an improvement on the additive stretch lower bounds for sparse spanners. By combining the approaches of two recent papers in the subject [23, 2], we have obtained a new lower bound for the exponent of the stretch factor. Our new lower bound is $1/11$, which is a slight improvement over the previous best lower bound of $1/13$. This construction also leads to improved bounds for a variety of other problems, such as shortcutting sets and additive emulators.

Our second result is an improvement in the All Pairs Almost Shortest Paths (APASP) problem, specifically for the cases of $+4$ and $+6$ stretch. Our approach was to modify the algorithm of Dor, Halperin, and Zwick using the greedy clustering method introduced by Knudsen [25]. These modifications give algorithms for $+4$ and $+6$ stretch in $O(n^{15/7})$ and $O(n^{19/9})$ time, respectively, an improvement over the previous bounds of $O(n^{11/5})$ and $O(n^{17/8})$. Unfortunately, we were unable to show any improvement over the original algorithm for spanners of other stretches.

Note that for both of these results, our new bounds are unlikely to be the optimal ones. In the following paragraphs, we go over some more ideas that we believe may have potential.

4.1 Knudsen’s clustering method and algorithm performance

Our main contribution to the APASP problem involved applying the clustering algorithm presented by Knudsen [25] to the current best known algorithm. Knudsen’s paper seems to have flown mostly under the radar, but may have potential applications in efficient algorithms for a variety of different problems. We describe a few ideas here.

Knudsen’s clustering algorithm may allow for an efficient construction for certain known fault tolerant spanners, such as [26]. Parter’s result for additive fault tolerant spanners is particularly interesting for this application because it utilizes a clustering method very similar to Knudsen’s. The limiting factor in this case is the fast construction of a fault-tolerant subset spanner.

Furthermore, there is still one spanner with $O(1)$ that does not have a fast algorithm - Chechik’s +4 spanner with $O(n^{7/5})$ edges [14]. As both the +2 and +6 spanners have known $\tilde{O}(n^2)$ algorithms for construction, it seems highly likely that +4 spanner should have one as well (although, it is still unclear if $O(n^{7/5})$ is the optimal size). We believe Knudsen’s clustering method may be useful here as well. The limiting factor in this case is a step that only adds paths to the spanner if there are a limited number of “heavy” vertices in the path. We have yet to find a way to do this step quickly.

4.2 Unexplored questions

In this section, we present some questions that are fundamental to spanners, but are, to the best of our knowledge, unexplored.

First, we have several constructions for various fault-tolerant spanners, giving upper bounds for the optimal spanner size for a fixed stretch function. However, there are no non-trivial lower bounds. New results here would likely follow a format similar to the lower bounds to “normal” additive spanners. A graph construction with

a certain number of provable critical edges would be required, and the lower bound for the spanner would be equal to the number of critical edges.

Second, we note that there are several problems closely related to both spanner construction and lower bounds - shortcutting sets, hopsets, emulators, and distance oracles are the most notable ones. However, there does not seem to be a well known relation between corresponding results in general, and each construction is treated on a case-by-case basis. We believe there are corresponding lower bounds to these problems based on **Theorem 1**, but have not been able to find them due to time constraints.

Bibliography

- [1] Amir Abboud and Greg Bodwin. The $4/3$ additive spanner exponent is tight. In *STOC*, pages 351–361, 2016.
- [2] Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *CoRR*, abs/1607.07497, 2016.
- [3] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
- [4] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.
- [5] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.
- [6] Imre Bárány and David G Larman. The convex hull of the integer points in a large ball. *Mathematische Annalen*, 312(1):167–181, 1998.
- [7] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β) -spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):5, 2010.
- [8] Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms-ESA 2015*, pages 167–178. Springer, 2015.
- [9] Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 855–872. Society for Industrial and Applied Mathematics, 2016.
- [10] Gregory Bodwin and Virginia Vassilevska Williams. Very sparse additive spanners and emulators. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 377–382. ACM, 2015.
- [11] B Bollobás, D Coppersmith, and M Elkin. Sparse subgraphs that preserve long distances and additive spanners. *SIAM J. Discr. Math*, 9(4):1029–1055, 2006.

- [12] Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Mathematics*, 19(4):1029–1055, 2005.
- [13] Gilad Braunschvig, Shiri Chechik, and David Peleg. Fault tolerant additive spanners. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 206–214. Springer, 2012.
- [14] Shiri Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 498–512. Society for Industrial and Applied Mathematics, 2013.
- [15] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.
- [16] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM (JACM)*, 47(1):132–166, 2000.
- [17] Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 169–178. ACM, 2011.
- [18] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.
- [19] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *CoRR*, abs/1607.08337, 2016.
- [20] Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 128–137. IEEE, 2016.
- [21] Michael Elkin and David Peleg. $(1 + \epsilon b)$ -spanner constructions for general graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 173–182. ACM, 2001.
- [22] Shay Halperin and Uri Zwick. Linear time deterministic algorithm for computing spanners for unweighted graphs, 1996.
- [23] Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *CoRR*, abs/1802.06271, 2018.
- [24] Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. In *Scandinavian Workshop on Algorithm Theory*, pages 277–281. Springer, 2014.
- [25] Mathias Bæk Tejs Knudsen. Additive spanners and distance oracles in quadratic time. *arXiv preprint arXiv:1704.04473*, 2017.

- [26] Merav Parter. Vertex fault tolerant additive spanners. *Distributed Computing*, 30(5):357–372, 2017.
- [27] Merav Parter and David Peleg. Sparse fault-tolerant bfs trees. In *European Symposium on Algorithms*, pages 779–790. Springer, 2013.
- [28] Merav Parter and David Peleg. Fault tolerant approximate bfs structures. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1073–1092. Society for Industrial and Applied Mathematics, 2014.
- [29] Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6(1):7, 2009.
- [30] Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010.
- [31] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of computer and system sciences*, 51(3):400–403, 1995.
- [32] Mikkel Thorup. On shortcutting digraphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 205–211. Springer, 1992.
- [33] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 802–809. Society for Industrial and Applied Mathematics, 2006.
- [34] Yashwanth Veluri, P Jayakumar, and Jyothisha J Nair. Computing multiplicative spanners efficiently for a class of simple graphs. In *2018 International Conference on Data Science and Engineering (ICDSE)*, pages 1–4. IEEE, 2018.
- [35] David P Woodruff. Lower bounds for additive spanners, emulators, and more. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 389–398. IEEE, 2006.
- [36] David P Woodruff. Additive spanners in nearly quadratic time. In *International Colloquium on Automata, Languages, and Programming*, pages 463–474. Springer, 2010.