Graph Spanners: A Tutorial Review

Reyan Ahmed[a], Greg Bodwin[c], Faryad Darabi Sahneh[a], Keaton Hamm[b], Mohammad Javad Latifi Jebelli[b], Stephen Kobourov[a], Richard Spence[a]

[a]*Department of Computer Science, University of Arizona*
[b]*Department of Mathematics, University of Arizona*
[c]*Department of Computer Science, Georgia Institute of Technology*

Abstract

This survey provides a guiding reference to researchers seeking an overview of the large body of literature about graph spanners. It surveys the current literature covering various research streams about graph spanners, such as different formulations, sparsity and lightness results, computational complexity, dynamic algorithms, and applications. As an additional contribution, we offer a list of open problems on graph spanners.

Contents

This work is supported in part by NSF grants CCF-1740858, CCF-0721503 and DMS-1839274.

1. Introduction

Given a graph G, a *graph spanner* (or simply *spanner*) is a subgraph which preserves lengths of shortest paths in G up to some amount of distortion or error, typically multiplicative and/or additive. Spanners were introduced by Peleg and Schäffer [136], and computing sparse or low-weight spanners has theoretical and practical applications in various network design problems, including graph compression and those in distributed computing and communication networks [18, 36]. This document provides a survey on the current literature of graph spanners, including different problem variants, hardness results, common techniques, and related open problems. In particular, geometric spanners are de-emphasized in this survey; we instead refer the reader to [109, 129]. The focus is on a method-based tutorial style which will allow those unfamiliar with the area to survey the problems considered and typical techniques used in the literature to compute spanners.

1.1. Organization and Layout

Throughout this survey, we use an annotated bibliography style in which important references are included within sections or subsections throughout the main text; this is done to enhance the readability of the survey so that the reader can find references quickly without going back and forth to the end of

the survey. Section 1.2 introduces notation and basic preliminaries which will be used throughout the survey. Section 2 gives a precise definition of various spanners considered in the literature as well as classical results and relationships to well-known concepts such as minimum spanning trees and Erdős' Girth Conjecture. Section 3 details the complexity and hardness of approximation for various spanner problems. Sections 4–9 comprise the bulk of the survey and describe the main techniques brought to bear on spanner problems to date including greedy algorithms, clustering and path buying algorithms, probabilistic constructions, ILP formulations, and LP-based algorithms. The emphasis of these sections is on illustrating the methods and inclusion of proofs to give the main ideas, as well as on posing open problems related to the works discussed. Section 10 concerns distributed algorithms. Section 11 briefly shows other kinds of spanners considered in the literature, while Section 12 discusses spanners when the underlying graph is allowed to change. Section 13 discusses spanners for restricted classes of graphs, and Section 14 ends the paper with some of the numerous applications of graph spanners. At the end of the paper, there are several summary tables for guarantees on the size and weight of spanners based on the type of problem.

1.2. Notation and Preliminaries

Graphs are denoted $G = (V, E)$ containing $|V| = n$ vertices (or nodes) and $|E| = m$ edges, and are assumed undirected and connected unless stated otherwise. We use uv to denote the undirected edge $\{u, v\}$, and (u, v) to denote the directed edge from u to v. We denote by $w_e = w_{uv}$ the weight of edge $e = uv$ (for weighted graphs), and denote by $d_G(u, v)$ the weight of a minimum-weight u-v path in G (or, the number of edges in a shortest path if G is unweighted). Given a subset $\bar{E} \subseteq E$ of edges, its weight is denoted by $W(\bar{E}) := \sum_{e \in \bar{E}} w_e$. In particular, we denote by $W(\text{MST}(G))$ the weight of a minimum spanning tree (MST) of G. Given two sets $A, B \subseteq V$, we define the distance from A to B in G as $\text{dist}_G(A, B) = \min\{d_G(u, v) : u \in A, v \in B\}$; if $A = \{v\}$ for some vertex v (resp. $B = \{v\}$), we will use $\text{dist}_G(v, B)$ (resp. $\text{dist}_G(A, v)$). The diameter of G is denoted by $\text{diam}(G) = \max_{u,v \in V} d_G(u, v)$. For degree-bounded graphs, we typically denote by Δ the maximum degree of any vertex in the graph.

Recall that for functions $f, g : \mathbb{R} \to \mathbb{R}_{\geq 0}$, we say that $f(n) = O(g(n))$ if there exist constants $C, n_0 > 0$ such that $f(n) \leq Cg(n)$ for $n \geq n_0$. Further, $f = \widetilde{O}(g)$ if $f = O(g \operatorname{polylog} n)$[1]. Additionally, $f = O_\varepsilon(g)$ if $f = O(\text{poly}(\varepsilon)g)$, where $\text{poly}(\varepsilon)$ is a polynomial in ε; this indicates that $f = O(g)$ for fixed ε, which is not necessarily the case if ε is allowed to vary.

We assume familiarity with the complexity classes P and NP. The complexity class $\text{DTIME}(f(n))$ (resp. $\text{NTIME}(f(n))$) represents the set of decision problems decidable by a deterministic (resp. non-deterministic) Turing machine in $O(f(n))$ time. The complexity class $\text{BPTIME}(f(n))$ represents the set of decision problems L solvable by a probabilistic algorithm in $O(f(n))$ time, such

[1] $\operatorname{polylog} n$ denotes a polynomial in terms of $\log n$, i.e., $a_k(\log n)^k + \ldots + a_1 \log n + a_0$.

that for all $x \in L$, the algorithm accepts x with probability at least $\frac{2}{3}$, and for all $x \notin L$, the algorithm accepts x with probability at most $\frac{1}{3}$.

Given an NP–hard optimization problem P, let $\mathcal{I}$ be the set of instances of P. If P is a minimization problem and $\alpha \geq 1$, we say that a polynomial-time algorithm A is an α–approximation if, for all instances $I \in \mathcal{I}$, $A(I)$ returns a feasible solution with cost c such that $\mathrm{OPT}_I \leq c \leq \alpha \mathrm{OPT}_I$ where OPT_I denotes the cost of the optimum solution for instance I. If P is a maximization problem, then A is an α–approximation ($\alpha \leq 1$) if $\alpha \mathrm{OPT}_I \leq c \leq \mathrm{OPT}_I$ for all instances I. Note that α may be a function of the size of the instance, or some other parameter related to the instance (such as the maximum vertex degree Δ of the input graph). In rarer cases, the approximation ratio given may be additive instead of multiplicative.

2. Graph Spanners

Given a graph G, possibly edge-weighted, a *graph spanner* (or *spanner* for short) is a subgraph G' which preserves lengths of shortest paths in G up to some error or distortion, e.g., additive and/or multiplicative error. There are several definitions and formulations of graph spanners. Most spanners can be specified using two parameters: a *distortion function* f which characterizes how much distances from the original graph are allowed to be distorted, and a subset $P \subseteq V \times V$ which prescribes pairs of vertices of the initial graph for which the distances need to be approximately preserved. That is, a subgraph $G' = (V', E')$ is a spanner with distortion f for $P \subseteq V \times V$ if

$$d_{G'}(u, v) \leq f\left(d_G(u, v)\right)$$

for all $(u, v) \in P$. It is worth noting that $d_G(u, v) \leq d_{G'}(u, v)$ for any subgraph G' of G; as such, we stipulate that the distortion function satisfies $f(x) \geq x$.

The following are typical choices for the distortion function f ~~(from most to least general)~~.

- **Multiplicative** (or **t-spanner**): distortion given by $f(x) = tx$ for some constant $t \geq 1$, where t is called the stretch factor. That is, distances are *stretched* by no more than a factor of t. Some authors use k instead of t to denote the stretch factor.
- **Additive**: distortion given by $f(x) = x + \beta$. In other words, distances in additive graph spanners are not *elongated* more than β units (or edges). These subgraphs preserve long distances with ratio close to 1. Additive spanners are sometimes called $+\beta$–spanners.
- **Linear** (or **(α, β)-spanner**): distortion given by $f(x) = \alpha x + \beta$ for $\alpha, \beta \geq 0$. In particular, a t-spanner is a $(t, 0)$-spanner, and an additive $+\beta$-spanner is a $(1, \beta)$-spanner.
- **Sublinear:** distortion given by $f(x) = x + o(x)$. In particular, distortions of the form $f(x) = x + O(x^{1-\frac{1}{k}})$ for positive integers k are of particular interest. Roughly, this is because this distortion function arises as

an *information-theoretic* barrier to compressing graph distances; that is, the most space-efficient data structures that approximate graph distances have essentially these error functions (but it is still open whether they are right for *spanners*; see [4] or Section 11.1 for further discussion). Like linear error, sublinear error can be viewed as a compromise between multiplicative and additive distortion, as it stretches small distances up to a constant multiplicative factor while the stretch factor for long distances approaches 1.

- **Distance Preservers:** no distortion, i.e., $f(x) = x$. These are $(1, 0)$–spanners.

Next, we illustrate different terminologies based on the subset $P \subseteq V \times V$ chosen (from most to least general).

- **Pairwise Spanners:** the given spanner condition must hold for specific pairs of vertices $P \subseteq V \times V$ (note that P is not necessarily symmetric).
- **Sourcewise Spanners:** the given spanner condition must hold for $P = S \times V$ for some subset $S \subseteq V$.
- **S–T (Source–Target) Spanners:** the given spanner condition must hold for $P = S \times T$ for subsets $S, T \subseteq V$ (not necessarily disjoint).
- **Subsetwise Spanners:** the given spanner condition must hold between all vertices in a fixed subset $S \subseteq V$. That is, subsetwise spanners are pairwise spanners where $P = S \times S \subseteq V \times V$.
- **Spanners:** with no additional terminology, it is typically implied that $P = V \times V$, i.e., distances are approximately preserved for all pairs of vertices.

Most spanner problems can be specified via the distortion parameters (α, β) and the pairs P of vertices for which distances must be preserved. To make notation concise, we typically write (α, β, P)–spanner, or (f, P)–spanner for general distortions.

Note also that while all variants of spanners are well-defined for weighted and unweighted graphs, it is somewhat more natural to consider additive spanners when the graph is unweighted. Indeed, if a graph G is weighted with arbitrarily large edge weights, then there exists β such that there is no additive β–spanner of G except for G itself.

2.1. Classical Results

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

[11] 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

[124] Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2):222–236, 1994

The classical and most common spanner problem is that of computing a sparse multiplicative t–spanner of an input graph, often called "basic t– (or k–) spanner" extensively throughout the literature. It is usually defined as follows:

Problem 1 (Basic t–Spanner Problem)**.** Given a connected graph $G = (V, E)$ and a fixed $t \geq 1$, find a subset $E' \subseteq E$ such that the subgraph $G' = (V, E')$ of G satisfies

$$d_{G'}(u, v) \leq t \cdot d_G(u, v), \quad \text{for all } u, v \in V. \tag{1}$$

The notion of a t–spanner was introduced by Peleg and Schäffer in [136] (see also Peleg and Ullman [137]), though the idea also appeared implicitly in earlier work of Awerbuch [18] and Chew [64]. Peleg and Schäffer [136] show that for unweighted graphs, determining if a t–spanner of G containing at most m edges exists is NP–complete. They also discuss at length a reduction of the t–spanner problem to particular classes of graphs – chordal graphs – and show that the generic lower bounds for the number of edges required to form a t–spanner for an arbitrary graph may be significantly improved for restricted graph classes.

Note that in order to check whether G' is a t–spanner of G, one only needs to check that inequality (1) holds for all edge pairs (u, v) where $uv \in E$, rather than for all $\binom{|V|}{2}$ vertex pairs, which is stated in the following simple proposition:

Proposition 2.1 ([136, Lemma 2.1])**.** *G' is a t–spanner of G if and only if $d_{G'}(u, v) \leq t \cdot d_G(u, v)$ for all $uv \in E$.*

Proof. The forward direction is obvious. For the reverse direction, let u and v be distinct vertices in V, and let $u_0 u_1 \dots u_m$ be a shortest u-v path in G, where $u_0 = u$ and $u_m = v$. Then note that by assumption,

$$d_{G'}(u, v) \leq \sum_{i=0}^{m-1} d_{G'}(u_i, u_{i+1}) \leq t \cdot \sum_{i=0}^{m-1} d_G(u_i, u_{i+1}) = t \cdot d_G(u, v).$$

□

Thus, one can check the spanner inequality (1) for $|E|$ pairs of vertices rather than for all $\binom{|V|}{2}$ vertex pairs to verify that a given subgraph is a t–spanner.

2.2. Sparsity and Lightness

Althöfer et al. [11] discuss a specific aspect of the t–spanner problem, namely *sparsity*. The idea is that a good t–spanner of a graph should contain very few edges while still approximately preserving distances in G. The sparsity of a spanner $G' = (V, E')$ is often defined as

$$\text{sparsity}(G') = \frac{|E'|}{|E|}.$$

Thus $0 \leq \text{sparsity}(G') \leq 1$ for any subgraph G' of G. Althöfer et al. [11] prove many lower and upper bounds for general graphs on the sparsity of a t–spanner, and also give a simple polynomial time algorithm for producing such a sparse spanner. One advantage of their method is that the algorithm presented provides a t–spanner not only with few edges, but also whose weight is comparable to the weight of the minimum spanning tree (in the case that G is a weighted graph). We will further discuss their results in Section 4.

For weighted graphs, a more natural consideration is the *lightness* of a spanner, which is related to the total weight of the spanner. It is usually compared with the weight of the minimum spanning tree. We may thus define

$$\text{lightness}(G') = \frac{W(E')}{W(\text{MST}(G))}.$$

Note that $\text{lightness}(G') \geq 1$ since the MST is the sparsest subgraph connecting all the nodes.

Often we are interested in finding the sparsest or lightest spanner of a given graph:

Problem 2 (Sparsest/Lightest Spanner problem)**.** Given a graph $G = (V, E)$, distortion f, and $P \subseteq V \times V$, find $G' = (V', E')$ such that

1. G' is a (f, P)–spanner for G, and
2. $\text{sparsity}(G') \leq \text{sparsity}(H)$ for all other (f, P)–spanners H if G is unweighted, or
3. $\text{lightness}(G') \leq \text{lightness}(H)$ for all other (f, P)–spanners H if G is weighted.

Kortsarz and Peleg [124] show that the problem of finding the sparsest 2–spanner of an unweighted graph $G = (V, E)$ admits a polynomial time $\log\left(\frac{|E|}{|V|}\right)$–approximation. Note that this problem is equivalent to finding an edge set $E' \subseteq E$ such that for every edge $e \in E \setminus E'$, there is a triangle (3–cycle) in G containing e whose remaining two edges belong to E'.

To describe their approximation algorithm, note that given $U \subseteq V$, the *density* of U, denoted $\rho_G(U)$, is defined by $\rho_G(U) = \frac{|E(U)|}{|U|}$, where $(U, E(U))$ is the subgraph of G induced by U. The *maximum density* problem is to find vertex subset $U \subseteq V$ such that $\rho_G(U)$ is maximized, which can be solved in polynomial time.

The algorithm given in [124] maintains three sets of edges: H^s, the set of edges in the spanner; H^c, the set of "covered edges" (edges either in the spanner, or edges on a triangle containing two edges in H^s); H^u, the remaining unspanned edges. Given a set H^u of unspanned edges and vertex v, the authors denote by $N(H^u, v)$ the subgraph of G whose vertex set is $N(v)$[2], and whose edge set is the set of edges in G induced by $N(v)$, which are also unspanned (i.e., $E(N(v)) \cap H^u$). The authors denote by $\rho(H^u, v)$ the maximum density of this neighborhood graph over all vertices $v \in G$.

The idea of the algorithm is: while $\rho(H^u, v) > 1$, find a vertex $v \in G$ such that $\rho(H^u, v)$ is maximum. Then within this restricted neighborhood graph, solve the maximum density problem to find a corresponding dense subset U_v of neighbors of v. Add the edges from v to each $u \in U(v)$ to the set of spanner

[2] the set of neighbors of v in G

edges H^s. The edges "covered" by the newly-added edges in H^s are added to H^c, and H^u also updates accordingly.

Since at least one edge is added to H^c at each iteration, the number of iterations is at most m. The authors show that, using the maximum density problem as a subroutine, a sparse 2-spanner with approximation ratio $O\left(\log \frac{|E|}{|V|}\right) = O(\log n)$ can be found in $O(m^2 n^2 \log(\frac{n^2}{m}))$ time.

2.3. Relation to Minimum Spanning Trees

Recall that G' is a *spanning tree* of G if G' is connected, acyclic, and spans all vertices of G. The well-known *minimum spanning tree* (MST) problem is to compute a spanning tree of minimum total weight. This can easily be done in polynomial time (e.g., using Kruskal's or Prim's algorithm). We denote by $W(\text{MST}(G))$ the weight of an MST of G.

Note that the MST of a graph can be an arbitrarily poor spanner; for example, let G be an unweighted cycle with $|V|$ edges. The MST is obtained by deleting any edge; however, the MST is only a $(|V|-1)$–spanner of the original graph. The MST problem can be interpreted as a special case of the t–spanner problem where t is arbitrarily large.

2.4. Erdős' Girth Conjecture

[103] Paul Erdős. Extremal problems in graph theory. In *Proceedings of the Symposium on Theory of Graphs and its Applications*, page 2936, 1963

[102] P. Erdős and M. Simonovits. Some extremal problems in graph theory. In *Combinatorial Theory and its Applications, I (Proc. Colloq., Balatonfüred, 1969)*, pages 377–390. North-Holland, Amsterdam, 1970

[130] Merav Parter. Bypassing Erdős' girth conjecture: Hybrid stretch and sourcewise spanners. In *Automata, Languages, and Programming*, pages 608–619. Springer, 2014

Recall that the *girth* of a graph G is defined as the length of the smallest cycle in G, and is infinity if G is acyclic. Letting $\gamma(n,k)$ denote the maximum possible number of edges in an n–vertex graph with girth $> k$, the authors in [11] give a simple algorithm that, for any n–vertex (undirected, possibly weighted) input graph and positive integer t, produces a t–spanner on $|E'| \leq \gamma(n, t+1)$ edges. This is best possible in the following sense:

Proposition 2.2. *An undirected unweighted graph $G = (V, E)$ of girth $> t+1$ has no proper subgraph that is a t–spanner.*

Proof. Consider any edge $uv \in E$. If uv is removed from G, then $d_G(u,v)$ changes from 1 to one less than the length of the shortest cycle containing uv in the original graph. Hence, if G has girth greater than $t+1$, then $d_{G\setminus\{uv\}}(u,v) > t$, so the spanner property is not satisfied for the pair (u,v) after the edge is removed. □

Thus, if we consider an n–vertex graph $G = (V, E)$ with girth greater than $t+1$ and $|E| = \gamma(n, t+1)$ edges, the only t–spanner of G is G itself. It follows that no algorithm can improve in general on the density bound of [11] with, say, a bound of $|E'| \leq \gamma(n, t+1) - 1$.

It still remains a major open problem to determine $\gamma(n,k)$, even asymptotically, though upper bounds called the *Moore Bounds* are given by a folklore counting argument:

Proposition 2.3. $\gamma(n,k) = O\left(n^{1+\frac{1}{\lfloor k/2 \rfloor}}\right)$.

Sketch of Proof. See [11] for full detail. Let $G = (V,E)$ be a graph with $|V| = n$ and girth $> k$, and let d be its average vertex degree. There are two steps in the proof. First, we find a nonempty subgraph $H \subseteq G$ of minimum degree $\Omega(d)$. We generate H by iteratively deleting any vertex in G of degree $\leq \frac{d}{4}$; one counts that at most $\frac{|E|}{2}$ edges are deleted in total, so H remains nonempty. Second, pick an arbitrary vertex v in H and consider a BFS tree T rooted at v to depth $\lfloor \frac{k}{2} \rfloor$. Since H has girth $> k$, each vertex in T can have only one edge to a vertex in the same or previous layer of the tree (else we would have a cycle of length $\leq k$). Since every vertex in T has degree $\Omega(d)$, one counts that there are $\Omega(d)^{\lfloor \frac{k}{2} \rfloor}$ total vertex in T. Thus $d = O\left(n^{\frac{1}{\lfloor k/2 \rfloor}}\right)$, so $|E| = O\left(n^{1+\frac{1}{\lfloor k/2 \rfloor}}\right)$. □

Erdős' Girth Conjecture is the statement that the Moore Bounds are tight; that is, $\gamma(n,k) = \Omega(n^{1+\frac{1}{\lfloor k/2 \rfloor}})$.[3] The conjecture generally plays two important roles in the literature on spanners. First, in some applications of spanners it is useful to know the number of edges the spanner could possibly have, and this currently requires the Girth Conjecture. Second, there are some known constructions of t–spanners on $O(n^{1+\frac{1}{\lfloor (t+1)/2 \rfloor}})$ edges with certain desirable properties relative to [11] (for example, Baswana and Sen [28] give a linear time algorithm to produce spanners of this quality). Whereas [11] produces spanners with optimal size/distortion tradeoff *regardless* of the truth of the Girth Conjecture, these other constructions have optimal tradeoff only if the Girth Conjecture is assumed. Thus, many natural algorithmic questions about spanners (such as linear time computation of spanners with optimal size/distortion tradeoff) are currently closed only if the Girth Conjecture is assumed.

So far, we have discussed the Girth Conjecture with respect to multiplicative spanners, but it more generally constrains spanners with additive or mixed error in the same way. Specifically, assuming the Girth Conjecture, any construction of (α,β)–spanners of size $O(n^{1+\frac{1}{k}})$ must have $\alpha + \beta \geq 2k-1$. The proof of this is essentially the same as that of Proposition 2.2. Independent of the veracity of the girth conjecture, Woodruff [153] proved this fact in the setting $\alpha = 1$: for any $k \in \mathbb{N}$, there exists a graph on n nodes for which any $(1, 2k-2)$–spanner has $\Omega(k^{-1}n^{1+\frac{1}{k}})$ edges.

[3] The conjecture comes from [103], page 5, where Erdős wrote "It seems likely that" a certain equation holds (7) which is equivalent to the statement $\gamma(n,k) = \Omega_k(n^{1+\frac{1}{\lfloor k/2 \rfloor}})$. Most modern papers quote a strengthened version of this statement where the implicit constant may not even depend on k.

2.5. Open Problems

1. Considering discontinuous distortion functions would also be of interest, e.g., one could allow small distances to be distorted more than large ones, or require distance preservation of close vertices, but small distortion for far away ones.

2. Erdős' Girth Conjecture is known to be true for $k = 1, 2, 3$, or 5 [152, 151]. Is it true for other values of k?

3. NP-hardness and Hardness of Approximation

Most interesting spanner problems are NP-complete. In this section, we focus primarily on hardness results for the basic t–spanner and additive spanner problems. For hardness results on other types of spanner problems (e.g., tree t–spanner or lowest-diameter t–spanner), see Section 11.

3.1. NP-hardness of the basic t–spanner problem

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

[52] Leizhen Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics*, 48(2):187–194, 1994

Peleg and Schäffer [136] show that, given an unweighted graph G, and integers $t, m' \geq 1$, determining if G has a t–spanner containing m' or fewer edges is NP–complete, even when t is fixed to be 2. The reduction is from the edge dominating set[4] (EDS) problem on bipartite graphs, defined as follows: given a bipartite graph $H = (V, E)$ with bipartition (X, Y), and an integer $K \geq 1$, determine if H has an edge dominating set consisting of at most K edges. The reduction can be summarized as follows: given an instance $\langle H = (V, E), K\rangle$ to EDS where H has bipartition (X, Y), $X = \{x_1, \ldots, x_{n_x}\}$, and $Y = \{y_1, \ldots, y_{n_y}\}$, let $G(H)$ be constructed by adding a vertex x_{ij} for each $1 \leq i < j \leq n_x$, and similarly for Y. Let $V(G(H)) = X \cup Y \cup \{x_{ij} \mid 1 \leq i < j \leq n_x\} \cup \{y_{ij} \mid 1 \leq i < j \leq n_y\}$.

For each $1 \leq i < j \leq n_x$, add edges $x_i x_j$, $x_i x_{ij}$, and $x_j x_{ij}$, and do so similarly for Y. The authors define E_{XY} as the union of all of these added edges. Let $E(G(H)) = E \cup E_{XY}$, $t = 2$, and $m' = K + n_x(n_x - 1) + n_y(n_y - 1)$. One can show that $\langle H, K\rangle$ is accepted iff $\langle G(H), t, m'\rangle$ is accepted.

Cai [52] also shows that for any fixed $t \geq 2$, the minimum t–spanner problem is NP–hard, and for $t \geq 3$, the problem is NP–hard even when the input is restricted to bipartite graphs. The reduction is from the 3–SAT problem.

[4] An *edge dominating set* of a graph $H = (V, E)$ is a subset of edges $E' \subseteq E$ such that every edge $e \in E \setminus E'$ is adjacent to at least one edge in E'.

3.2. Label Cover and hardness of approximation of the basic t–spanner problem

[13] Sanjeev Arora and Carsten Lund. *Hardness of Approximations*, page 399–446. PWS Publishing Co., USA, 1996

[96] Michael Elkin and David Peleg. Approximating k–spanner problems for $k > 2$. *Theoretical Computer Science*, 337(1-3):249–277, 2005

[97] Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory of Computing Systems*, 41(4):691–729, Dec 2007

[76] Michael Dinitz, Guy Kortsarz, and Ran Raz. Label cover instances with large girth and the hardness of approximating basic k-spanner. *ACM Transactions on Algorithms (TALG)*, 12(2):25, 2016

For $k = 2$, the basic k–spanner problem admits an $O(\log n)$ approximation ([124], Section 2.1). Dinitz et al. [76] show that for $k \geq 3$, and for all $\varepsilon > 0$, the basic k–spanner problem cannot be approximated with ratio better than $2^{(\log^{1-\varepsilon} n)/k}$ unless NP $\subseteq$ BPTIME($2^{\text{polylog}(n)}$) (This result was initially claimed by Elkin and Peleg [94], but retracted due to error). Elkin and Peleg [96] propose approximation algorithms with sublinear approximation ratio, and study certain classes of graphs for which logarithmic approximation is feasible. Elkin and Peleg [97] extend the hardness results to other spanner problems, and also show strong inapproximability for the *directed* unweighted k–spanner problem.

Many strong hardness results rely on the hardness of *Label Cover*, initially defined by Arora and Lund [13], which may be stated as follows.

Definition 3.1 (Label Cover)**.** Let $G = (V, E)$ be a biregular bipartite[5] graph with bipartition $V = V_1 \cup V_2$. Additionally, let Σ_1 and Σ_2 be two alphabets representing the sets of possible labels associated with V_1 and V_2. For every edge $e \in G$, there is a nonempty relation $\pi_e \subseteq \Sigma_1 \times \Sigma_2$. A *labeling* is an assignment of one or more labels to every vertex in V. We say that edge $e = uv$ ($u \in V_1$, $v \in V_2$) is *covered* by the labeling if there is a label a_1 assigned to u and a label a_2 assigned to v, such that $(a_1, a_2) \in \pi_e$.

Often, Label Cover is stated as a maximization problem: every vertex in G is assigned exactly one label, and the goal is to maximize the number of covered edges. It is known that both the maximization and minimization versions are quasi-NP–hard to approximate with ratio $2^{\log^{1-\varepsilon} n}$ for all $\varepsilon > 0$ [13].

Dinitz et al. [76] show that if there is an additional requirement that the girth of G is larger than k, where $3 \leq k \leq \log^{1-2\varepsilon} n$, then the maximization version of Label Cover cannot be approximated with ratio better than $2^{(\log^{1-\varepsilon} n)/k}$ unless NP $\subseteq$ BPTIME($2^{\text{polylog}(n)}$), for sufficiently large n. By considering a variant of the minimization problem of Label Cover (Min-Rep), they show the hardness of the basic k–spanner problem; specifically, it is also hard to approximate the basic k–spanner problem within a factor better than $2^{(\log^{1-\epsilon} n)/k}$.

[5] A *biregular bipartite* graph is a graph with bipartition $V = V_1 \cup V_2$ such that every vertex in V_1 has the same degree, and every vertex in V_2 has the same degree.

3.3. NP–Hardness for Planar Graphs

[49] Ulrik Brandes and Dagmar Handke. NP-completeness results for minimum planar spanners. *Discrete Mathematics and Theoretical Computer Science*, 3(1), 1998

[121] Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theoretical Computer Science*, 746:88–97, 2018

Brandes and Handke [49] show that the basic t–spanner problem is NP–complete for fixed $t \geq 5$ when the input graph is planar and unweighted, and is also NP–complete for fixed $t \geq 3$ when the input graph is planar and weighted. For unweighted graphs, the reduction is from *planar 3–SAT*, a variant of 3–SAT where the underlying bipartite graph induced by the variables and clauses is planar. Kobayashi [121] recently showed that the minimum t–spanner problem is NP–hard on planar graphs for $t \in \{2, 3, 4\}$.

3.4. NP–Hardness of Additive Spanners

[49] Arthur Liestman and Thomas Shermer. Additive graph spanners. *Networks*, 23:343 – 363, 07 1993

[122] Yusuke Kobayashi. An FPT algorithm for minimum additive spanner problem. *preprint, arXiv:1903.01047*, 2019

Liestman and Shermer [127] show that for all integers $\beta \geq 1$, determining if a graph G contains an additive $+\beta$-spanner containing m' or fewer edges is NP–hard via a reduction from the edge dominating set problem.

Kobayashi [122] considers a parametrized version of the additive $(1, \beta)$–spanner problem where the number of removed edges is regarded as a parameter k, and a fixed-parameter algorithm is given for it. The main result is that there exists a fixed-parameter tractable algorithm for the Parameterized Minimum Additive $(1, \beta)$–spanner problem that runs in $(\beta + 1)^{O(k^2+\beta k)}mn$, or $2^{O(k^2)}mn$ if β is fixed.These results are generalized for (α, β)–spanners. However, we remark that these algorithms run in polynomial time only if one wishes to remove only a constant number of edges. In many settings of interest one hopes for a spanner that is much sparser than the input graph, and for these a different algorithmic paradigm is needed.

3.5. Open Problems

1. Kobayashi [121] leaves as an open question whether the minimum t–spanner problem on bounded-degree graphs of degree at most Δ is NP–hard for certain fixed t and Δ, namely $(t, \Delta) = (2, 5)$, $(2, 6)$, $(2, 7)$, $(3, 4)$, $(3, 5)$, $(4, 3)$, $(4, 4)$, $(4, 5)$.

4. The Greedy Algorithm for Multiplicative Spanners

[11] 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

[105] Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 9–17. ACM, 2016

One of the original methods for constructing spanners is to use a greedy algorithm. The essential idea is to first sort the edges in nondecreasing order by weight, choose the edge with the smallest weight first, and then each subsequent edge is chosen or not according to some criteria which guarantees that the end result is a spanner of the desired type. Despite being the oldest construction of spanners, greedy algorithms remain one of the most utilized methods for achieving this task.

4.1. Kruskal's Algorithm for Computing Minimum Spanning Trees

There are many algorithms for computing MSTs, including Borůvka's, Prim's, and Kruskal's algorithm. All of these are examples of greedy algorithms. Since it is the basis for the greedy algorithm to construct t–spanners, we review Kruskal's algorithm in Algorithm 1.

Algorithm 1 MST($G = (V, E)$); Kruskal's MST Algorithm

```
Sort edges in nondecreasing order of weight
G' = (V, E' ← ∅)
for uv ∈ E do
    if d_{G'}(u, v) = ∞ (i.e. u, v are previously disconnected in G') then
        E' ← E' ∪ {uv}
    end if
end for
return G'
```

4.2. The Greedy Algorithm for Multiplicative Spanners

Althöfer et al. [11] proposed and analyzed the first greedy algorithm for computing a sparse t–spanner of a weighted graph; see Algorithm 2.

Algorithm 2 GreedySpanner($G = (V, E)$, t); Greedy t–spanner Algorithm

```
Sort edges in nondecreasing order of weight
G' = (V, E' ← ∅)
for uv ∈ E do
    if d_{G'}(u, v) > t · w_{uv} then
        E' ← E' ∪ {uv}
    end if
end for
return G'
```

Note first, that Algorithm 2 guarantees that the lowest weighted edge is chosen. Likewise, if at any iteration in the for loop there is no path in G' from u to v, then the edge uv is chosen. This algorithm is based on Kruskal's algorithm, but the key difference is that Algorithm 2 does not enforce the condition that G' is acyclic. Indeed, suppose edges u_1u_2 and u_1u_3 have been chosen already to

be in E', and that u_2u_3 is an edge in the original graph G, and the shortest path in G from u_2 to u_3 is the two-edge path going from u_2 to u_1 to u_3. Kruskal's algorithm would reject the new edge because its endpoints are already connected. However, provided $t \cdot w_{u_2u_3} < w_{u_2u_1} + w_{u_1u_3}$, this edge would be added to E'.

That said, an analogous cycle-free property holds for the greedy algorithm: a graph returned by Algorithm 2 with parameter t will not have any cycles on $\leq t+1$ edges. To see this, let C be a cycle on $\leq t+1$ edges in the input graph G, and let $uv \in C$ be the last edge considered by the greedy algorithm. When edge uv is considered, either another edge in C has been discarded, or else (due to the edge ordering) there is a u-v path through C of length $\leq t \cdot w_{uv}$, and thus we will discard uv. In either case, C does not survive in the output graph G'. Thus, one can reasonably view Kruskal's algorithm as the special case of Algorithm 2 with $t = \infty$.

Althöfer et al. prove that any G' constructed from Algorithm 2 is a t–spanner for G. However, we note here that their proof is actually not dependent upon the greedy reordering of the edges, which is an interesting fact in its own right.

Proposition 4.1 ([11])**.** *Algorithm 2 yields a t–spanner for G regardless of the ordering of the edges in the first step.*

Proof. First, we may assume without loss of generality that for each edge uv in the input graph G, we have $d_G(u,v) = w_{uv}$. Otherwise, we may remove uv from G without changing its shortest path metric at all, and thus any spanner of the remaining graph is also a spanner of G itself.

For each $uv \in E$, when we consider uv in the greedy algorithm, we either have

$$d_{G'}(u,v) \leq t \cdot w_{uv} = t \cdot d_G(u,v),$$

or else we add uv to G' and thus have $d_{G'}(u,v) = d_G(u,v)$. In either case, the pair (u,v) satisfies the spanner property. The proposition then follows from Proposition 2.1. □

Despite the fact that Algorithm 2 yields a t–spanner for any ordering of the edges, the rest of the analysis of Althöfer et al. crucially hinges upon the greedy edge ordering. In [11, Lemma 2], it is shown that the output G' of Algorithm 2 is such that its girth is at least $t+1$. However, this need not hold if we allow the algorithm to run with a different edge ordering as the example in Figure 1 demonstrates.

Remark 4.1. It should be noted that the greedy algorithm presented here essentially relies on Proposition 2.1, which allows one to only enforce the t–spanner condition on edges in the initial graph G. This proposition does not hold for additive spanners (for example), and so to find a greedy algorithm to produce an additive (or mixed) spanner, something else would need to be done. Indeed, [119] might be viewed as a greedy algorithm for additive spanners, but for this reason it requires an extra preprocessing step before the greedy part, and thus is not a direct analog of the multiplicative greedy algorithm discussed here.

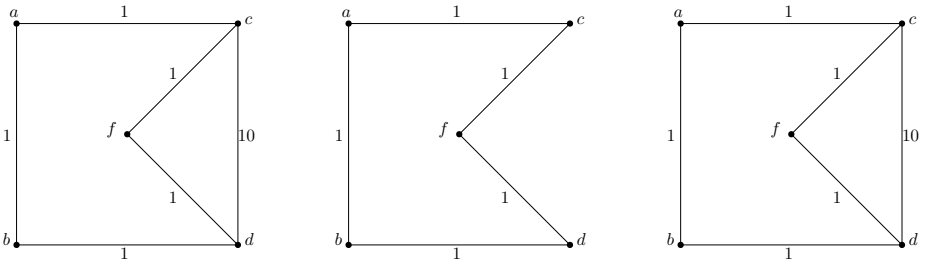

Figure 1: Ordering the edges by their weights is pivotal for the girth property of the greedy $t-$spanner. The rightmost graph would be output of the algorithm for $t = 3$ if we allowed non-ordered edges by picking the edge cd with weight 10. This graph has girth 3, which is smaller than $(t + 1) = 4$.

4.3. Existential Optimality of Greedy t–spanners

Garay et al. [107] (see also [135, Chapter 24]) distinguish between different notions of optimality of a given algorithmic output. The essential difference is one of a single quantifier: namely one is a "for all" statement and the other is a "there exists" statement. An algorithmic solution is *universally optimal* for a given class of graphs $\mathcal{G}$ if for every graph $G \in \mathcal{G}$, the algorithm gives the optimal solution. On the other hand, an algorithm is *existentially optimal* for a class of graphs $\mathcal{G}$ provided there exists a graph $G \in \mathcal{G}$ for which the algorithm constructs the optimal solution.

Filtser and Solomon [105], following [11], prove existential optimality of the greedy t–spanner; although they prove something somewhat stronger than the notion of Garay et al. described above. They prove that the greedy t–spanner for a graph G in a given (fixed) class of graphs $\mathcal{G}$ is never worse than the worst-case optimal solution for the whole class $\mathcal{G}$. For example, given a class of graphs $\mathcal{G}$ on n vertices, the worst-case number of edges in an optimal t–spanner is given by

$$m(t) := \sup_{G \in \mathcal{G}} \inf_{\substack{G'=(V,E'), \\ G' \text{ is a } t\text{-spanner for } G}} |E'|.$$

Then the greedy t–spanner algorithm is such that for every $G \in \mathcal{G}$, the algorithm's output is a t–spanner with at most $m(t)$ edges. In particular, this implies that the greedy t–spanner is existentially optimal because there exists a graph G whose optimal t–spanner has $m(t)$ edges, and its greedy t–spanner has at most $m(t)$ edges as well, and hence is an optimal solution.

In our setting, optimality of a t–spanner construction will be considered via two parameters, the number of edges in the spanner, and its total weight. Filtser and Solomon prove that the greedy t–spanner algorithm is existentially optimal. We state their result in a stronger way here, but in such a way that the proof becomes much simpler than the original. To begin, we state a crucial lemma which is interesting in its own right, which states that the only t–spanner of a greedy t–spanner is itself.

Lemma 4.1 ([105, Lemma 3]). *Let $t \geq 1$ be fixed, and let G be any weighted connected graph with n vertices. Let G' be the greedy t–spanner of G. If G'' is a t–spanner of G', then $G'' = G'$.*

Proof. Suppose $G' = (V, E')$ and $G'' = (V, E'')$. By way of contradiction, suppose that there exists an edge $e = uv \in E' \setminus E''$. Let P be a shortest path in G'' connecting u and v (and note that $e \notin P$). Consider the last edge in $P \cup \{e\}$, say $\tilde{e}$, examined by the greedy algorithm when forming G'. Since the edges are sorted, we have $w_e \leq w_{\tilde{e}}$. Since $P \cup \{e\}$ lies in G', and each of these edges has weight at most that of $\tilde{e}$, it follows that all edges in $P \cup \{e\} \setminus \{\tilde{e}\}$ have already been added to E' by the time the greedy algorithm examines $\tilde{e}$. Thus this set forms a path connecting the endpoints of $\tilde{e}$, and we have

$$W(P) - w_{\tilde{e}} + w_e \leq W(P) \leq t w_e \leq t w_{\tilde{e}},$$

which implies that the greedy algorithm will not add edge $\tilde{e}$ to E', which is a contradiction. Hence no such e exists, and $E'' = E'$. □

Theorem 4.1 ([105, Theorem 4]). *Suppose that $\mathcal{G}$ is a class of graphs on n vertices which is closed under edge deletion. Let $t > 1$ be fixed. Let*

$$m(t, n) := \sup_{G \in \mathcal{G}} \inf_{\substack{G' = (V, E'), \\ G' \text{ is a } t\text{-spanner for } G}} |E'|,$$

and

$$\ell(t, n) := \sup_{G \in \mathcal{G}} \inf_{\substack{G' = (V, E'), \\ G' \text{ is a } t\text{-spanner for } G}} W(E').$$

Then for every $G \in \mathcal{G}$, the greedy t–spanner, G' of G has at most $m(t, n)$ edges and weight at most $\ell(t, n)$.

Proof. The key ingredient is that since a t–spanner of $G \in \mathcal{G}$ can be obtained by edge deletion, it must be in $\mathcal{G}$ as well. Combining this observation with Lemma 4.1 yields the desired bounds immediately. Indeed, let $G \in \mathcal{G}$ be arbitrary, and let G' be its greedy t–spanner. Since G' can be obtained from G by edge deletion, it is in $\mathcal{G}$; hence by assumption G' has a t–spanner, say G'', which has at most $m(t, n)$ edges and weight at most $\ell(t, n)$. But since $G'' = G'$, the proof is complete. □

For a proof of optimality of the greedy algorithm for geometric graphs, see [48].

4.4. Arbitrarily Bad Greedy t–spanners

Universal optimality of course implies existential optimality, but the reverse is patently untrue. Here, we note that not only is the greedy t–spanner not universally optimal, but moreover there is a whole family of graphs for which the greedy t–spanner is as far away as possible from the optimal t–spanner. This family of examples is inspired by the one given by Filtser and Solomon [105]

based on the Petersen graph. Consider a complete bipartite graph on n edges (seen in Figure 2) where each edge has weight 1. Subsequently, we add an extra vertex, which is connected to every vertex in the original bipartite graph by an edge with weight $1 + \varepsilon$. Suppose that $2 < t < 3$ is fixed; then if ε is suitably small, the greedy t–spanner of G is the the bipartite graph and half of the edges connecting the additional vertex. On the other hand, the optimal t–spanner is the star graph that sits atop of the original bipartite graph.

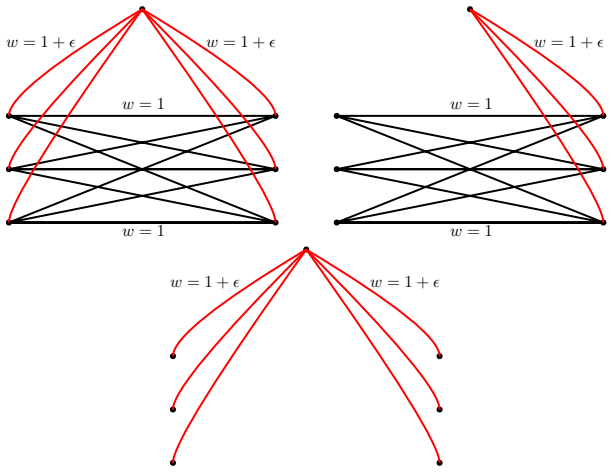

Figure 2: (**Top Left**) Input Graph G, (**Top Right**) The Greedy t–spanner for any $2 < t < 3$, (**Bottom**) the optimal t–spanner for any $2 < t < 3$.

In this case, if $G' = (V, E')$ is the greedy t–spanner, and $G_{\text{OPT}} = (V, E_{\text{OPT}})$ is the optimal t–spanner, we find that

$$|E_{\text{OPT}}| = n, \quad |E'| = n^2 + n,$$

and

$$W(E_{\text{OPT}}) = n(1 + \varepsilon), \quad W(E') = n^2 + n(1 + \varepsilon).$$

In other words,

$$\frac{|E'|}{|E_{\text{OPT}}|} = \Theta(n), \quad \frac{W(E')}{W(E_{\text{OPT}})} = \Theta(n).$$

Thus, greedy t–spanners can be arbitrarily worse in terms of both sparsity and lightness than optimal t–spanners.

4.5. Sparsity and Lightness Guarantees for Greedy t–spanners

[59] Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proceedings of the eighth annual Symposium on Computational Geometry*, pages 192–201. ACM, 1992

[93] Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 442–452. Springer, 2014

[99] Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(3):29, 2016

[63] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):33, 2018

Implicit in the formulation of the t–spanner problem is the idea that a good t–spanner for a graph should contain very few edges while still preserving distances in the manner prescribed. While Algorithm 2 does not require the greedy ordering of the edges to produce a t–spanner, this ordering is crucial to constructing a sparse t–spanner, i.e. one that has few edges and/or small total edge weight. The original bounds given by Althöfer et al. are the following.

Theorem 4.2 ([11, Theorem 1])**.** *Let* $G' = (V, E') = \text{GREEDYSPANNER}(G, 2t+1)$ *where* $t > 0$*. Then* G' *is a* $(2t+1)$*–spanner of* G *and*

1. $|E'| < n\lceil n^{\frac{1}{t}} \rceil$,
2. $W(E') < (1 + \frac{n}{2t})W(\text{MST}(G))$.

However, in modern papers a slightly strengthened version of this theorem is usually quoted:

Theorem 4.3 ([11, Theorem 1, Strengthened])**.** *Let* k *be a positive integer and let* $G' = (V, E') = \text{GREEDYSPANNER}(G, 2k-1)$*. Then* G' *is a* $(2k-1)$*–spanner of* G *and*

1. $|E'| = O\left(n^{1+\frac{1}{k}}\right)$,
2. $W(E') = O(1 + \frac{n}{k})W(\text{MST}(G))$.

Proof of Theorem 4.3, (1). As discussed in Section 4.2 above, the greedy spanner with parameter $2k-1$ has no cycles on $\leq 2k$ edges; that is, it has girth $> 2k$. By the Moore Bounds (Proposition 2.3), it thus has $O\left(n^{1+\frac{1}{k}}\right)$ edges. □

An important observation implicit in the stronger phrasing is that it is without loss of generality to consider only *odd integer* stretch parameters for multiplicative spanners, at least with respect to extremal spanner size. This essentially follows from two graph-theoretic facts. First is that (as discussed in Section 4.2) the extremal sparsity of a t–spanner is the same as the extremal sparsity $\gamma(n, t)$ of a graph with girth $> t$, and girth is an integer parameter. Thus we have (say) $\gamma(n, 5) = \gamma(n, 5.5)$, and so the extremal sparsity of a 5–spanner is the same as the extremal sparsity of a 5.5–spanner, even though the latter is strictly more accurate. Thus the size bound for 5–spanners is a strictly stronger result than the size bound for 5.5–spanners.

The second fact is that $\gamma(n, 2k) = \Theta(\gamma(n, 2k+1))$. This holds for the following reason: given a graph G with girth $> 2k$, one can find a bipartite subgraph H by placing each node on the left or right side of the bipartition with probability $\frac{1}{2}$, and then keeping only edges crossing the divide. One computes that each edge survives in H with probability $\frac{1}{2}$, and thus (in expectation) the density of

H is within a constant fraction of the density of G. On the other hand, all cycles in H are even, so H has girth $> 2k+1$. Together, these facts that the extremal size of (say) a 5–spanner is the same as the extremal size of a 6.99–spanner, up to constant factors. Thus if we hide constant factors, as in Theorem 4.3, the size bounds for odd integer stretch subsumes all other possible stretch values.

The size bound in Theorem 4.3 is tight assuming the Girth Conjecture; however, the weight bound is not. Indeed, Chandra et al. [59] improved the bound to $O_\epsilon(k \cdot n^{\frac{1}{k}})$ for the greedy $(2k-1)(1+\epsilon)$–spanner. Elkin et al. [93] further improved the bound to $O_\epsilon(\frac{k}{\log k} \cdot n^{\frac{1}{k}})$; the construction time of this result was then improved in [99]. Recently, Chechik and Wulff-Nilsen [63] proposed a new spanner construction with lightness $O_\epsilon(n^{\frac{1}{k}})$, completely removing dependency of the factor on k. According to the existential optimality of the greedy spanner as shown by Filtser and Solomon [105], the same sparseness bound holds for the original greedy spanner algorithm. This result is optimal up to a $(1+\epsilon)$ factor in the stretch provided the girth conjecture is true.

4.6. Summary of Greedy Algorithm Guarantees

Stretch (t)	Size: $O(\lvert E' \rvert)$	Weight: $O(W(E'))$	Time	Ref.
$2k-1$	$n^{1+\frac{1}{k}}$	$(1+\frac{n}{2k})W(\text{MST}(G))$	$m(n^{1+\frac{1}{k}}+n\log n)$	[11]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{1+\frac{1}{k}}$	$m(n^{1+\frac{1}{k}}+n\log n)$	[59]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{1+\frac{1}{k}}$	$n^{\frac{1}{k}}\left(1+\frac{k}{\varepsilon^{1+\frac{1}{k}}\log k}\right)$	[93]

Table 1: Spanners given by the Greedy Algorithm. Here $n = |V|$ and $m = |E|$.

4.7. Open Questions

1. Is there a natural reverse greedy algorithm for constructing a t–spanner? That is, one which begins with the full edge set E ordered in nonincreasing order, and deletes edges according to some criteria to arrive at a t–spanner.

2. Is there a greedy algorithm to produce additive spanners, (α, β)–spanners, subsetwise spanners, or more generally pairwise spanners? (The additive spanner construction of [119] has a greedy step; can it be brought completely in line with the multiplicative greedy algorithm, or understood by a similar analysis?)

5. Clustering and Path Buying Methods

Another prominent set of techniques for computing graph spanners is what we call *clustering and path buying* techniques. In the clustering phase, one starts from a set of vertices of a given graph as initial clusters and then expand each to produce a clustering of the graph. Typically at the same time, we add edges within the clusters to the candidate spanner. In the path buying phase, we then

add (buy) cheap edges to achieve the desired spanner. Let us stress here that the clustering we are discussing is much different than common graph clustering algorithms such as k–means or spectral clustering whose aim is to partition the graph into disjoint clusters.

5.1. General Approach

Here we provide a general definition for clustering in the context of spanners.

Definition 5.1. A clustering $\mathscr{C} = \{C_1, \ldots, C_k\}$ of the graph $G = (V, E)$ is a collection of clusters $C_i \subseteq V$.

While this definition is quite general, typically one imposes extra conditions on the clustering based on the specific problem at hand. For instance, one may require $\bigcup_i C_i = V$, or one may require that clusters are pairwise disjoint.

Intuitively, the clustering step aims to attack the spanner problem locally, and the final step makes global adjustments to find the appropriate spanner. As an example of this global adjustment, one can iterate through a given set of vertex pairs (u, v) and make sure there exists a u-v path with the desired stretch factor.

To better illustrate common variants of this set of techniques, we first list some useful general concepts to keep in mind. Variants of each of these will be used in the different algorithms discussed in the sequel.

- **Value of a Path**: The value of the path is the overall improvement of the stretch factors that we gain by adding the path to the spanner under construction. For example, for a given path ρ we can measure the number of clusters that will be closer together after adding ρ to the spanner.
- **Cost of a Path**: In the path buying phase of the algorithm, we may decide to add some edges to the spanner to maintain a given path in G. The cost of a path typically is the weight of the new edge set that we are adding to the spanner.

Most clustering and path buying algorithms run in two stages, which we summarize in the following proto-algorithms. Typical clustering algorithms begin with singleton vertices as clusters, which are then grown according to some rule which we simply call Rule 1, and subsequently edges are added to the spanner based on different criteria (called Rule 2) depending on if they connect vertices within a cluster or not.

Algorithm 3 PROTO-CLUSTERING(G, Rule 1, Rule 2)

Initialize $E' \leftarrow \emptyset$
Choose initial clusters $C_1, \ldots C_k$ (deterministically or randomly)
for $v \in V$ **do**
    Add nearby vertices to a cluster, and add nearby edges to E' according to Rule 1
end for
If an edge is not nearby a cluster, add it to E' according to Rule 2
return E'

To specify a proto-algorithm for the path buying stage, one first chooses a notion of the cost and value of a path which will be denoted $\text{cost}(\rho)$ and $\text{value}(\rho)$ for a given path ρ, respectively. With these as parameters, the path-buying proto-algorithm may be stated as follows (the set of all paths in G will be denoted by $\mathcal{P}$).

Algorithm 4 PROTO-PATH BUYING($G, \text{cost}(\rho), \text{value}(\rho), \alpha$, Rule 1, Rule 2)

```
Initialize E' ←PROTO-CLUSTERING(G, Rule 1, Rule 2)
for ρ ∈ P do
    if cost(ρ) ≤ α · value(ρ) then
        E' ← E' ∪ {e ∈ ρ}
    end if
end for
return E'
```

The proto-algorithms given here have many degrees of flexibility: namely the initialization of clusters, the rules for adding clustered and unclustered edges to E', the definitions of cost and value of a path, and the relation of the final two quantities. As a general rule, the clustering phase is the cheap one in terms of run-time complexity, whereas the path buying phase is more expensive due to the fact that one typically runs through all possible paths in G.

Let us also note that several of the clustering algorithms utilize a random edge selection step; however, the guarantees for the resulting spanners are deterministic. For algorithms which produce spanners only with high probability, see Section 6.

5.2. Illustrating Example

[26] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):5, 2010

Let us begin with one of the simpler algorithms in the vein described above given by Baswana et al. [26] which computes a $(1, 6, V \times V)$–spanner (i.e. an additive 6–spanner) of an unweighted graph. We describe only the terminology and rules required to state their algorithm in terms of the proto-algorithms of the previous section.

Initialization of Clusters: first, $|V|^{\frac{2}{3}}$ cluster centers are chosen uniformly randomly from V. That is, we have $C_1 = \{v_1\}, \ldots, C_k = \{v_k\}$ for $k = |V|^{\frac{2}{3}}$.

Rule 1: if v is adjacent to a cluster center, it joins an arbitrary cluster that it is adjacent to, and the corresponding edge is added to E'.

Rule 2: if v has no adjacent cluster center, then it is left unclustered and all edges incident to v are added to E'.

For the path-buying phase of this algorithm, the parameters involved are defined as follows:

$$\text{cost}(\rho) := |\rho \setminus E'|, \tag{2}$$

that is, the number of edges in ρ that are not already in the spanner, and

$$\begin{aligned}\text{value}(\rho) := |\{(C_1, C_2) \in \mathscr{C} \times \mathscr{C} : \rho \text{ intersects } C_1, C_2 \text{ and} \\ \text{dist}_{G'}(C_1, C_2) \text{ decreases after adding } \rho \text{ to } G'\} \\ \alpha := 2\end{aligned}$$

where G' is the current spanner and $\text{dist}_{G'}(C_1, C_2)$ denotes the length of the shortest path in G' between terminal vertices that lie in C_1, C_2 respectively.

Theorem 5.1 ([26, Theorem 2.7])**.** *Given the cluster initialization, Rule 1, 2, and definitions of α, cost and value of paths above, the algorithm* PROTO-PATH BUYING *returns a $(1, 6, V \times V)$–spanner of G.*

Sketch of Proof. For the sake of illustration, we sketch the proof that the algorithm described in Theorem 5.1 produces a $(1, 6)$–spanner. Consider the special case of two clustered nodes u and v. For a shortest path ρ from u to v, we look at a sequence of clusters $C_1, \ldots C_\ell$ intersecting ρ with $u \in C_1$ and $v \in C_\ell$. We will prove that after the path-buying phase, the spanner G' satisfies the following property.

Intermediate Cluster Property (ICP): A shortest path ρ satisfies the ICP if there exists a cluster C_i $(1 < i < \ell)$ along ρ such that

$$\text{dist}_{G'}(C_1, C_i) \leq \text{dist}_\rho(C_1, C_i), \quad \text{dist}_{G'}(C_i, C_\ell) \leq \text{dist}_\rho(C_i, C_\ell).$$

Here, $\text{dist}_\rho(C_i, C_j)$ is the distance from C_i to C_j along the prescribed path ρ. Note that in the description of Rule 2, v joins only one of the neighbor clusters and hence there are two types of missing edges in ρ: those with both endpoints in the same cluster (intracluster) and those with endpoints belonging to different clusters (intercluster). Now, assuming that the ICP is true for some shortest path ρ and counting intracluster and intercluster missing edges, we can apply the triangle inequality and use the fact that the diameter of each cluster is two, to get the desired $(1, 6)$–spanner condition.

Now, our goal is to prove the ICP; to do so, we first define the following sets:

$$A = \{\{C_i, C_j\} : i = 1 \, or \, j = \ell\}$$

$$A_0 = \{\{C_i, C_j\} \in A : \text{dist}_{G'}(C_i, C_j) > \text{dist}_\rho(C_i, C_j)\},$$

$$A_1 = A \setminus A_0.$$

Recall the definition for the value of a path:

$$\text{value}(\rho) = |\{(C_i, C_j) : \text{dist}_{G'}(C_i, C_j) > \text{dist}_\rho(C_i, C_j)\}|,$$

hence we have $|A_0| \leq \text{value}(\rho)$.

Claim: $\text{value}(\rho) < \ell - 2$.

If the claim is true then $|A_1| > \ell - 1$, and using the pigeonhole principle there exists some C_i satisfying the ICP. It remains to prove the claim. With a simple counting argument, and if shortest path ties are broken properly (roughly,

shortest paths should stay in their current cluster as long as possible), given that ℓ is the number of clusters intersecting ρ, the number of missing intercluster edges in G' is at most $\ell - 1$ and the number of missing intracluster edges is at most $\ell - 2$. Therefore, $\text{cost}(\rho) < 2\ell - 3$ and using the fact that $2\text{value}(\rho) < \text{cost}(\rho)$, we get $\text{value}(\rho) < \ell - 2$ which finishes the proof. □

5.3. Theme and Variations

[28] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a $(2k-1)-$spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Automata, Languages and Programming*, pages 384–396, 2003

[139] Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6(1):7, 2009

[73] Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On Pairwise Spanners. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20, pages 209–220, 2013

[117] Telikepalli Kavitha and Nithin M Varma. Small stretch pairwise spanners. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 601–612. Springer, 2013

[116] Telikepalli Kavitha. New pairwise spanners. *Theory of Computing Systems*, 61(4):1011–1036, 2017

Cygan et al. [73] give a polynomial time clustering and path buying algorithm for computing (α, β) pairwise, subsetwise, and sourcewise spanners. Their clustering step is essentially a deterministic version of the one used by Baswana et al. [26]; this derandomization is orthogonal to the change from all-pairs to pairwise spanners, but we will include it here for completeness.

Clustering Phase

This follows Algorithm 3 with the following rules. In the sequel, let d be an integer parameter of the construction that we will choose later.

Choice of Clusters: Unmark all nodes, and then while there is a (possibly marked) node $v \in V$ with $\geq d$ unmarked neighbors, choose a set C of exactly d of its neighbors and add C as a new cluster. The node v is called its *center* (note that $v \notin C$). We then mark all nodes in C, and repeat until we can do so no longer.

Rule 1: All edges with both endpoints contained in the same cluster are included in the spanner. All edges between a cluster center and a node in its cluster are included in the spanner.

Rule 2: For each node v not contained in a cluster, include all of its incident edges in the spanner.

The essential properties of this clustering step are as follows.

Lemma 5.1 ([73])**.** *Algorithm 3, with the above parameters, provides a clustering $\mathscr{C}$ of G and a subgraph $G' = (V, E')$ with the following properties:*

1. *The size of each cluster $C \in \mathscr{C}$ is exactly $|C| = d$.*
2. *The total number of clusters is $|\mathscr{C}| \leq n/d$.*

3. *Any two nodes in the same cluster have a common neighbor in G' and hence the diameter of any cluster is at most 2.*

4. $|E'| = O(nd)$

5. *If an edge uv is absent in E', then u and v belong to two different clusters (In particular, they cannot be unclustered nodes).*

Proof. Items (1), (3) and (5) follow directly from the algorithm. (2) follows directly from (1) since clusters are node-disjoint. Finally, we argue (4) as follows. In Rule 1, we add at most $\binom{d}{2}$ edges per cluster, which is $O(nd)$ edges in total. For each edge uv added in Rule 2, we note that one (or both) endpoints are unmarked. Since every node has $< d$ unmarked neighbors, by a union bound over the n nodes we add $O(nd)$ edges in this step as well. □

Path Buying Phase

Cygan et al. give several variants of the path buying phase to obtain different types of spanners. All begin by using the above Clustering algorithm, but vary the definition of value of paths and α as in the Proto-path buying algorithm. The first variant is to define cost as in (2), and value as

$$\text{value}(\rho) := |\{(x, C) \in S \times \mathscr{C} : \rho \text{ intersects } x, C, \text{ and } \text{dist}_{G'}(x, C) \text{ decreases after adding } \rho \text{ to } G'\}|. \quad (3)$$

With these definitions of cost and value of a path, the clustering + path-buying algorithm yields an additive subsetwise spanner with additive stretch 2 by setting $\alpha = 2$.

Theorem 5.2 ([73, Theorem 1.3]; also [83, 139])**.** *For any $S \subseteq V$, given the above cluster initialization with parameter $d = \sqrt{|S|}$, cost and value of paths specified by* (2)*,* (3)*, and $\alpha = 2$, the algorithm* PROTO-PATH BUYING *returns a $(1, 2, S \times S)$–spanner of G of size $O(n\sqrt{|S|})$.*

We remark that any choice of d in the CLUSTER step will produce a $(1, 2, S \times S)$–spanner, but the choice of parameter $d = \sqrt{|S|}$ is needed to minimize the size of the final spanner. In particular, the clustering phase costs $O(nd)$ edges and the path buying step costs $O(|S|n/d)$ edges, and these balance at the choice $d = \sqrt{|S|}$.

The second variant of the path-buying algorithm uses path-buying as a *preprocess* in a larger spanner construction. This is more complicated to prove, but the statement is as follows:

Theorem 5.3 ([73, Theorem 1.1])**.** *For any $\varepsilon > 0$, positive integer k, and any $P \subseteq V \times V$, given the above cluster initialization with parameter*

$$d = n^{\frac{1}{k}}(2k + 3)|S|^{\frac{k}{2k+1}},$$

cost and value of a path specified in Section 5.2, and

$$\alpha = \frac{12 \log n}{\varepsilon},$$

let the edge set of the output of the algorithm PROTO-PATH BUYING *be* E_1. *Let* E_2 *be the edges of a* $(2\log n, 0, V \times V)$*–spanner of* G *having size* $|E_2| = O(n)$ *[111]. Then* $G' = (V, E_1 \cup E_2)$ *is a* $(1+\varepsilon, 4, P)$*–spanner of* G.

The final variant involves the same clustering step, but a more complicated path-buying phase. For full details, see [73]; details of their other results are in the tables in the Appendix.

Baswana et al. [26] also give another variant of their algorithm which computes a $(k, k-1, V \times V)$–spanner. Rather than force it into our proto-algorithm framework, we reproduce it in full here in Algorithm 5. The clustering used here is related to one in [150].

Algorithm 5 Randomized $(k, k-1)$–Spanner Construction(G)

Initialize $E' \leftarrow \emptyset$ and $\mathscr{C}_0 \leftarrow \{\{v\} : v \in V\}$
for i from 1 to k **do**
  Let $\mathscr{C}_i$ be sampled from $\mathscr{C}_{i-1}$ with probability $n^{-\frac{1}{k}}$ (If $i = k$, then $\mathscr{C}_k = \emptyset$)
  for v not belonging to a cluster in $\mathscr{C}_i$ **do**
    If v is adjacent to some $C \in \mathscr{C}_i$, add v to C and add some edge of $E(v, C)$ to E'
    Otherwise, add to E' some edge from $E(v, C)$, for each $C \in \mathscr{C}_{i-1}$ adjacent to v
  end for
end for
Add to E' one edge from $E(C, C')$ for each adjacent pair $C \in \mathscr{C}_i$ and $C' \in \mathscr{C}_{k-1-i}$ for i from 0 to $k-1$
Add to E' one edge from $E(C, C')$ for each adjacent pair $C \in \mathscr{C}_i$ and $C' \in \mathscr{C}_{i-1}$ for i from $\lceil \frac{k}{2} \rceil$ to $k-1$
return E'

Kavitha and Varma [117] give yet another clustering and path buying algorithm for computing (α, β, P)–spanners which utilizes *breadth-first search (BFS) trees* [72] in the path-buying phase. Recently, Kavitha [116] presents a modified version of the BFS strategy to compute additive pairwise spanners. The broad idea of both is that BFS trees provide highways along which distances are preserved, and they are used to connect clustered nodes together.

Kavitha [116] has provided algorithms similar in spirit to those of Cygan et al. to compute additive pairwise spanners.

Baswana and Sen [28] use a similar clustering algorithm to Algorithm 5 but a different path-buying phase than [26] to yield a $(2k-1)$–spanner with $O(n^{1+\frac{1}{k}})$ edges in expected linear time $O(kn)$. This gives a spanner with optimal number of edges (up to a constant and assuming the Girth Conjecture) in optimal time.

Pettie [139] proposes a *modular* scheme for constructing (α, β)–spanners through utilizing *connection schemes*, many of which are variants of this clustering + path buying approach. Through assembling connection schemes properly, most existing results can be generated. Also, substantially improved results for almost additive spanners are obtained. In particular, it is shown that linear

size ($|E'| = O(n)$) spanners can be constructed with good stretch including $(5+\epsilon, \text{polylog}(n))$–spanners, and $(1, \widetilde{O}(n^{\frac{9}{16}}))$–spanners. The latter bound was later improved to $(1, O(n^{\frac{3}{7}+\varepsilon}))$ in [45].

5.4. Further Reading

[95] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$–spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004

[25] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)–spanners and purely additive spanners. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005

[30] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007

[61] Shiri Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512. Society for Industrial and Applied Mathematics, 2013

Baswana et al. [25] give two important results on additive spanners: an additive 6–spanner of size $O(n^{\frac{4}{3}})$, and a linear time construction of $(k, k-1)$–spanners with size $O(n^{1+\frac{1}{k}})$. In Baswana and Sen [30], the first linear time randomized algorithm that computes a t–spanner of a given weighted graph was given. Recall that this size/error tradeoff is optimal assuming the Girth Conjecture (see Section 2.4).

Chechik [61] gives a construction for an additive 4–spanner containing $\widetilde{O}(n^{\frac{7}{5}})$ edges. In addition, a construction is given for additive spanners with $\widetilde{O}(n^{1+\delta})$ edges and additive stretch $\widetilde{O}(n^{\frac{1}{2}-\frac{3\delta}{2}})$ for any $\frac{3}{17} \leq \delta < \frac{1}{3}$, improving the stretch of the existing constructions from $O(n^{1-3\delta})$ to $\widetilde{O}(\sqrt{n^{1-3\delta}})$. Finally, it is shown that this construction can be modified to give a sublinear additive spanner of size $\widetilde{O}(n^{1+\frac{3}{17}})$ with additive stretch $O(\sqrt{d(u,v)})$ for each pair $u, v \in V$.

Elkin and Peleg [95] show that the multiplicative factor can be made arbitrarily close to 1 while keeping the spanner size arbitrarily close to $O(n)$ at the cost of allowing the additive term to be a sufficiently large constant. In other words, they show that for any constant $\lambda > 0$ there exists a constant $\beta = \beta(\epsilon, \lambda)$ such that for every n–vertex graph there is an efficiently constructible $(1+\epsilon, \beta)$–spanner of size $O(n^{1+\lambda})$.

5.5. Open Problems

1. Suppose one runs a traditional graph clustering algorithm (in the unsupervised learning sense; e.g. Normalized Cuts, Spectral Clustering, or Hierarchical Clustering) which produces clusters which are disjoint and cover the whole graph. Can one use this clustering to give rise to a good spanner of G?

6. Probabilistic Methods

[128] Gary L Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 192–201. ACM, 2015

[90] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Transactions on Algorithms (TALG)*, 15(1):1–29, 2018

Miller et al. [128] construct the first randomized algorithm for producing a spanner with high probability (other algorithms using randomization give deterministic guarantees). The main result therein (Theorem 1.1) yields a multiplicative $O(k)$–spanner with high probability of expected size $O(n^{1+\frac{1}{k}} \log k))$ in expected time $O(m)$.

Elkin and Neiman [90] improved on the work of Miller et al. [128] by giving a randomized construction which computes a $(2k-1)$–spanner with $O(n^{1+\frac{1}{k}} \varepsilon^{-1})$ edges with probability $1-\varepsilon$. Moreover, they give a runtime and edge count analysis for both PRAM and CONGEST computational models. To illustrate this recent technique, we give a partial proof of the main theorem in [90].

6.1. Probabilistic $(2k-1)$–spanners

To begin, note that given a parameter $\lambda > 0$, the *exponential distribution* is defined by the probability density function (pdf)

$$p_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0. \end{cases}$$

We denote a random vector x drawn from this distribution by $x \sim Exp(\lambda)$.

Algorithm 6 RAND EXP $(2k-1)$–SPANNER(G, k, c)

$\lambda \leftarrow \frac{\ln(cn)}{k}$
for $u \in V$ **do**
  Draw $r_u \sim Exp(\lambda)$
  u broadcasts r_u to all vertices v within distance k
  if x receives a message from u **then**
    x stores the value $m_u(x) = r_u - d_G(u, x)$
    x also stores a neighboring vertex $p_u(x)$ along a shortest path from u to x (if there is more than one possibility, one is selected randomly)
  end if
end for
$m(x) \leftarrow \max_{u \in V} m_u(x)$
for $x \in V$ **do**
  $E' \leftarrow E' \cup C(x) = \{(x, p_u(x)) : m_u(x) \geq m(x) - 1\}$
end for
return E'

The key reason for using the exponential distribution to determine the messages passed from each vertex is its *memoryless property*, which is the fact that

if $x \sim Exp(\lambda)$ and $s, t \in \mathbb{R}$, then $\mathbb{P}(x > s + t \mid x > t) = \mathbb{P}(x > s)$, i.e. does not depend on the value of t. The main result of [90] is the following.

Theorem 6.1 ([90, Theorem 1])**.** *Let $G = (V, E)$, $k \in \mathbb{N}$, $c > 3$, and $\delta > 0$ be fixed. Then with probability at least $(1 - \frac{1}{c})\frac{\delta}{1+\delta}$, Algorithm 6 computes a $(2k-1)$–spanner of G which has at most*

$$(1+\delta)\frac{(cn)^{1+\frac{1}{k}}}{c-1} - \delta(n-1)$$

edges.

Sketch of Proof. Choose $\lambda = \frac{\ln cn}{k}$, and let X be the event $\{\forall u \in V, r_u < k\}$. Note that $\mathbb{P}(r_u \geq k) = e^{-\lambda k} = \frac{1}{cn}$; hence the union bound implies that $\mathbb{P}(\exists u \in V \text{ s.t. } r_u \geq k) \leq \frac{1}{c}$. Hence $\mathbb{P}(X) \geq 1 - \frac{1}{c}$.

The key observation made in [90] is that if the event X holds, then the subgraph $G' = (V, E')$ output by Algorithm 6 is a spanner with stretch at most $2k - 1$. Additionally, since G' is a spanner, it must have at least $n - 1$ edges, and so if Y is the random variable $|E'| - (n-1)$, then Y is positive. Moreover, $\mathbb{E}(Y) \leq n(cn)^{\frac{1}{k}} - (n-1)$ (Lemmas 1 and 2 of [90]), and so

$$\mathbb{E}(Y \mid X) \leq \frac{\mathbb{E}(Y)}{\mathbb{P}(X)} \leq \frac{1}{1-\frac{1}{c}}\left(n(cn)^{\frac{1}{k}} - (n-1)\right).$$

To turn this into a concrete bound, note that Markov's inequality implies that for the given δ,

$$\mathbb{P}(Y \geq (1+\delta)\mathbb{E}(Y \mid X) \mid X) \leq \frac{1}{1+\delta},$$

whereby we have

$$\mathbb{P}((Y < (1+\delta)\mathbb{E}(Y \mid X)) \cap X) \geq \left(1 - \frac{1}{c}\right)\frac{\delta}{1+\delta}.$$

Consequently, if these events hold, then $|E'| = Y + n - 1$ is at most $(1 + \delta)\mathbb{E}(Y \mid X)$, which is at most the quantity in the statement of the theorem by combining the above estimates. □

6.2. Open Problems

1. Can one optimize the construction of Elkin and Neiman over the pdf to get a better guarantee for the size of a $(2k-1)$–spanner?

2. Can one design a probability distribution over the edges of a graph from which random sampling yields a spanner with high probability?

7. Subsetwise, Sourcewise, and Pairwise Spanners

7.1. Pairwise Distance Preservers

[46] Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Mathematics*, 19(4):1029–1055, 2005

[71] Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006

[45] 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 (SODA)*, pages 855–872. Society for Industrial and Applied Mathematics, 2016

[1] Amir Abboud and Greg Bodwin. Error amplification for pairwise spanner lower bounds. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 841–854. Society for Industrial and Applied Mathematics, 2016

[38] Greg Bodwin. Linear size distance preservers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 600–615. Society for Industrial and Applied Mathematics, 2017

[106] Kshitij Gajjar and Jaikumar Radhakrishnan. Distance-preserving subgraphs of interval graphs. In *25th Annual European Symposium on Algorithms (ESA)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2017

[39] Greg Bodwin. On the structure of unique shortest paths in graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2071–2089. Society for Industrial and Applied Mathematics, 2019

A special case of the spanner problem is that of finding *distance preservers*, which requires that distances are preserved exactly between specified pairs of vertices of G. This variant of the problem is important given that some spanner constructions use distance preservers as a subroutine. Distance preservers were first introduced and studied in [46] and [71]. Coppersmith and Elkin [71] proved the existence of a linear size pairwise distance preserver for any set P of vertex pairs $|P| = O(\sqrt{n})$. Further work has been done in [38, 45, 106].

Recall we may consider subsetwise distance preservers when $P = S \times S$ and sourcewise distance preservers when $P = S \times V$. A fundamental question about distance preservers is the following:

Problem 3. For a fixed n and $P \subseteq V \times V$, what can be said about the number of edges, $|E'|$, in a distance preserver of G over P?

For the moment, consider the simplest case where P has the form $\{s\} \times V$. Here, the distance preserver is a tree with $O(n)$ edges; this fact is known as the Shortest Path Tree Lemma. Note that without further assumptions on the structure of G or P, one cannot improve this construction in general. Indeed, if G is a single path from u to v and $P = \{(u, v)\}$, the preserver must have $n - 1$ edges. On the other hand, if G is a clique, there must be $|P|$ edges in the preserver, and hence in general, $|E'| = \Omega(n + |P|)$.

Here, we present two theorems on the existence and size of distance preservers along with an outline of one of the proofs. The original result of [71] is the following.

Theorem 7.1 ([71])**.** *Given an undirected (weighted or unweighted) graph G and a set of vertex pairs P, there exists a distance preserver of size $|E'| = O(n + n^{\frac{1}{2}}|P|)$.*

The next theorem is a slightly worse in size complexity, but is more general in that it holds for a larger class of graphs.

Theorem 7.2 ([38])**.** *Given a (directed or undirected, weighted or unweighted) graph G and a set of node pairs P, there exists a distance preserver of size $|E'| = O(n + n^{\frac{2}{3}}|P|)$.*

7.2. Sketch of Proof of Theorem 7.2

Definition 7.1 (Tiebreaking Scheme)**.** A *tiebreaking scheme* is a map π that sends each vertex pair (u, v) to a shortest path from u to v.

Definition 7.2. A *consistent tiebreaking* is a tiebreaking scheme π such that for every $w, x, y, z \in V$, if $x, y \in \pi(w, z)$ then $\pi(x, y)$ is a subpath of $\pi(w, z)$.

Lemma 7.1. *Every graph has a consistent tiebreaking.*

Proof. Add a small random number to each edge so that we have a unique shortest path between any pair (u, v); one can see that the output is a consistent tiebreaking. □

Definition 7.3 (Branching Triple)**.** A *branching triple* is a set of three distinct directed edges (u_1, v), (u_1, v), (u_1, v) in E that all enter the same vertex.

Branching triples are based on the closely related notion of *branching events*, used in the proof of Theorem 7.1. We will prove a bound for the edge size of the distance preserver using a bound for the number of branching triples.

Lemma 7.2. *Let $H = (V, \pi(P))$ where π is any consistent tiebreaking. Then H has at most $\binom{|P|}{3}$ branching triples.*

Proof. Assign each edge e to some pair $p = (v, v') \in P$ such that $e \in \pi(p)$ (i.e. edge e belongs to the shortest path from v to v'). We show that any three pairs in P cannot share two or more branching triples. By way of contradiction, if three pairs p_1, p_2, p_3 share two branching triples (e_1, e_2, e_3) and (e'_1, e'_2, e'_3), then a situation similar to the following will always happen: the edge e_1 precedes e'_1 in p_1 and e_2 precedes e'_2 in p_2. Since π is a consistent tiebreaking, one can check that $e'_1 = e'_2$ which is a contradiction. That means for every three pairs in P, there is at most one corresponding branching triple. Consequently the number of branching triples is at most $\binom{|P|}{3}$. □

The first two edges incident to any vertex v will not create a branching triple; after that, each new edge will contribute at least one new branching triple, so a graph with $O(n)$ branching triples has $O(n)$ edges. From the previous lemma, we conclude that if $|P| = O(n^{\frac{1}{3}})$ then $|E(H)| = O(n)$. Now, given a graph G and a set of vertex pairs P, we can partition P into subsets of size $O(n^{\frac{1}{3}})$ and apply this fact, which gives the desired conclusion.

7.3. Further Reading

Bodwin and Williams [45] proved an upper bound of $O(n^{\frac{2}{3}}|P|^{\frac{2}{3}} + n|P|^{\frac{1}{3}})$ for undirected and unweighted graphs using a new type of tiebreaking scheme. They also rely on a clustering technique to prove new upper bounds for additive spanners.

Abboud and Bodwin [1] consider lower bounds for pairwise spanners. Most importantly, they prove that lower bounds for pairwise distance preservers imply lower bounds for pairwise spanners. These connections were significantly expanded in [2].

The existing trivial lower bounds show that in the worst case, the size of a distance preserver is at least linear in n and p. Bodwin [38] makes some progress on identifying the cases that these bounds are tight, i.e., when a linear size ($|E'| = O(n + p)$) distance preserver is guaranteed. For example, by the result discussed above, if the number of pairs is $O(n^{\frac{1}{3}})$ then one can always find a distance preserver with $O(n)$ edges.

Abboud and Bodwin [3] introduce and study *reachability preservers*, in which only reachability (typically in a directed graph) rather than distances between demand pairs must be preserved.

7.4. Multiplicative Pairwise Spanners

[86] Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theoretical Computer Science*, 697:1–36, 2017

The paper [86] consists of various embedding techniques of a metric space (or graph) into a normed space (or a family of graphs), bounding metric distortion for connecting nodes to terminals. As an example of the generic results in the paper, if one can embed any m–node graph into ℓ_p with distortion $\alpha(m)$ using $\gamma(m)$ dimensions, then there is an algorithm for embedding any graph with k terminals into ℓ_p, with distortion $2\alpha(k) + 1$, using $\gamma(k) + n - 1$ dimensions.

A *terminal graph spanner* is another name for a sourcewise spanner; that is, given a graph $G = (V, E)$ and a set of "terminal nodes" $K \subseteq V$, the spanner property must hold for all pairs in $K \times V$. This name is more common when dealing with metric embeddings rather than general graphs. Another result of [86] is the following construction of a $(4t - 1)$–terminal spanner with $O(n + \sqrt{n}|K|^{1+\frac{1}{t}})$ edges, for a graph G with terminal set $K \subseteq G$:

- Create a $(2t - 1)$ metric spanner H' of G, with $P = \mathcal{E}(H')$ and $|P| < O(|K|^{1+\frac{1}{t}})$
- Applying a distance preserver algorithm to P, obtain $G' \subseteq G$ with $O(n + \sqrt{n}|P|)$ edges, preserving distances in K
- Create $H \supset G'$ by adding shortest path tree in G with K as root.

One reason that pairwise spanners with multiplicative error are perhaps less well studied than others types of error is that they are nontrivial only in a restricted range of parameters. Suppose we have $|P| = n^{1+c}$ demand pairs for some $c > 0$, and we want a multiplicative t–spanner. If c is not too large relative to t, then at least $|P|$ edges may be needed in the spanner. On the

other hand, $O(|P|)$ edges always suffice by the following construction (which is folklore). Preprocess the graph with the all-pairs mixed error spanner of [95], with parameters chosen such that the spanner has only $O(|P|)$ edges. One can then argue that all pairs in P have been well spanned, except maybe for pairs at constant (depending only on c, t) distance in the original graph. Hence, we may add an exact shortest path for all remaining pairs, and these cost only $O(|P|)$ edges in total.

7.5. Open Problems

1. What is the largest $p = p(n)$ so that, for any $|P| = p$ node pairs in an undirected unweighted n–node graph, there is a pairwise spanner of P on $O(n)$ edges with $+c$ additive error? In particular, it is known that any $|P| = O(n^{\frac{1}{2}})$ pairs have a distance preserver ($+0$ error) on $O(n)$ edges, but it is not clear if one can handle more pairs with (say) a $+2$ error tolerance.

8. Extremal Bounds for Additive, Mixed, and Sublinear Spanners

8.1. Purely Additive Spanners

[10] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28:1167–1181, 04 1999

[153] David P Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 389–398. IEEE, 2006

[26] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):5, 2010

[61] Shiri Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512. Society for Industrial and Applied Mathematics, 2013

[119] Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. In *Scandinavian Workshop on Algorithm Theory*, pages 277–281. Springer, 2014

[2] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *Journal of the ACM (JACM)*, 64(4):28, 2017

[113] Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. In *16th Scandinavian Symposium and Workshops on Algorithm Theory*, 2018

There are three classic upper bounds results for spanners of unweighted graphs with purely additive error:

1. Implicit in Aingworth et al. [10] is the fact that all graphs have additive $+2$–spanners with $\widetilde{O}(n^{\frac{3}{2}})$ edges; it has been subsequently observed (in folklore) that the bound can be improved to $O(n^{\frac{3}{2}})$. A qualitatively different construction of the same result follows from [26, 73, 83, 119].

2. Chechik [61] proved that all graphs have additive $+4$–spanners with $\widetilde{O}(n^{\frac{7}{5}})$ edges. The size was recently improved slightly to $O(n^{\frac{7}{5}})$ by Bodwin [40].

3. Baswana et al. [26] proved that all graphs have additive +6–spanners with $O(n^{\frac{4}{3}})$ edges. A simple reframing of the construction was given by Knudsen [119], and a qualitatively different construction with slightly worse size $\widetilde{O}(n^{\frac{4}{3}})$ but faster construction time $\widetilde{O}(n^2)$ was given by Woodruff [154]. Knudsen [120] later improved the size to $O(n^{\frac{4}{3}})$ and the construction time to $O(n^2)$, at the cost of increased error of +8.

These upper bounds on the size of a spanner versus additive error begin to exhibit a tradeoff curve, where one can construct a sparser additive spanner if more additive error is allowed. One natural question is: can we construct even sparser additive spanners while only incurring a small amount of additive error? Specifically, given $0 < \varepsilon < \frac{1}{3}$, is there a constant k_ε such that every graph G has an additive $+k_\varepsilon$–spanner containing $O(n^{1+\varepsilon})$ edges? Following some progress in [1, 153], Abboud and Bodwin [2] give a negative resolution to this question; they show that for $0 < \varepsilon < \frac{1}{3}$, one cannot even compress an input graph into $O(n^{1+\varepsilon})$ bits, so that one can recover distance information for each pair of vertices within $n^{o(1)}$ additive error. This implies that one cannot construct an additive spanner using $O(n^{1+\varepsilon})$ edges in this parameter regime, unless poly(n) additive error is allowed. This construction was subsequently optimized by Huang and Pettie [113]. In particular, they proved that an additive spanner on $O(n)$ edges must have $+\Omega(n^{\frac{1}{13}})$ error, improving over a lower bound of $+\Omega(n^{\frac{1}{22}})$ from [1].

The reason that all constructions of additive spanners have even error is because there is a reduction to the setting of bipartite input graphs, where error must come in increments of 2. The reduction is as follows: given an n–node graph $G = (V, E)$, make a graph G' in the following two steps:

1. Make two identical copies of the nodes in V, called V_L, V_R.
2. For each edge $(u, v) \in V$, add two $(u_L, v_R), (u_R, v_L)$ to G', where u_L, v_L, u_R, v_R are the copies of u, v in V_L, V_R, respectively.

Since G' is bipartite, any additive spanner of G' will have only an even amount of error; that is, a $+(2c + 1)$–spanner on G' is necessarily also a $+2c$–spanner. Additionally, a $+2c$–spanner of G' can be converted to a $(+2c)$–spanner of G on at most the same number of edges, by merging together v_L, v_R in the natural way. To state the consequences of this reduction another way, if one can prove that every n–node graph has (say) a +5 additive spanner on $\leq e^*(n)$ edges, then in fact every graph has a +4 additive spanner on $O(e^*(n))$ edges. Thus, generally, it is not interesting to consider purely-additive spanners with odd error. Note however that this reduction only applies to *extremal* problems where the goal is to determine the worst-case size/error tradeoff. For *algorithmic* problems, where the goal is just to find a sparse additive spanner of a particular input graph, the cases of $+2k$ and $+2k+1$ error remain non-equivalent. Indeed, in [66], the authors prove that the algorithmic problem of approximating the sparsest +1–spanner of an input graph is already hard (whereas a +0–spanner is trivial).

8.2. Mixed and Sublinear Spanners

[4] Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM Journal on Computing*, 47(6):2203–2236, 2018

[95] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$–spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004

In light of the previously-mentioned lower bounds of [2], additive spanners cannot generally achieve near-linear size $O(n^{1+\varepsilon})$. Thus, it is natural to ask how one *can* achieve spanners of this quality, necessarily paying more than purely-additive error, but hopefully less than purely-multiplicative error. Indeed, this is possible, and it is studied in two closely related paradigms. These are:

1. *Mixed spanners*, which avoid the above lower bounds by allowing a small $(1+\varepsilon)$ multiplicative stretch in addition to $+\beta$ additive stretch. That is, a $(1+\varepsilon, \beta)$ mixed spanner H of an input graph G satisfies

$$d_H(u,v) \leq (1+\varepsilon) d_G(u,v) + \beta$$

   for all node pairs u, v. The main construction of mixed spanners was given in [95]; see also [4, 31] for followup work.

2. *Sublinear spanners*, which have additive error that depends sublinearly on the distance in question. That is, an $f(d)$–sublinear spanner H of an input graph G satisfies

$$d_H(u,v) \leq d_G(u,v) + f(d_G(u,v))$$

   for all node pairs u, v. So, $f(d) = O(1)$ (independent of d) corresponds to the case of purely additive spanners. The main construction of sublinear spanners was given in [150], as well as sublinear *emulators*, which satisfy the same distance requirement but are allowed to have new long-range edges (u, v) of weight $d_G(u, v)$ (even though the input graph G is still unweighted). (See Section 11.1 for more on emulators.) Follow-up work on sublinear spanners can be found in [61, 139].

The upper bounds for mixed spanners from [95] have a hierarchical nature: for any fixed positive integer k and any $\varepsilon > 0$, they can construct $(1+\varepsilon, \beta)$ mixed spanners of size

$$O(\text{poly}(1/\varepsilon) n^{1+\frac{1}{2^{k+1}-1}})$$

with $\beta = O(1/\varepsilon)^{k-1}$. The Thorup–Zwick *emulators* are similarly hierarchical: for any fixed positive integer k, they have size

$$O(n^{1+\frac{1}{2^{k+1}-1}}),$$

and sublinear error function $f(d) = O(d^{1-\frac{1}{k}})$. This sublinear error function is slightly stronger than $(1+\varepsilon, O(1/\varepsilon)^{k-1})$ mixed error; in fact, it can be viewed as obtaining mixed error of the above type for every possible choice of ε simultaneously. These emulators can also be converted to $(1+\varepsilon, \beta)$ spanners of the

same quality as the above, by replacing each edge of weight $\leq \beta$ with a path through the original graph of length $\leq \beta$ (folklore). It is a major open question in the area to determine if one can build sublinear spanners of the same quality as the Thorup–Zwick emulators without taking on this extra poly$(1/\varepsilon)$ factor.

On the lower bounds side, it is proved in [4] that the Thorup–Zwick emulator hierarchy is essentially optimal (even for all data structures): for any fixed k, one cannot improve the error function to $f(d) = o(d^{1-\frac{1}{k}})$, *and* one cannot cannot polynomially improve the size (e.g. to $O(n^{1+\frac{1}{2^{k+1}-1}-0.001})$). Thus Elkin and Peleg's mixed spanners are similarly optimal, at least up to their dependence on poly$(1/\varepsilon)$.

To highlight the surprising hierarchical nature of distance compression revealed by these results, *a priori* it seems reasonable to ask: what kind of spanner size can be achieved with sublinear error function $f(d) = O(d^{\frac{1}{3}})$? The lower bounds of [4] say that one needs $\Omega(n^{\frac{4}{3}-o(1)})$ edges – but if one is willing to pay $\Omega(n^{\frac{4}{3}})$ edges, then one can even have $f(d) = O(1)$. Thus not all sublinear error functions are equally interesting to consider; rather, Thorup–Zwick type sublinear error of the form $f(d) = O(d^{1-\frac{1}{k}})$ is of particular interest. A similar story holds for the hierarchical tradeoff between ε and β for mixed spanners.

8.3. Additive and Mixed Spanners of Weighted Graphs

[88] Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and pram distance oracles in weighted graphs. *arXiv preprint arXiv:1907.11422*, 2019

[6] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen Kobourov, and Richard Spence. Weighted additive spanners. *arXiv preprint arXiv:2002.07152*, 2020

[114] Shang-En Huang and Seth Pettie. Thorup–Zwick emulators are universally optimal hopsets. *Information Processing Letters*, 142:9–13, 2019

[91] Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in rnc. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 333–341, 2019

The above results all concern spanners for *unweighted* graphs. One can also consider spanners for weighted input graphs. It will not be possible to have a nontrivial construction of (say) an additive spanner with $+2$ error for any weighted input graph, since one can always scale up the edge weights until no edge can be removed without introducing > 2 additive error between its endpoints. Thus it is more natural to look for (say) $+2W$ additive spanners, where W is the maximum edge weight of the input graph. Purely additive spanners of this type were provided in [6], and mixed spanners whose additive part scales with W were given in [88]. In fact, the latter construction has a stronger property: for each node pair u, v, the additive error scales with $W = W(u, v)$ the maximum edge weight along the true $u \rightsquigarrow v$ shortest path in the input graph, rather than the maximum edge weight globally.

The bounds for mixed and sublinear spanners/emulators are closely related to the bounds for *hopsets*, a related graph-theoretic object where the goal is to add some long-range weighted edges to a (possibly weighted) graph so that an approximate shortest path between any two nodes can be realized in a small

number of hops. Formal connections between these objects were introduced in [91, 114], and expanded in a recent survey by Elkin and Neiman [92].

8.4. Open Problems

1. Do all undirected unweighted graphs have an additive spanner with $+4$ error on $O(n^{\frac{4}{3}})$ edges? Currently, the best known bound for $+4$ additive error is $O(n^{\frac{7}{5}})$ edges [40, 61]; any progress beyond this would be interesting.

2. Can Thorup and Zwick's construction of sublinear emulators [150] be converted to spanners with similar size/stretch tradeoffs?

3. How much error is needed for an additive spanner on $O(n)$ edges? Currently there is a construction of additive spanners on $O_\varepsilon(n)$ edges with $+O(n^{\frac{3}{7}+\varepsilon})$ error [45], and the lower bound is $+\Omega(n^{\frac{1}{13}})$ from [113].

4. Can the $+6$ additive spanner on $O(n^{\frac{4}{3}})$ edges from [26] be converted to a $+6W$ spanner of the same size for weighted input graphs with max weight W? (In [6, 88], the corresponding question is resolved affirmatively for the $+2$ and $+4$ additive spanners.

9. Linear Programming Formulations

[144] Mikkel Sigurd and Martin Zachariasen. Construction of minimum-weight spanners. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, pages 797–808, 2004

[77] Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2011

[67] Eden Chlamtáč, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. *In 16th Annual Symposium on Foundations of Computer Science*, 05 2012

[33] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed Steiner forest. *38th International Colloquium on Automata, Languages and Programming (ICALP)*, 222:93 – 107, 2013

[65] Eden Chlamtáč and Michael Dinitz. Lowest-degree k-spanner: Approximation and hardness. *Theory of Computing*, 12(15):1–29, 2016

[80] Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840. SIAM, 2016

[7] Reyan Ahmed, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, Faryad Darabi Sahneh, and Richard Spence. Approximation algorithms and an integer program for multi-level graph spanners. In *Proceedings of the Special Event on Analysis of Experimental Algorithms*, 2019

[79] Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. Lasserre integrality gaps for graph spanners and related problems. *arXiv preprint arXiv:1905.07468*, 2019

9.1. Quick Introduction to Linear Programming and Duality

(For an in-depth reference on Linear Programming and related topics, see [35].) A Linear Programming problem (LP) is an optimization problem of the form

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{4}$$

when $c, x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are column vectors, and A is an $m \times n$ matrix. The number of rows and columns of A are equal to the number of constraints and variables respectively.

If we add the additional constraint that the variables x_i (components of x) are integers, the corresponding problem is called an Integer Linear Programming problem (ILP). A 0-1 ILP is an ILP where the variables x_i are restricted to be binary (0 or 1). A *mixed* ILP is one where some variables x_i are required to be integers.

Computing an optimal solution to an ILP is NP–hard, as NP–hard problems can be formulated as instances of an ILP. To obtain a reasonable approximation, one usually constructs a *relaxation* of the ILP problem, where the variables are no longer required to be integral. The *integrality gap* is defined as the maximum ratio between the ILP solution and the solution of the corresponding relaxed LP.

The most obvious way to obtain an integer solution from a non-integer vector optimizing the relaxation is to simply round to the closest feasible integer vector. However, there are sometimes far better ways to convert to an integer solution, and more sophisticated tools have been developed to address this rounding. One such a tool is the *lift and project* method, where we introduce new variables and lift the LP problem to higher dimensions, and then project it back to get a better approximation to the ILP solution.

To illustrate, let's assume that x_1 and x_2 are binary variables corresponding to paths P_1, P_2 in G, with the meaning that if $x_i = 1$ we add the path to the spanner and if $x_i = 0$ we don't add that path. Say we also want to force the solution to have at least one of the two paths. The obvious way of doing this is by adding a constraint

$$x_1 + x_2 \geq 1$$

We are looking for a better alternative to the above constraint. Consider the following non-linear constraint

$$(1 - x_1)(1 - x_2) = 0$$

with $0 \leq x_i \leq 1$. This is a better representation if we want to allow variables to have non-integer values, as it forces one of the variables to take the integer value of one. To overcome non-linearity we have to *lift* the problem and introduce a new variable $x' = x_1 x_2$. We then have the corresponding linear constraint:

$$0 = (1 - x_1)(1 - x_2) = 1 - x_1 - x_2 + x_1 x_2 = 1 - x_1 - x_2 + x'$$

we cannot guarantee that $x' = x_1 x_2$ in the final solution, but we know that this relaxation is not worse than the simple LP relaxation because for $x' \geq 0$, $1 - x_1 - x_2 + x' = 0$ implies $x_1 + x_2 \geq 1$. This is the basic idea of the lift and projection method.

In general, when we are dealing with many such constraints the degree of the corresponding non-linear polynomial can get arbitrarily large. Hence, one can consider an upper bound d for the degree of the corresponding non-linear polynomial. This choice will produce a *hierarchy* of LP relaxations to the original ILP problem. An example of such a method with a hierarchy of LP relaxations is the *Sherali–Adams* lift and projection method. There is also a slightly stronger version known as *Lasserre* hierarchy which has been applied to spanner problems.

9.2. An LP Relaxation of the Multiplicative Spanner Problem

Let us look at a well known LP relaxation of the *directed* t–spanner problem, given by Dinitz and Krauthgamer [77], where the problem is to compute a t–spanner of a strongly connected directed graph with the minimum number of edges. For an edge $(u, v) \in E$, let $\mathcal{P}_{uv}$ denote the set of all $u \to v$ paths whose length is at most $td_G(u, v)$. The indicator variable x_e is 1 if e belongs to the spanner and 0 otherwise. The variable f_P represents the flow along path $P \in \mathcal{P}_{uv}$.

$$\min \quad \sum_{e \in E} x_e \tag{5}$$

$$\text{s.t.} \quad \sum_{p \in P_{uv} : e \in p} f_P \geq x_e \quad \forall (u, v) \in E, \forall e \in E \tag{6}$$

$$\sum_{P \in P_{uv}} f_P \geq 1 \qquad \forall (u, v) \in E \tag{7}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{8}$$

$$f_P \geq 0 \qquad \forall (u, v) \in E, \forall P \in \mathcal{P}_{uv} \tag{9}$$

$$\tag{10}$$

If the number of paths is polynomial then the LP problem can be solved optimally, as both the number of variables and constraints will be polynomial. In general, this is not true and we may have an exponential number of variables and constraints at the same time. Here, the idea is to write down the dual LP problem:

$$\max \quad \sum_{(u,v)\in E} z_{u,v} \tag{11}$$

$$\text{s.t.} \quad \sum_{(u,v)\in E} y^e_{u,v} \leq 1 \quad \forall e \in E \tag{12}$$

$$z_{u,v} - \sum_{e\in E} y^e_{u,v} \leq 0 \quad \forall (u,v) \in E, \forall P \in \mathcal{P}_{uv} \tag{13}$$

$$z_{u,v} \geq 0 \qquad \forall (u,v) \in E \tag{14}$$

$$y^e_{u,v} \geq 0 \qquad \forall (u,v) \in E, \forall e \in E \tag{15}$$

$$\tag{16}$$

The next step is to look at the constraint of the dual problem and observe whether checking a subset of constraints can be done in polynomial time (in more technical terms we need to construct a separation oracle for the dual LP; see [35]). In this case, the only nontrivial set of constraints is of the form $z_{u,v} - \sum_{e\in E} y^e_{u,v} \leq 0$, and we have an exponential number of constraints of this type. One then shows that a polynomial subset of these constraints gives a good approximation to the dual problem. More specifically, the rest of the constraints are violated only by a factor of $(1-\epsilon)$, i.e. there might be paths with $(1-\epsilon)z_{u,v} \leq \sum_{e\in E} y^e_{u,v}$.

We can, therefore, obtain an optimal solution to the dual problem up to a $(1-\epsilon)$ factor. Using a strong version of the duality theorem, if we remove the variables from the primal problem corresponding to these constraints, we acquire the following result.

Theorem 9.1 ([77]). *For any $\epsilon \geq 0$, there is an algorithm that in polynomial time computes a $(1+\epsilon)$–approximation to the optimal solution of above primal LP problem* (5)–(9).

Dinitz and Krauthgamer also prove an integrality gap for this problem:

Theorem 9.2 ([77]). *The integrality gap of the primal LP* (5)–(9) *for the directed t–spanner problem is $\Omega(\frac{1}{t}n^{\frac{1}{3}-\epsilon})$, for any constant $\epsilon > 0$.*

9.3. ILPs for Pairwise Spanners

Now we turn to considering ILPs which exactly solve the spanner problem for a given input graph. ILP solvers typically require significantly more computation than LP relaxations, but are practical in some instances.

Let $G = (V, E)$ be a graph (weighted or unweighted), let $P \subseteq V \times V$ be the desired pairs for which the spanner condition will hold, and let $t \geq 1$ be a prescribed stretch factor. Recall that w_e is the weight of an edge $e \in E$ (with $w_e = 1$ in the unweighted case). Let $x_e = 1$ if e is selected in the spanner, and $x_e = 0$ otherwise. Further, let $\mathcal{P}_{uv}$ denote the set of all u–v paths of length at most $t \cdot d_G(u,v)$, let $\mathcal{P}$ denote the union of all such paths ($\mathcal{P} = \bigcup_{(u,v)\in P} \mathcal{P}_{uv}$),

and let $y_\rho = 1$ if the path $\rho \in \mathcal{P}$ is selected in the spanner, and 0 otherwise. Let $\delta_\rho^e = 1$ if edge e is on path ρ, and 0 otherwise. The ILP formulation by Sigurd and Zachariasen for the pairwise multiplicative spanner problem is as follows:

$$\min \sum_{e \in E} w_e x_e \text{ subject to} \tag{17}$$

$$\sum_{\rho \in \mathcal{P}_{uv}} y_\rho \delta_\rho^e \leq x_e \qquad \forall e \in E; \forall (u,v) \in P \tag{18}$$

$$\sum_{\rho \in \mathcal{P}_{uv}} y_\rho \geq 1 \qquad \forall (u,v) \in P \tag{19}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{20}$$

$$y_\rho \in \{0,1\} \qquad \forall \rho \in \mathcal{P} \tag{21}$$

Constraint (18) enforces that if a path $\rho \in \mathcal{P}_{uv}$ is selected in the spanner, then all edges on that path are selected as well. Constraint (19) enforces that every pair $(u,v) \in P$ has at least one selected t–spanner path.

This ILP has $O(|\mathcal{P}|)$ constraints which can be exponential in the size of the graph; however, the authors apply column generation to solve a restricted master problem (i.e. using only a subset of $\mathcal{P}$ for the constraints), and show that this procedure yields an optimal solution. The authors then use the ILP to show that the greedy algorithm of Althöfer et al. (see Section 4) performs optimally in many cases.

Sigurd and Zachariasen left as an open question whether the pairwise spanner problem admits a practical polynomial size ILP. Recently, Ahmed et al. [7, 8] proposed an affirmative resolution to this question, giving a polynomial-size flow-based ILP formulation for the pairwise spanner problem which can be used to compute minimum-weight graph spanners with arbitrary distortion function, even if the input graph is directed, and experiments were provided in support of its feasibility.

The ILP of Ahmed et al. [7, 8] is set up as follows. Again let $P \subseteq V \times V$, and let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be a distortion function satisfying $f(x) \geq x$ for all x (note the function need not be continuous). Replacing each edge (u,v) with two directed edges (u,v) and (v,u) of equal weight. Let $\bar{E}$ be the set of all directed edges (so $|\bar{E}| = 2|E|$). Given $(i,j) \in \bar{E}$ and $(u,v) \in P$, the variable $x_{(i,j)}^{uv}$ is 1 if edge (i,j) is included in the selected u–v path in the spanner G', or 0 otherwise.

The formulation is then as follows; by $\text{In}(v), \text{Out}(v)$ we mean the set of incoming and outgoing edges to v (respectively), and the vertices are arbitrarily ordered so that constraints (24)–(26) are well-defined.

$$\text{Minimize} \sum_{e \in E} w_e x_e \text{ subject to} \tag{22}$$

$$\sum_{(i,j)\in\bar{E}} x^{uv}_{(i,j)} w_e \leq f(d_G(u,v)) \quad \forall (u,v) \in P, u < v; e = (i,j) \in E \tag{23}$$

$$\sum_{(i,j)\in \mathrm{Out}(i)} x^{uv}_{(i,j)} - \sum_{(j,i)\in \mathrm{In}(i)} x^{uv}_{(j,i)} = \begin{cases} 1 & i = u \\ -1 & i = v \\ 0 & \text{else} \end{cases} \qquad \forall (u,v) \in P, u < v; \forall i \in V \tag{24}$$

$$\sum_{(i,j)\in \mathrm{Out}(i)} x^{uv}_{(i,j)} \leq 1 \qquad \forall (u,v) \in P, u < v; \forall i \in V \tag{25}$$

$$x^{uv}_{(i,j)} + x^{uv}_{(j,i)} \leq x_e \qquad \forall (u,v) \in P, u < v; \forall e = (i,j) \in E \tag{26}$$

$$x_e, x^{uv}_{(i,j)} \in \{0,1\} \tag{27}$$

This ILP has $2|E||P|$ binary variables, or $2|E|\binom{|V|}{2} = |E||V|(|V|-1)$ variables in the full spanner problem where $P = V \times V$. In the multiplicative spanner case, the number of ILP variables can be significantly reduced (see [7] for more details). These reductions are somewhat specific to multiplicative spanners, and so it would be interesting to determine if other simplifications are possible for more general distortion. We note that the distortion f can have any form, which allows for considerable flexibility in the types of pairwise spanners represented by the above ILP.

Dinitz, Nazari and Zhang [79] prove a stronger integrality gap than Theorem 9.2 by considering Lasserre lifts of the flow LP. They essentially show that even the strongest lift and project methods cannot help significantly in the approximation of spanners.

Chlamtáč, Dinitz, and Krauthgamer [67] give an LP relaxation for the lowest-degree 2–spanner (LD–2SP) problem using the Sherali–Adams lift method. They establish a polynomial time algorithm with approximation ratio $O(\Delta^{3-2\sqrt{2}})$, where Δ is the maximum degree of the graph.

Chlamtáč and Dinitz [65] formulate an (I)LP for the lowest-degree k–spanner (LD–kSP) problem. They also provide a hardness of approximation result based on the Min-REP problem. The basic idea is that small REP-covers correspond to small spanners, when constructing a yes/no decision oracle. We refer the reader to [65] for the full details of REP.

Theorem 9.3. *For any integer $k \geq 3$, there is no polynomial-time algorithm that can approximate lowest degree k-spanner better than $\Delta^{\Omega(1/k)}$ unless $NP \subset BPTIME(2^{polylog(n)})$, where Δ is the maximum degree of the input graph.*

Berman et al. [33] formulate an instance of anti-spanner LP problems. An anti-spanner is a subset of edges that don't form a t–spanner and is maximal with respect to this property.

Dinitz and Zhang [80] prove that the approximation ratio for the t–spanner problem is at most $O(n^{\frac{1}{3}})$. They also prove an integrality gap for a flow-based LP.

10. Distributed and Streaming Algorithms

[84] Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms (TALG)*, 1(2):283–323, 2005

[100] Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\varepsilon, \beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006

[74] Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In *International Symposium on Distributed Computing*, pages 179–192. Springer, 2007

[75] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing*, pages 273–282. ACM, 2008

[23] Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Information Processing Letters*, 106(3):110–114, 2008

[138] Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. In *Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing*, pages 253–262. ACM, 2008

[85] Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011

[125] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, pages 375–382. ACM, 2013

[115] Michael Kapralov and David Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, pages 272–281. ACM, 2014

[55] Keren Censor-Hillel, Telikepalli Kavitha, Ami Paz, and Amir Yehudayoff. Distributed construction of purely additive spanners. In *International Symposium on Distributed Computing*, pages 129–142. Springer, 2016

[99] Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(3):29, 2016

[63] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):33, 2018

[90] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Transactions on Algorithms (TALG)*, 15(1):1–29, 2018

[56] Keren Censor-Hillel, Ami Paz, and Noam Ravid. The sparsest additive spanner via multiple weighted BFS trees. In *22nd International Conference on Principles of Distributed Systems*, 2019

[89] Michael Elkin and Shaked Matar. Near-additive spanners in low polynomial deterministic congest time. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 531–540, 2019

[87] Michael Elkin, Arnold Filtser, and Ofer Neiman. Distributed construction of light networks. *arXiv preprint arXiv:1905.02592*, 2019

In distributed algorithms, computations happen at each vertex of the graph and processors communicate to collectively accomplish a graph-level task. In Peleg's terminology [134], two models of distributed computing are LOCAL and CONGEST, where respectively, unbounded and short messages are communicated between processors. Distributed algorithms for computing spanners are considered particularly important because a common use of spanners is to preprocess a distributed network, sparsifying the network without changing its communication latency too much. Indeed, the historically first appearances of spanners were in applications to *synchronizers* [18, 137, 136], which are protocols that convert between the asynchronous distributed model (where nodes send messages to their neighbors at unpredictable times) and the synchronous distributed model (where nodes send messages at each tick of a centralized clock). It was only later that applications to centralized computation were developed, and that spanners were considered an interesting object of study in their own right.

Elkin [84] gives a randomized distributed algorithm which produces a $(1+\varepsilon,\beta)$–spanner with $\beta = \left(\frac{k}{\varepsilon}\right)^{O(\log k)} \rho^{-\frac{1}{\rho}}$ of size $\widetilde{O}(\beta n^{1+\frac{1}{k}})$ in time $O(n^{1+\frac{1}{2k}})$. Elkin and Matar [89] study distributed algorithms computed in the CONGEST model for finding near-additive spanners, i.e. $(1+\varepsilon,\beta)$–spanners. The difference of their approach from previous works is that the algorithm is deterministic. The spanners constructed have three parameters ε, k, ρ and require $\beta = \left(\frac{O(\log k\rho+\rho^{-1})}{\rho\varepsilon}\right)^{\log k\rho+\rho^{-1}+O(1)}$, and yield $(1+\varepsilon,\beta)$–spanners of size $O(\beta n^{1+\frac{1}{k}})$ with time complexity $O(\beta n^{\rho}\rho^{-1})$.

Pettie [138] describes algorithms for computing sparse low distortion spanners in distributed networks and provides some non-trivial lower bounds on the tradeoff between time, sparseness, and distortion. The algorithms assume a synchronized distributed network, where relatively short messages may be communicated in each time step. The first result is a fast distributed algorithm for finding an $O(2^{\log^* n}\log n)$–spanner with size $O(n)$.[6] The second result is a new class of efficiently constructible (α,β)–spanners called Fibonacci spanners whose distortion improves with the distance being approximated. At their sparsest, Fibonacci spanners can have nearly linear size, namely $O(n(\log\log n)^{\phi})$, where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

Lenzen and Peleg [125] show that additive 2–spanners can be computed in the CONGEST model in $O(n^{\frac{1}{2}}\log n + \mathrm{diam}(G))$ rounds. In [56], the authors give a sequential algorithm for computing an additive 6–spanner for unweighted graphs which yields near-optimal sparsity $O(n^{\frac{4}{3}}\log^{\frac{4}{3}} n)$ edges, but allows for a distributed construction algorithm (using the CONGEST model of computation) which runs via an efficient construction of weighted BFS trees in $O(n^{\frac{2}{3}}\log^{-\frac{1}{3}} n + \mathrm{diam}(G))$ rounds.

[6] Recall that $\log^* n$ is the inverse tower function; essentially, it is the least height of a tower $2^{2^{2^{\cdots}}}$ whose value is $> n$.

In Derbel et al. [74] several deterministic distributed algorithms have been provided to compute different kinds of spanners. They give an algorithm to construct an $O(k)$–spanner of an unweighted graph with $O(kn^{1+\frac{1}{k}})$ edges in $O(\log^{k-1} n)$ time. Also, they have shown that in $O(\frac{\log n}{\epsilon})$ time one can construct a spanner with $O(n^{\frac{3}{2}})$ edges that is both a 3–spanner and a $(1+\epsilon, 8\log n)$–spanner. Furthermore, they have shown that in $n^{O\left(\frac{1}{\sqrt{\log n}}\right)} + O\left(\frac{1}{\epsilon}\right)$ time one can construct a spanner with $O(n^{\frac{3}{2}})$ edges which is both a 3–spanner and a $(1+\varepsilon, 4)$–spanner. In [75], the authors propose a deterministic, distributed algorithm that computes a $(2k-1, 0)$–spanner in k rounds of computation (or $3k-2$ rounds if n is not known) with $O(kn^{1+\frac{1}{k}})$ edges. Also, it is further shown that no (randomized) algorithm producing $O(n^{1+\frac{1}{k}})$ size spanners (possibly with additive stretch) can run distributively in sub-polynomial rounds of computation. Recently, Elkin, Filtser, and Neiman [87] gave distributed constructions of spanners with good controls on lightness.

Censor-Hillel et al. [55] study the complexity of distributed constructions of purely additive spanners. The provided spanner algorithms have three general steps: first, each node tosses a coin to be a cluster center; second, each cluster center tosses another coin to be a BFS tree; third, add to the current graph edges that are part of certain short paths.

Elkin [85] gave a streaming algorithm for building multiplicative spanners with near-optimal size/stretch tradeoff and $O(1)$ expected processing time per edge. A close variant of this algorithm also gives a fully dynamic algorithm, with similar update time, for maintaining spanners with near-optimal size/stretch tradeoff. (A similar result was proved simultaneously by Baswana [23].) Streaming algorithms for mixed spanners were given in [100, 90].

Feigenbaum et al. [104] consider the semi-streaming model where at most a logarithmic number of passes over the graph stream are allowed. It was shown that a single pass is enough to compute a $(1+\varepsilon)\log n$–spanner for a weighted undirected graph under mild conditions for weight values.

A somewhat distinct, yet relevant, body of works pertains to dynamic stream models where both insertion and deletion are allowed [9]. Kapralov and Woodruff [115] construct linear sketches of graphs that (approximately) preserve the spectral information of the graph in a few passes over the stream of graph data. They use a multi-layer clustering approach to construct a multiplicative 2^k–spanner using $O(n^{1+\frac{1}{k}})$ bits of space from a dynamic stream of inputs.

11. Special Types of Spanners

In this section, we describe some of the many variants of spanners that have been considered in the literature. We emphasize primarily the definitions and basic results for brevity, and readers are encouraged to consult the listed papers for more information.

11.1. Emulators

[81] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000

[150] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 802–809. Society for Industrial and Applied Mathematics, 2006

[60] Hsien-Chih Chang, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Near-Optimal Distance Emulator for Planar Graphs. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA)*, volume 112, pages 16:1–16:17, 2018

The word *spanner* generally implies a *subgraph* that approximates the distance metric of an input graph. When the approximating graph can instead be arbitrary (not necessarily a subgraph), it is called an *emulator*. Spanners are a special case of emulators. Emulators are typically allowed to be weighted and sometimes directed, even if the input graph is not.

Emulators hold an interesting place in the literature because optimal bounds are essentially known for the size/error tradeoff of emulators, whereas the optimal bounds for spanners remain open. The first example of this is for emulators with purely additive error. In [81], the authors prove that every n–node undirected unweighted input graph has an emulator on $O(n^{\frac{4}{3}})$ edges with 4–additive error. This is optimal in the sense that neither the edge bound nor the additive error in this result can be unilaterally improved (this follows from the Girth Conjecture which is confirmed in the relevant setting $k = 3$; see Section 2.4). However, it is only known that every graph has a *spanner* on $O(n^{\frac{4}{3}})$ edges and 6–additive error [26], and it is still open whether one can unilaterally improve the additive error to 4.

Essentially tight bounds for emulators with sublinear error are known as well. An important result in [150] is a simple construction of emulators with $O(n^{1+\frac{1}{2^k-1}})$ edges and error function $f(d) = d + O(d^{1-\frac{1}{k}})$, for any fixed positive integer k. The main result of [4] is that this is essentially optimal; see Section 8 for more details.

Emulators have also been central to a line of research seeking fast approximation algorithms for all-pairs shortest paths (APSP). In [81], the authors give approximate APSP algorithms essentially by constructing a series of emulators and then running standard shortest path algorithms on these emulators, which are fast due to their sparsity. A related approach was taken in [60]: the authors prove that, for any undirected unweighted planar input graph $G = (V, E)$ and set of terminals $T \subseteq V$, there is an emulator on $\widetilde{O}(\min\{|T|^2, \sqrt{n|T|}\})$ that *exactly* preserves the distances between node pairs in T (called a *distance emulator* in analogy with distance preservers). With a similar technique to the above, the authors then show that one can compute distances between any $|T| = \widetilde{O}(n^{\frac{1}{3}})$ terminals in an n–node planar graph in $\widetilde{O}(n)$ time.

11.2. Approximate Distance Oracles

[148] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, 2001

[141] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Automata, Languages and Programming*, pages 261–272, 2005

[29] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms (TALG)*, 2(4):557–577, 2006

[24] Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 609–621. Springer, 2008

[146] Christian Sommer. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)*, 46(4):45, 2014

Related to emulators are *approximate distance oracles*. Given a graph G, a *distance oracle* is a data structure that can answer exact distance queries between any pair of vertices $u, v \in V$. By preprocessing the graph using algorithms such as All-Pairs Shortest Path (APSP), these queries can be answered in constant time. However, algorithms like APSP are inefficient for this purpose as they have a runtime of $O(n^3)$ and space requirement of $O(n^2)$. If approximate distances are acceptable to the user, then the runtime and space bounds can be much improved. An *approximate distance oracle* is a data structure which approximately answers distance queries, and similar to a spanner, the answer of a query is often allowed to be at most t times the actual distance between the two vertices. In this case, spanners are special cases of approximate distance oracles, but are more restrictive given that they need to be subgraphs. Related objects are *emulators* which are graphs $G^* = (V, E^*)$ on the same vertices as the original graph, but with different edge sets, but so that distances in G^* approximate distance in G [150].

Thorup and Zwick [148] propose a randomized algorithm to compute an approximate distance oracle with a stretch $(2k-1)$ such that for all $u, v \in V$, the oracle returns a distance $d_k(u, v)$ with $d_k(u, v) \leq (2k-1)d_G(u, v)$. Note that the algorithm will not return a subgraph of G, and hence is not a spanner algorithm, but is rather returning an approximation to the length of the shortest paths in G. This algorithm has an improved $O(kmn^{\frac{1}{k}})$ runtime and $O(kn^{1+\frac{1}{k}})$ space requirement.

As a follow-up to [148], Roditty et al. [141] propose two extensions: restricting the approximation to a set of sources $S \subseteq V$, and derandomizing the algorithm. The former results in an algorithm with $O(km|S|^{\frac{1}{k}})$ runtime and $O(kn|S|^{\frac{1}{k}})$ space requirement, which becomes very useful when $|S| \ll n$. The deterministic variant of the algorithm leads to an increase in runtime by a logarithmic factor but retains the same space requirements as the randomized algorithm.

Finally, Roditty et al. [141] extend the derandomization technique to the linear time algorithm by Baswana and Sen [28] to compute a $(2k-1)$–spanner for a given graph G. The randomization in this algorithm involves the initial step of choosing cluster centers randomly with probability $|V|^{-\frac{1}{2}}$. The authors propose doing this deterministically by building a bipartite graph of $B = (V_1, V_2, E)$

and applying the linear time algorithm for computing a closed dominating set on this graph to find the cluster centers.

Baswana and Sen [29] provide an algorithm to construct approximate distance oracles in expected $O(n^2)$ time if the graph is unweighted. One of the new ideas used in the algorithm also leads to the first expected linear time algorithm for computing an optimal size $(2,1)$–spanner of an unweighted graph, whereas existing algorithms had $\Theta(n^2)$ running time. In [24], they break this quadratic barrier at the expense of introducing a (small) constant additive error for unweighted graphs. In achieving this goal, they have been able to preserve the optimal size–stretch tradeoffs of the oracles. One of their algorithms can be extended to weighted graphs, where the additive error becomes $2w_{max}(u,v)$, where $w_{max}(u,v)$ is the heaviest edge in the shortest path between vertices u and v.

In [81], a $\widetilde{O}(\min\{n^{\frac{3}{2}}m^{\frac{1}{2}}, n^{\frac{7}{3}}\})$–time algorithm for computing all distances in a graph with an additive error of at most 2 has been described. For every even $k > 2$, they describe an $\widetilde{O}(\min\{\ n^{2-\frac{2}{(k+2)}}m^{\frac{2}{(k+2)}}, n^{2+\frac{2}{(3k-2)}}\})$–time algorithm for computing all distances with an additive one-sided error of at most k. Then, they show that every unweighted graph on n vertices has an additive 2–spanner with $\widetilde{O}(n^{\frac{3}{2}})$ edges and a 4–spanner with $\widetilde{O}(n^{\frac{4}{3}})$ edges. Finally, they show that any weighted undirected graph on n vertices has a multiplicative 3–spanner with $\widetilde{O}(n^{\frac{3}{2}})$ edges and that such a 3–spanner can be built in $\widetilde{O}(mn^{\frac{1}{2}})$ time.

Sommer [146] gives a survey on the shortest path query problem, with both theoretical results and practical applications, and should be consulted for further reading. This problem has numerous applications including Geographic Information Systems (GIS), proximity search in a database, packet routing, and metabolic networks. The shortest path query problem is different than the classical single source shortest path problem and APSP problem in that there are two steps: the first is a preprocessing algorithm, and the second is to process queries efficiently. The preprocessing algorithm provides a data structure which can be used to process multiple queries without changing the structure. Unlike the APSP problem, here the data structure is computed in a space efficient manner.

11.3. Diameter and Eccentricity Spanners

[20] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 267–280, 2018

[69] Keerti Choudhary and Omer Gold. Extremal distances in directed graphs: Tight spanners and near-optimal approximation algorithms. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 495–514. SIAM, 2020

Choudhary and Gold [69] consider the following problem: given a directed graph $G = (V, E)$, compute a subgraph H with the property that the graph diameter is preserved up to a multiplicative factor t, which the authors term a *t–diameter spanner*. The authors show that, given an unweighted directed graph

G, there is a polynomial time algorithm that computes a 1.5–diameter spanner containing $O(n^{\frac{3}{2}}\sqrt{\log n})$ edges, in time $\widetilde{O}(m\sqrt{n})$ deterministically. They define the *t-eccentricity spanner* similarly, where the eccentricity[7] of each vertex v in H should be no more than t times the eccentricity of v in G. The authors show that given a weighted directed graph, there is a Las Vegas algorithm which computes a 2–eccentricity spanner containing $O(n \log^2 n)$ edges in $\widetilde{O}(m)$ expected time.

Hardness

Backurs et al. [20] show that unless the Strong Exponential Time Hypothesis[8] (SETH) fails, there is no $O(n^{\frac{3}{2}-\delta})$ time algorithm ($\delta > 0$) which can approximate the diameter of a sparse weighted graph with ratio better than $\frac{5}{3}$. Unless SETH fails, no $O(n^{\frac{3}{2}-\delta})$–time algorithm can approximate the eccentricities of an undirected, unweighted sparse graph with ratio better than $\frac{9}{5}$.

11.4. Ramsey Spanning Trees

[5] Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. Ramsey spanning trees and their applications. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1650–1664. Society for Industrial and Applied Mathematics, 2018

Abraham et al. [5] consider a generalization of the metric Ramsey problem to graphs. The problem is to find a subset $S \subseteq V$ of sources and a spanning tree T which gives a t–spanner for small t; i.e. one wants to simultaneously find the sources and sourcewise spanner for a given graph. Their main result is to find (in polynomial time) a subset S of size at least $n^{1-\frac{1}{k}}$ and spanning tree T which is an $(O(k \log\log n), 0, S \times V)$–spanner of G.

11.5. Slack Spanners

[57] T-H Hubert Chan, Michael Dinitz, and Anupam Gupta. Spanners with slack. In *European Symposium on Algorithms (ESA)*, pages 196–207. Springer, 2006

In [57], the authors consider the problem of computing a subgraph H that preserves shortest distances to all but an ϵ fraction of vertices. Specifically, $H = (V, E_H)$ is an *ϵ–slack spanner* with distortion D if for all $v \in V$, the distances in H are preserved for the furthest $(1-\epsilon)n$ vertices from v up to distortion D. The authors show that given a graph G, one can find a subgraph H which is a $O(\log \frac{1}{\epsilon})$–distortion ϵ–slack spanner containing $O(n)$ edges, for every ϵ. Furthermore, they showed that gracefully degrading existing spanners also have $O(1)$ average stretch with size $O(n)$.

[7] The *eccentricity* of a vertex $v \in V$ is the maximum graph distance between v and any other vertex.

[8] Let $s_k := \inf\{\delta : k\text{–SAT has an } O(2^{n\delta})\text{–time algorithm}\}$. The Strong Exponential Time Hypothesis (SETH) asserts that $\lim_{k\to\infty} s_k = 1$.

11.6. Spanners in Sparse Graphs

[82] Feodor F Dragan, Fedor V Fomin, and Petr A Golovach. Spanners in sparse graphs. *Journal of Computer and System Sciences*, 77(6):1108–1119, 2011

Dragan et al. [82] show that the problem of computing a t–spanner of a planar graph G of treewidth[9] at most k is fixed parameter tractable parametrized by k and t. They then show that the problem is also fixed parameter tractable on graphs that do not contain a fixed apex graph[10] as a minor.

11.7. Steiner t-Spanners

[11] 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

[112] Dagmar Handke and Guy Kortsarz. Tree spanners for subgraphs and related tree covering problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 206–217. Springer, 2000

Given an unweighted graph $G = (R \cup S, E)$ where R is the set of terminals and the induced graph of R is connected, a subgraph H of G is a *Steiner t–spanner* of G if $d_H(u, v) \leq t d_G(u, v)$, for all $u, v \in R$. Althöfer et al. [11] deal with absolute lower bounds on the number of edges that an Steiner t–spanner can have.

Hardness

Handke and Kortsarz [112] show that the minimum Steiner t–spanner cannot be approximated with ratio $(1 - \varepsilon) \log |S|$ for any $\varepsilon > 0$, unless NP $\subseteq$ DTIME($n^{\log \log n}$) where $S = V \setminus R$ is the set of Steiner vertices.

11.8. Tree t–spanners

[53] Leizhen Cai and Derek G Corneil. Tree spanners. *SIAM Journal on Discrete Mathematics*, 8(3):359–387, 1995

[101] Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM Journal on Computing*, 38(5):1761–1781, 2008

[12] Eduardo Álvarez-Miranda and Markus Sinnl. Mixed-integer programming approaches for the tree t^*-spanner problem. *Optimization Letters*, pages 1–17, 2018

A *tree t-spanner* of a graph G is a spanning tree of G which is also a t-spanner. Cai and Corneil [53] show that a tree 1-spanner, if it exists, can be found in $O(m \log \beta(m, n))$ time, where $\beta(m, n) = \min\{i : \log^i n \leq m/n\}$. Further, a tree 2-spanner (if it exists) can be found in linear $O(m + n)$ time. For $t \geq 4$, determining whether a tree t-spanner exists is NP-hard via a reduction from 3-SAT.

[9] The *treewidth* of a graph G is the minimum width over all tree decompositions of G, where the width of a tree decomposition equals 1 less than the maximum size of a vertex set in the decomposition.

[10] An *apex graph* is a graph that is planar if one vertex is removed.

Emek and Peleg [101] consider the optimization version, referred to as *minimum max-stretch spanning tree*: given a graph G, find a spanning tree minimizing the stretch factor t. Emek and Peleg give a divide-and-conquer $O(\log n)$ approximation algorithm for MMST. Álvarez-Miranda and Sinnl [12] give a compact, polynomial-size mixed integer program (MIP) for this problem. They obtain a formulation with fewer variables by using a cut-based formulation to impose a spanning tree topology, then using Benders decomposition to get rid of path variables. For other approximation algorithms, see [145, 147].

Hardness

Emek and Peleg [101] show that it is NP–hard to approximate the MMST problem with ratio better than $\frac{5}{4}$, and that MMST cannot be approximated additively by any $o(n)$ term unless P = NP.

11.9. Lowest-degree k-Spanners

[123] Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM J. Comput.*, 27:1438–1456, 1994

[67] Eden Chlamtáč, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. *In 16th Annual Symposium on Foundations of Computer Science*, 05 2012

[65] Eden Chlamtáč and Michael Dinitz. Lowest-degree k-spanner: Approximation and hardness. *Theory of Computing*, 12(15):1–29, 2016

Given integer $k \geq 2$, the *lowest-degree k-spanner* (LD-kSP) problem is to compute a k-spanner of a given undirected graph G, with the objective of minimizing the maximum degree of any vertex in the spanner. Let Δ denote the maximum vertex degree of the input graph G.

For $k = 2$, Kortsarz and Peleg [123] give an LP-based $\tilde{O}(\Delta^{1/4})$-approximation for LD-2SP. This was improved more recently by Chlamtáč et al. [67], who give an $\tilde{O}(\Delta^{3-2\sqrt{2}-\varepsilon}) \approx \tilde{O}(\Delta^{0.172})$-approximation for LD-2SP. For $k \geq 3$, Chlamtáč et al. [65] give an LP-based $\tilde{O}(\Delta^{(1-(1/k))^2})$-approximation for LD-$k$SP (see Section 9.2).

Hardness

For $k = 2$, Kortsarz and Peleg show that LD–2SP cannot be approximated with ratio better than $(\ln n)/5$ unless NP $\subseteq$ DTIME($n^{\log \log n}$) via a reduction from the set cover[11] problem.

For $k \geq 3$, Chlamtáč and Dinitz [65] show that LD–kSP admits no polynomial time approximation with ratio $\Delta^{\Omega\left(\frac{1}{k}\right)}$ unless NP $\subseteq$ BPTIME($2^{\text{polylog}(n)}$). Further, if the input graph is directed (in which case the "degree" of a spanner vertex is the sum of its in-degree and out-degree), then directed LD–kSP cannot be approximated with ratio $\Delta^{O(1)}$ unless NP $\subseteq$ DTIME($2^{\text{polylog}(n)}$).

[11] The *set cover* problem is as follows: given a universe $U = \{1, 2, \ldots, n\}$, a collection of sets $S_1, \ldots, S_m$ such that $\bigcup_{i=1}^{m} S_i = U$, and an integer $t \geq 0$, determine if there is a collection of t sets whose union is U.

[14] Sunil Arya, Gautam Das, David M Mount, Jeffrey S Salowe, and Michiel Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 489–498, 1995

[98] Michael Elkin and Shay Solomon. Optimal Euclidean spanners: Really short, thin, and lanky. *Journal of the ACM (JACM)*, 62(5):1–45, 2015

[58] T-H Hubert Chan, Mingfei Li, Li Ning, and Shay Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015

[48] Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2371–2379. Society for Industrial and Applied Mathematics, 2019

[108] Lee-Ad Gottlieb. A light metric spanner. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 759–772. IEEE, 2015

For a metric space $M = (V, d)$, its *doubling dimension* $\mathcal{D}(M)$ is the quantity

$$\mathcal{D}(M) := \inf\{d \in \mathbb{R} \ : \text{for all } v \in V, r > 0, \text{ the ball } B(v, r) \text{ can be covered by } 2^d \text{ balls of radius } r/2.\}$$

There is a body of work that builds spanners for graphs assuming bounds on the doubling dimension of their associated metric space. This is motivated in part as a natural stepping stone between general graph spanners and *geometric spanners*, where the input graphs are essentially finite submetric spaces of Euclidean space (the book [129] covers these extensively, but they are out of scope for this survey). In particular, a basic fact is that $\mathbb{R}^d$ and any submetric space thereof has doubling dimension $\Theta(d)$, and some results for geometric spanners in $\mathbb{R}^d$ generalize readily to any graph with similar doubling dimension. For example, there was a famous conjecture by Arya et al. [14] that, given an n-point submetric space of $\mathbb{R}^{O(1)}$, in $O(n \log n)$ time one can construct a geometric spanner with constant maximum degree, $O(\log n)$ hop-diameter (i.e., maximum number of edges in a shortest path), and $O(\log n)$ lightness (i.e., total weight $O(\log n) \cdot W(\text{MST}(G))$). This conjecture was proved by Elkin and Solomon [98] (and later simplified by Chan et al. [58]), more generally in any metric space of constant doubling dimension.

An important fact about graphs of low doubling dimension is that the greedy construction algorithm (presented in Section 4) has an improved lightness analysis: following [108], Borradaile et al. [48] proved that a greedy $(1+\varepsilon)$–spanner of a graph of doubling dimension d has total weight $\varepsilon^{-O(d)} W(\text{MST}(G))$. (The corresponding bound for geometric graphs, which is tight, is in [129].)

12. Spanners For Changing Graphs

In recent years, there have been variants of spanners studied in the literature which typically involve changes allowed in the initial graph G, for example edge addition or deletion.

12.1. Fault Tolerant Spanners

[62] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010

[17] Giorgio Ausiello, Andrea Ribichini, Paolo G Franciosa, and Giuseppe F Italiano. Computing graph spanners in small memory: fault-tolerance and streaming. *Discrete Mathematics, Algorithms and Applications*, 2(04):591–605, 2010

[78] 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 (PODC)*, pages 169–178, 2011

[50] Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault tolerant additive and (μ, α)–spanners. *Theoretical Computer Science*, 580:94–100, 2015

[37] Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, pages 167–178. Springer, 2015

[132] Merav Parter. Vertex fault tolerant additive spanners. *Distributed Computing*, 30(5):357–372, October 2017

[41] Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1884–1900. Society for Industrial and Applied Mathematics, 2018

[44] Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 541–543. ACM, 2019

Chechik et al. [62] study *fault tolerant spanners* which were introduced originally for geometric graphs by Levcopoulos et al. [126]. Given a stretch parameter $t \geq 1$ and a fault parameter $f \in \mathbb{N}$, an f–vertex (resp. f–edge) fault tolerant multiplicative t–spanner is a subgraph $G' = (V, E')$ such that for any set $F \subseteq V$ (resp. $F \subseteq E$) with $|F| \leq f$, we have

$$d_{G' \setminus F}(u, v) \leq t \cdot d_{G \setminus F}(u, v), \quad u, v \in V \setminus F \quad (\text{ resp. } u, v \in V).$$

The main result of [62] is to provide a randomized algorithm which computes an f–vertex fault tolerant $(2k-1)$–spanner with high probability containing $O(f^2 k^{f+1} n^{1+\frac{1}{k}} \log^{1-\frac{1}{k}} n)$ edges in running time $O(f^2 k^{f+2} n^{3+\frac{1}{k}} \log^{1-\frac{1}{k}} n)$. For the edge fault tolerant case, their algorithm produces a $(2k-1)$–spanner with high probability containing $O(f n^{1+\frac{1}{k}})$ edges.

Ausiello et al. [17] give improved constructions of fault tolerant spanners both in terms of time and number of edges compared to [62]. They construct an f–vertex fault tolerant $(3, 2)$–spanner with $O(f^{\frac{4}{3}} n^{\frac{4}{3}})$ edges in $\widetilde{O}(f^2 m)$ time, and an f–vertex fault tolerant $(2, 1)$–spanner with $O(f n^{\frac{3}{2}})$ edges in $\widetilde{O}(fm)$ time.

Dinitz and Krauthgamer [78] improve on the size of a fault tolerant spanner compared to Chechik et al. [62] by replacing the factor k^{f+2} with f^2, thus giving a spanner polynomial in the fault size rather than exponential. The main results of [78] give a way to transform a spanner into a fault tolerant spanner. Specifically, if a t–spanner has $g(n)$ edges, then their algorithm produces an f–vertex fault tolerant t–spanner with $O(f^2 \log n) g(\frac{2n}{f})$ edges. Applying this to the greedy $(2k-1)$–spanner construction of [11] yields an f–vertex fault tolerant

$(2k-1)$–spanner with $\widetilde{O}(f^2 n^{1+\frac{2}{k+1}})$ edges. Additionally, in the unweighted case, they provide a $O(\log n)$–approximation algorithm for finding the minimal cost f vertex fault 2–spanner problem.

Bodwin et al. [41] give improved bounds on the size of fault tolerant spanners created by the direct analogue of the greedy algorithm of Althöfer et al. for fault tolerant spanners. They show that for any positive integer f, there is an f–vertex (resp. f–edge) fault tolerant $(2k-1)$–spanner with $O(f^{1-\frac{1}{k}} n^{1+\frac{1}{k}} \exp(k))$ edges. They also demonstrate that this bound is tight, up to the $\exp(k)$ factor, in the setting of vertex faults assuming Erdős' Girth Conjecture. Subsequently, Bodwin and Patel [44] greatly simplify the analysis of the so-called Fault Tolerant Greedy Algorithm, removing the $\exp(k)$ factor in the upper bound of [41], and proving existential optimality (up to constant factors) even if the Girth Conjecture fails.

Braunschvig et al. [50] study additive and linear fault tolerant spanners. They use a novel technique to combine a $(1,\beta)$–spanner construction with an $(\alpha,0)$–spanner construction to obtain a vertex and edge fault tolerant (α,β)–spanner. In particular, they prove that given an algorithm which produces an (α,β)–spanner of size $O(n^{1+\delta})$, then for any $\varepsilon > 0$, one can obtain an f_e–edge, f_v–vertex fault tolerant $(\alpha+\varepsilon,\beta)$–spanner of size $O((f_e+f_v)(\frac{\beta}{\varepsilon})^{f_e+f_v} n^{1+\delta} \log n)$ provided $\max\{f_e, f_v\} < \lceil \frac{\beta}{\varepsilon} \rceil + 1$. As a corollary, they use existing spanner constructions of Elkin and Peleg [95] to give a randomized construction of an f–edge, f–vertex fault tolerant $(1+\varepsilon,\beta)$–spanner of size $O(f\beta(\frac{\beta}{\varepsilon})^{2f} n^{1+\frac{1}{k}} \log n))$ with high probability. Additionally, applying their result to the construction of Baswana et al. [26] gives an f–edge, f–vertex fault tolerant $(k+\varepsilon, k-1)$–spanner with $O(f(\frac{k-1}{\varepsilon})^{2f} n^{1+\frac{1}{k}} \log n)$ edges with high probability.

Bilò et al. [37] improve the constructions of other additive fault tolerant spanners in certain cases. For $f = 1$, they produce edge fault tolerant additive 2–spanners of size $O(n^{\frac{5}{3}})$, 4–spanners of size $O(n^{\frac{3}{2}})$, 10–spanners of size $\widetilde{O}(n^{\frac{7}{5}})$ (with high probability), and 14–spanners of size $O(n^{\frac{4}{3}})$.

Parter [132] studies clustering + path buying algorithms for producing vertex fault tolerant additive spanners and sourcewise spanners for a single fault vertex. The main results are to produce an additive 2–spanner of size $O(n^{\frac{5}{3}})$, a 4–additive sourcewise spanner of size $O(n|S| + (\frac{n}{|S|})^3)$, an additive 6–spanner of size $O(n^{\frac{3}{2}})$, and an 8–additive sourcewise spanner of size $O(n^{\frac{4}{3}})$ provided $|S| = O(n^{\frac{1}{3}})$.

Just as (non-faulty) BFS trees are frequently useful towards building non-faulty spanners, a corresponding notion of *fault tolerant BFS structures (FTBFS)* is useful in many of the above constructions of fault tolerant spanners. An f–edge or –vertex FTBFS is defined as an $S \times V$ distance preserver resilient to f edge or vertex faults, in the same sense as the above. These were introduced by Parter and Peleg [133], who showed that $O(|S|^{\frac{1}{2}} n^{\frac{3}{2}})$ edges are needed when $f = 1$, for edge or vertex faults, and this is tight in either setting. Subsequently, Parter [131] and Gupta and Khan [110] proved tight bounds of $O(|S|^{\frac{1}{3}} n^{\frac{5}{3}})$ for either setting with $f = 2$. For general f, however, there remains a gap: Parter

[131] proved a general lower bound of

$$\Omega(|S|^{\frac{1}{f+1}} n^{1-\frac{1}{f+1}})$$

for f edge or vertex faults, but the current best upper bound for general f is

$$O(|S|^{\frac{1}{2^f}} n^{1-\frac{1}{2^f}})$$

by [42]. It is a significant open question in the area to close this gap. There is also a related area of fault tolerant *reachability* trees, which must preserve reachability between all pairs in $S \times V$, not distance.

12.2. Resilient Spanners

[16] Giorgio Ausiello, Paolo Giulio Franciosa, Giuseppe F. Italiano, and Andrea Ribichini. On resilient graph spanners. *Algorithmica*, 74(4):1363–1385, 2016

Ausiello et al. [16] introduce the notion of *resilience* in graph spanners. Informally, a spanner is said to be *resilient* if the stretch factor is not increased much by deleting an edge from a spanner. Formally, if G is any graph, then the *fragility* of an edge e is defined by

$$\text{frag}_G(e) := \max_{u,v \in V} \frac{d_{G \setminus e}(u,v)}{d_G(x,y)}.$$

Given a graph G, $\sigma \geq 1$, $t \geq 1$, and a t–spanner G', an edge e is *σ–fragile* in G' if $\text{frag}_{G'}(e) > \max\{\sigma, \text{frag}_G(e)\}$, and the t–spanner G' is *σ–resilient* if every edge is *not* σ–fragile. That is, G' is σ–resilient provided

$$\text{frag}_{G'}(e) \leq \max\{\sigma, \text{frag}_G(e)\}, \quad e \in G'.$$

Resilience is a strong property, as the authors show that there is an infinite family of dense graphs which do not admit any 2–resilient spanners other than the graphs themselves. Moreover, resilience is a stronger notion than fault tolerance, and the authors show that there are 1–edge fault tolerant t–spanners containing edges with fragility at least $\frac{t^2}{2}$ for any $t \geq 3$. Additionally, a polynomial time algorithm is given for producing a σ–resilient $(2k-1)$–spanner (for $\sigma \geq 2k-1$ and $k \geq 2$) with $O(Wn^{\frac{3}{2}})$ edges, where $W = \frac{w_{\max}}{w_{\min}}$ with $w_{\max}$ being the largest weighted edge of G and $w_{\min}$ being the smallest weighted edge. The runtime for this algorithm is $O(mn + n^2 \log n)$. The authors also demonstrate that (α, β)–spanners can be turned into σ–resilient spanners for any $\sigma \geq \alpha + \beta$.

12.3. Dynamic Algorithms

[15] Giorgio Ausiello, Paolo Giulio Franciosa, and Giuseppe F. Italiano. Small stretch spanners on dynamic graphs. *Journal of Graph Algorithms and Applications*, 10(2):365–385, 2006. Announced at ESA'05

[22] Surender Baswana. Dynamic algorithms for graph spanners. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pages 76–87, 2006

[27] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic algorithms for graph spanners. *ACM Transactions on Algorithms*, 8(4):35:1–35:51, 2012

[43] Greg Bodwin and Sebastian Krinninger. Fully dynamic spanners with worst-case update time. In *24th Annual European Symposium on Algorithms (ESA)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016

[34] Aaron Bernstein, Sebastian Forster, and Monika Henzinger. A deamortization approach for dynamic spanner and dynamic maximal matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1899–1918, 2019

Given a spanner for a graph, dynamic algorithms attempt to maintain the properties of the spanner while edges are being added to or deleted from the initial graph. Thus, edges may need to be added or deleted in the spanner to maintain the distortion property. Dynamic spanners have so far been studied with multiplicative error; it is unclear at present whether this is coincidental or if there is a hardness barrier to obtaining other types of error.

The initial work on the problem was in [15], where the authors present algorithms for maintaining a 3– or 5–spanner of an input graph with essentially optimal size and update time proportional to the maximum degree. This update time is *amortized*, meaning that it holds on average over a sequence of insertions and deletions, but individual updates could take much longer. In [27], the authors present two algorithms for maintaining a sparse (multiplicative) t–spanner of an unweighted graph, again with optimal size (assuming the Girth Conjecture). The first algorithm achieves $O(7^{\frac{t}{4}})$ amortized update time (independent of the size of the graph n), and the second achieves $O(\text{polylog}n)$ amortized update time (independent of the stretch factor t).

Bodwin and Krinninger [43] address the problem of improving from *amortized* to *worst-case* update time, meaning that every individual update runs within the stated time with high probability. They provide randomized algorithms to maintain a 3–spanner with $\widetilde{O}(n^{1+\frac{1}{2}})$ edges with worst-case update time $\widetilde{O}(n^{\frac{3}{4}})$, or a 5–spanner with $\widetilde{O}(n^{1+\frac{1}{3}})$ edges with worst-case update time $\widetilde{O}(n^{\frac{5}{9}})$. Subsequently, [34] improved on these results by essentially converting the amortized construction of [27] to worst-case update time, with only minor changes to the construction parameters.

A notable open question in the area is to progress from *oblivious* to *non-oblivious* update time. That is: the update times in [43] and [34] hold with high probability against the randomness used in the algorithms, but only if the adversary choosing the graph updates is not allowed to see the random bits chosen by the algorithm. A *non-oblivious* construction would hold with high probability even if the adversary can base their updates on the random choices made by the construction. This is an important property because, if dynamic spanners are used as a subroutine in other graph algorithms, the next graph update may depend on the current state of the spanner, which thus requires non-obliviousness to keep the guarantees. The next step beyond non-obliviousness, of course, would be to obtain fully deterministic algorithms that maintain these spanners.

13. Spanners for Special Classes of Graphs

While the sparse/light spanner problem is typically stated for generic graphs, it has also been studied when the class of input graphs is restricted. In many cases, much stronger guarantees can be made for spanners. Here we highlight some of the literature in this vein, but we highlight the results rather than the techniques except where appropriate.

13.1. Geometric Spanners

[129] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007

The book [129] provides an extensive treatment of geometric spanners, so we do not cover them in this survey.

13.2. Directed Graphs

[140] Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 844–851. Society for Industrial and Applied Mathematics, 2002

[77] Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2011

[32] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Improved approximation for the directed spanner problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12. Springer, 2011

[80] Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840. SIAM, 2016

[155] Chun Jiang Zhu and Kam-Yiu Lam. Source-wise round-trip spanners. *Information Processing Letters*, 124:42–45, 2017

[156] Chun Jiang Zhu and Kam-Yiu Lam. Deterministic improved round-trip spanners. *Information Processing Letters*, 129:57–60, 2018

Dinitz and Krauthgamer study multiplicative spanners for directed graphs, which is more subtle than the undirected case as one must maintain connectivity of the graph when computing a spanner. Dinitz et al. [77] propose a flow based linear program formulation of the directed t–spanner problem, and give an approximation algorithm to find sparsest t–spanner for a given directed graph. Berman et al. [32] give a non-flow based formulation (an anti-spanner formulation) and improved the approximation of [77]. Dinitz et al. [80] provide a $\tilde{O}(n^{1/3})$–approximation for directed 4–spanner using a new rounding algorithm for the standard flow-based linear program. Roditty et al. [140] study roundtrip spanners, and Zhu and Lam [155, 156] introduce the notion of source-wise roundtrip spanners for directed graphs.

13.3. Further Reading

[70] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM (JACM)*, 47(1):132–166, 2000

[47] Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Minor-free graphs have light spanners. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 767–778. IEEE, 2017

14. Applications of Graph Spanners

Lastly, we survey some of the theoretical and practical applications of spanners in various fields.

Distributed Computing. A central challenge in distributed computing is message passing between vertices or nodes in an efficient manner. Naively, nodes could constantly broadcast everything they know to all their neighbors – thus propagating information around the system fairly quickly – but this requires high information throughput and processing. Often, a better idea is to build a sparse spanner of the network, greatly reducing the demands of sharing information at the price of only minor latency in propagation. This paradigm appears, for example, in communication networks in parallel computing [36], synchronizers [18], broadcasting [134], arrow distributed queuing protocols [51], wireless sensor networks [143], online load balancing [19], and motion planning in robotics control optimization [54]. In all of these applications, the quality of the spanner that can be built controls the above tradeoff between latency and communication complexity.

Network Routing. Another related class of applications arises in the task of passing messages throughout a network (this is related to distributed computing in some ways, but often generalized or abstracted differently in the literature). A classic example is where one wishes to pass packets or messages around the internet in a timely manner, but these "messages" can generally be construed quite broadly, e.g., as cars in a road network. The routing challenge is more involved than simply computing a spanner, as solutions must balance the efficiency of the chosen paths with the amount of information stored at each node and in the "packet header" being passed around the network. However, many modern routing algorithms exploit spanners as a useful step along the way. See [5, 140, 149, 56, 93] and references within for further information.

Computational Biology. To understand and model the history of organisms, Biologists have developed various ways to measure similarity between species, based either on their DNA sequence or on their level of interaction in an environment. But given a matrix of pairwise similarities, how can these be arranged into a graph that succinctly captures biological history? It turns out a good approach is to treat the matrix as a weighted graph (with all possible edges), and then build a spanner of the graph to determine its "most important" connections. This application has been used, for example, to measure genetic distance between contemporary species [21] and to visualize interactions between various proteins. [142]

Theoretical Applications. Since many graph algorithms have a runtime dependence on the number of edges in the input graph, one can often preprocess an input graph into a spanner in order to improve speed at the cost of a little accuracy. One area where this has been applied is in polynomial time approximation schemes (PTAS) for the traveling salesman problem [48]. Since improved spanners are often available on special graph classes, so too the PTAS can be improved. Some of these graph classes include planar graphs, bounded-genus graphs, unit disk graphs, and bounded path width graphs [93].

Another class of theoretical applications show that spanners efficiently capture the "backbone" of the network and thus often implicitly represent other important network properties besides distances [12]. This frequently includes the network spectrum, and accordingly spanners have been used to construct spectral sparsifiers [115].

Other miscellaneous applications of spanners include computing distances and shortest paths between points embedding in a geometric space [46, 59], testing graph properties (approximately) in sublinear time [65], the facility location problem [68], and key management in access control hierarchies [32]. Closer applications of spanners themselves include cycle covers of graphs [53], for which the extremal instances can often be decomposed into a union of tree spanners, and *labelling schemes* in which the vertices of a graph are labelled in such a way that one can (approximately) recover the distance between nodes by inspecting only their labels [26].

15. Conclusion

Since their advent, graph spanners have become an important object of study and have found a broad range of applications as well as a rich theory. The goal of this survey was to introduce readers to the overarching techniques that have been employed to compute various types of spanners, and to tabulate the state-of-the-art algorithmic bounds in an accessible way. Additionally, we have posed several open problems along the way which may be of interest to experts and non-experts alike. The literature on spanners continues to grow at a rapid pace, and is unlikely to stop in the near future. Nonetheless, it is hoped that this survey will provide a guiding reference for the state of the field for some time.

16. Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback which substantially improved the presentation of the material in this survey. Additionally, we take pleasure in thanking Michael Elkin for helpful comments, especially related to Sections 8 and 10.

17. References

References

[1] Amir Abboud and Greg Bodwin. Error amplification for pairwise spanner lower bounds. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 841–854. Society for Industrial and Applied Mathematics, 2016.

[2] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *Journal of the ACM (JACM)*, 64(4):28, 2017.

[3] Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1865–1883. Society for Industrial and Applied Mathematics, 2018.

[4] Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM Journal on Computing*, 47(6):2203–2236, 2018.

[5] Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. Ramsey spanning trees and their applications. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1650–1664. Society for Industrial and Applied Mathematics, 2018.

[6] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen Kobourov, and Richard Spence. Weighted additive spanners. *arXiv preprint arXiv:2002.07152*, 2020.

[7] Reyan Ahmed, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, Faryad Darabi Sahneh, and Richard Spence. Approximation algorithms and an integer program for multi-level graph spanners. In *Proceedings of the Special Event on Analysis of Experimental Algorithms*, 2019.

[8] Reyan Ahmed, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, Faryad Darabi Sahneh, and Richard Spence. Multi-level graph sketches via single-level solvers. *preprint, arXiv:1905.00536*, 2019.

[9] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 5–14, 2012.

[10] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28:1167–1181, 04 1999.

[11] 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.

[12] Eduardo Álvarez-Miranda and Markus Sinnl. Mixed-integer programming approaches for the tree t^*-spanner problem. *Optimization Letters*, pages 1–17, 2018.

[13] Sanjeev Arora and Carsten Lund. *Hardness of Approximations*, page 399–446. PWS Publishing Co., USA, 1996.

[14] Sunil Arya, Gautam Das, David M Mount, Jeffrey S Salowe, and Michiel Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC)*, pages 489–498, 1995.

[15] Giorgio Ausiello, Paolo Giulio Franciosa, and Giuseppe F. Italiano. Small stretch spanners on dynamic graphs. *Journal of Graph Algorithms and Applications*, 10(2):365–385, 2006. Announced at ESA'05.

[16] Giorgio Ausiello, Paolo Giulio Franciosa, Giuseppe F. Italiano, and Andrea Ribichini. On resilient graph spanners. *Algorithmica*, 74(4):1363–1385, 2016.

[17] Giorgio Ausiello, Andrea Ribichini, Paolo G Franciosa, and Giuseppe F Italiano. Computing graph spanners in small memory: fault-tolerance and streaming. *Discrete Mathematics, Algorithms and Applications*, 2(04):591–605, 2010.

[18] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

[19] Baruch Awerbuch, Shay Kutten, and David Peleg. Online load balancing in a distributed network. In *Proceedings of the 24th ACM Symposium on Theory of Computing (STOC)*, pages 571–580, 1992.

[20] Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 267–280, 2018.

[21] Hans-Jürgen Bandelt and Andreas Dress. Reconstructing the shape of a tree from observed dissimilarity data. *Advances in Applied Mathematics*, 7(3):309 – 343, 1986.

[22] Surender Baswana. Dynamic algorithms for graph spanners. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA)*, pages 76–87, 2006.

[23] Surender Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Information Processing Letters*, 106(3):110–114, 2008.

[24] Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 609–621. Springer, 2008.

[25] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)–spanners and purely additive spanners. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005.

[26] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):5, 2010.

[27] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic algorithms for graph spanners. *ACM Transactions on Algorithms*, 8(4):35:1–35:51, 2012.

[28] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a $(2k-1)-$spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Automata, Languages and Programming*, pages 384–396, 2003.

[29] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms (TALG)*, 2(4):557–577, 2006.

[30] Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007.

[31] Uri Ben-Levy and Merav Parter. New (α, β) spanners and hopsets. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1695–1714. SIAM, 2020.

[32] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Improved approximation for the directed spanner problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12. Springer, 2011.

[33] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed Steiner forest. *38th International Colloquium on Automata, Languages and Programming (ICALP)*, 222:93 – 107, 2013.

[34] Aaron Bernstein, Sebastian Forster, and Monika Henzinger. A deamortization approach for dynamic spanner and dynamic maximal matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1899–1918, 2019.

[35] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[36] Sandeep Bhatt, Fan Chung, Tom Leighton, and Arnold Rosenberg. Optimal simulations of tree machines. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 274–282, 1986.

[37] Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Proceedings of the 23rd European Symposium on Algorithms (ESA)*, pages 167–178. Springer, 2015.

[38] Greg Bodwin. Linear size distance preservers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 600–615. Society for Industrial and Applied Mathematics, 2017.

[39] Greg Bodwin. On the structure of unique shortest paths in graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2071–2089. Society for Industrial and Applied Mathematics, 2019.

[40] Greg Bodwin. Some general structure for extremal sparsification problems. *arXiv preprint arXiv:2001.07741*, 2020.

[41] Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1884–1900. Society for Industrial and Applied Mathematics, 2018.

[42] Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 73:1–73:14. European Association for Theoretical Computer Science, 2017.

[43] Greg Bodwin and Sebastian Krinninger. Fully dynamic spanners with worst-case update time. In *24th Annual European Symposium on Algorithms (ESA)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016.

[44] Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 541–543. ACM, 2019.

[45] 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 (SODA)*, pages 855–872. Society for Industrial and Applied Mathematics, 2016.

[46] Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Mathematics*, 19(4):1029–1055, 2005.

[47] Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Minor-free graphs have light spanners. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 767–778. IEEE, 2017.

[48] Glencora Borradaile, Hung Le, and Christian Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2371–2379. Society for Industrial and Applied Mathematics, 2019.

[49] Ulrik Brandes and Dagmar Handke. NP-completeness results for minimum planar spanners. *Discrete Mathematics and Theoretical Computer Science*, 3(1), 1998.

[50] Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault tolerant additive and (μ, α)–spanners. *Theoretical Computer Science*, 580:94–100, 2015.

[51] Costas Busch and Srikanta Tirthapura. Concurrent counting is harder than queuing. *Theoretical Computer Science*, 411(43):3823–3833, October 2010.

[52] Leizhen Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics*, 48(2):187–194, 1994.

[53] Leizhen Cai and Derek G Corneil. Tree spanners. *SIAM Journal on Discrete Mathematics*, 8(3):359–387, 1995.

[54] Leizhen Cai and J. Mark Keil. Computing visibility information in an inaccurate simple polygon. *Int. J. Comput. Geometry Appl.*, 7(6):515–538, 1997.

[55] Keren Censor-Hillel, Telikepalli Kavitha, Ami Paz, and Amir Yehudayoff. Distributed construction of purely additive spanners. In *International Symposium on Distributed Computing*, pages 129–142. Springer, 2016.

[56] Keren Censor-Hillel, Ami Paz, and Noam Ravid. The sparsest additive spanner via multiple weighted BFS trees. In *22nd International Conference on Principles of Distributed Systems*, 2019.

[57] T-H Hubert Chan, Michael Dinitz, and Anupam Gupta. Spanners with slack. In *European Symposium on Algorithms (ESA)*, pages 196–207. Springer, 2006.

[58] T-H Hubert Chan, Mingfei Li, Li Ning, and Shay Solomon. New doubling spanners: Better and simpler. *SIAM Journal on Computing*, 44(1):37–53, 2015.

[59] Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. In *Proceedings of the eighth annual Symposium on Computational Geometry*, pages 192–201. ACM, 1992.

[60] Hsien-Chih Chang, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Near-Optimal Distance Emulator for Planar Graphs. In *Proceedings of the 26th Annual European Symposium on Algorithms (ESA)*, volume 112, pages 16:1–16:17, 2018.

[61] Shiri Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512. Society for Industrial and Applied Mathematics, 2013.

[62] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.

[63] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Transactions on Algorithms (TALG)*, 14(3):33, 2018.

[64] L Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[65] Eden Chlamtáč and Michael Dinitz. Lowest-degree k-spanner: Approximation and hardness. *Theory of Computing*, 12(15):1–29, 2016.

[66] Eden Chlamtáč, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed Steiner forest: Upper and lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 534–553. SIAM, 2017.

[67] Eden Chlamtáč, Michael Dinitz, and Robert Krauthgamer. Everywhere-sparse spanners via dense subgraphs. *In 16th Annual Symposium on Foundations of Computer Science*, 05 2012.

[68] Keerti Choudhary and Omer Gold. Diameter spanner, eccentricity spanner, and approximating extremal graph distances: Static, dynamic, and fault tolerant. *preprint, arXiv:1812.01602*, 2018.

[69] Keerti Choudhary and Omer Gold. Extremal distances in directed graphs: Tight spanners and near-optimal approximation algorithms. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 495–514. SIAM, 2020.

[70] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM (JACM)*, 47(1):132–166, 2000.

[71] Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

[72] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009.

[73] Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On Pairwise Spanners. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20, pages 209–220, 2013.

[74] Bilel Derbel, Cyril Gavoille, and David Peleg. Deterministic distributed construction of linear stretch spanners in polylogarithmic time. In *International Symposium on Distributed Computing*, pages 179–192. Springer, 2007.

[75] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing*, pages 273–282. ACM, 2008.

[76] Michael Dinitz, Guy Kortsarz, and Ran Raz. Label cover instances with large girth and the hardness of approximating basic k-spanner. *ACM Transactions on Algorithms (TALG)*, 12(2):25, 2016.

[77] Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2011.

[78] 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 (PODC)*, pages 169–178, 2011.

[79] Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. Lasserre integrality gaps for graph spanners and related problems. *arXiv preprint arXiv:1905.07468*, 2019.

[80] Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840. SIAM, 2016.

[81] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

[82] Feodor F Dragan, Fedor V Fomin, and Petr A Golovach. Spanners in sparse graphs. *Journal of Computer and System Sciences*, 77(6):1108–1119, 2011.

[83] Michael Elkin. personal communication.

[84] Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms (TALG)*, 1(2):283–323, 2005.

[85] Michael Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms (TALG)*, 7(2):1–17, 2011.

[86] Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theoretical Computer Science*, 697:1–36, 2017.

[87] Michael Elkin, Arnold Filtser, and Ofer Neiman. Distributed construction of light networks. *arXiv preprint arXiv:1905.02592*, 2019.

[88] Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and pram distance oracles in weighted graphs. *arXiv preprint arXiv:1907.11422*, 2019.

[89] Michael Elkin and Shaked Matar. Near-additive spanners in low polynomial deterministic congest time. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 531–540, 2019.

[90] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Transactions on Algorithms (TALG)*, 15(1):1–29, 2018.

[91] Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in rnc. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 333–341, 2019.

[92] Michael Elkin and Ofer Neiman. Near-additive spanners and near-exact hopsets, a unified view. *arXiv preprint arXiv:2001.07477*, 2020.

[93] Michael Elkin, Ofer Neiman, and Shay Solomon. Light spanners. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 442–452. Springer, 2014.

[94] Michael Elkin and David Peleg. Strong inapproximability of the basic k-spanner problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 636–648. Springer, 2000.

[95] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$–spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004.

[96] Michael Elkin and David Peleg. Approximating k–spanner problems for $k > 2$. *Theoretical Computer Science*, 337(1-3):249–277, 2005.

[97] Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory of Computing Systems*, 41(4):691–729, Dec 2007.

[98] Michael Elkin and Shay Solomon. Optimal Euclidean spanners: Really short, thin, and lanky. *Journal of the ACM (JACM)*, 62(5):1–45, 2015.

[99] Michael Elkin and Shay Solomon. Fast constructions of lightweight spanners for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(3):29, 2016.

[100] Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\varepsilon, \beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006.

[101] Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM Journal on Computing*, 38(5):1761–1781, 2008.

[102] P. Erdős and M. Simonovits. Some extremal problems in graph theory. In *Combinatorial Theory and its Applications, I (Proc. Colloq., Balatonfüred, 1969)*, pages 377–390. North-Holland, Amsterdam, 1970.

[103] Paul Erdős. Extremal problems in graph theory. In *Proceedings of the Symposium on Theory of Graphs and its Applications*, page 2936, 1963.

[104] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Departmental Papers (CIS)*, page 236, 2005.

[105] Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 9–17. ACM, 2016.

[106] Kshitij Gajjar and Jaikumar Radhakrishnan. Distance-preserving subgraphs of interval graphs. In *25th Annual European Symposium on Algorithms (ESA)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2017.

[107] Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.

[108] Lee-Ad Gottlieb. A light metric spanner. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 759–772. IEEE, 2015.

[109] Joachim Gudmundsson, Giri Narasimhan, and Michiel Smid. *Geometric Spanners*, pages 360–364. Springer US, Boston, MA, 2008.

[110] Manoj Gupta and Shahbaz Khan. Multiple Source Dual Fault Tolerant BFS Trees. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 80, pages 127:1–127:15, 2017.

[111] S. Halperin and U. Zwick. Unpublished result, 1996.

[112] Dagmar Handke and Guy Kortsarz. Tree spanners for subgraphs and related tree covering problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 206–217. Springer, 2000.

[113] Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. In *16th Scandinavian Symposium and Workshops on Algorithm Theory*, 2018.

[114] Shang-En Huang and Seth Pettie. Thorup–Zwick emulators are universally optimal hopsets. *Information Processing Letters*, 142:9–13, 2019.

[115] Michael Kapralov and David Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, pages 272–281. ACM, 2014.

[116] Telikepalli Kavitha. New pairwise spanners. *Theory of Computing Systems*, 61(4):1011–1036, 2017.

[117] Telikepalli Kavitha and Nithin M Varma. Small stretch pairwise spanners. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 601–612. Springer, 2013.

[118] Philip N. Klein. A subset spanner for planar graphs, with application to subset tsp. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 749–756, New York, NY, USA, 2006. ACM.

[119] Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. In *Scandinavian Workshop on Algorithm Theory*, pages 277–281. Springer, 2014.

[120] Mathias Bæk Tejs Knudsen. Additive spanners and distance oracles in quadratic time. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2017.

[121] Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theoretical Computer Science*, 746:88–97, 2018.

[122] Yusuke Kobayashi. An FPT algorithm for minimum additive spanner problem. *preprint, arXiv:1903.01047*, 2019.

[123] Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM J. Comput.*, 27:1438–1456, 1994.

[124] Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *J. Algorithms*, 17(2):222–236, 1994.

[125] Christoph Lenzen and David Peleg. Efficient distributed source detection with limited bandwidth. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, pages 375–382. ACM, 2013.

[126] Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 186–195. ACM, 1998.

[127] Arthur Liestman and Thomas Shermer. Additive graph spanners. *Networks*, 23:343 – 363, 07 1993.

[128] Gary L Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 192–201. ACM, 2015.

[129] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007.

[130] Merav Parter. Bypassing Erdős' girth conjecture: Hybrid stretch and sourcewise spanners. In *Automata, Languages, and Programming*, pages 608–619. Springer, 2014.

[131] Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 481–490. ACM, 2015.

[132] Merav Parter. Vertex fault tolerant additive spanners. *Distributed Computing*, 30(5):357–372, October 2017.

[133] Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *European Symposium on Algorithms (ESA)*, pages 779–790. Springer, 2013.

[134] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.

[135] David Peleg. Distributed computing. *SIAM Monographs on Discrete Mathematics and Applications*, 5, 2000.

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

[137] David Peleg and Jeffrey D Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18(4):740–747, 1989.

[138] Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. In *Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing*, pages 253–262. ACM, 2008.

[139] Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6(1):7, 2009.

[140] Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 844–851. Society for Industrial and Applied Mathematics, 2002.

[141] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Automata, Languages and Programming*, pages 261–272, 2005.

[142] Daniel Russel and L Guibas. Exploring protein folding trajectories using geometric spanners. In *Biocomputing 2005*, pages 40–51. World Scientific, 2005.

[143] H. Shpungin and M. Segal. Near optimal multicriteria spanner constructions in wireless ad-hoc networks. In *IEEE INFOCOM 2009*, pages 163–171, April 2009.

[144] Mikkel Sigurd and Martin Zachariasen. Construction of minimum-weight spanners. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA)*, pages 797–808, 2004.

[145] Kavita Singh and Shyam Sundar. Artifical bee colony algorithm using problem-specific neighborhood strategies for the tree t-spanner problem. *Applied Soft Computing*, 62:110–118, 2018.

[146] Christian Sommer. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)*, 46(4):45, 2014.

[147] Shyam Sundar. A steady-state genetic algorithm for the tree t-spanner problem. In *Soft Computing: Theories and Applications*, pages 387–398. Springer, 2019.

[148] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, 2001.

[149] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. Association of Computing Machinery, 2001.

[150] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 802–809. Society for Industrial and Applied Mathematics, 2006.

[151] Jacques Tits. Sur la trialité et certains groupes qui s'en déduisent. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 2(1):14–60, 1959.

[152] R Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *J. Combin. Theory Ser. B*, 52(1):113–116, 1991.

[153] David P Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 389–398. IEEE, 2006.

[154] David P Woodruff. Additive spanners in nearly quadratic time. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 463–474. Springer, 2010.

[155] Chun Jiang Zhu and Kam-Yiu Lam. Source-wise round-trip spanners. *Information Processing Letters*, 124:42–45, 2017.

[156] Chun Jiang Zhu and Kam-Yiu Lam. Deterministic improved round-trip spanners. *Information Processing Letters*, 129:57–60, 2018.

Appendix A. Tables of Spanner Guarantees

Stretch (t)	Size: $O(\|E'\|)$	Weight: $O(W(E'))$	Time $O(\cdot)$	Reference
3	$O(n^{\frac{2}{3}})$	**	$\log n$	[74]
$2k-1$	$n^{1+\frac{1}{k}}$	$(1+\frac{n}{2k})$	$m(n^{1+\frac{1}{k}}+n\log n)$	[11]
$2k-1$	$n^{1+\frac{1}{k}}$	$kn^{1+\frac{1}{k}}$	km	[28, 30]
$2k-1$	$O(kn^{1+\frac{1}{k}})$	**	**	[75]
$2k-1$	$O(\min(m, kn^{1+\frac{1}{k}}))$	**	m	[23]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{1+\frac{1}{k}}$	$m(n^{1+\frac{1}{k}}+n\log n)$	[59]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{1+\frac{1}{k}}$	$n^{\frac{1}{k}}\left(1+\frac{k}{\varepsilon^{1+\frac{1}{k}}\log k}\right)$	[93]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{2+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{2+\frac{1}{k}}$	$kn^{2+\frac{1}{k}}$	[99]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}\left(k+\frac{1}{\varepsilon}\right)^{2+\frac{1}{k}}$	$kn^{\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{2+\frac{1}{k}}$	$km+\min\{n\log n, m\alpha(n)\}$	[99]
$(2k-1)(1+\varepsilon)$	$n^{1+\frac{1}{k}}$	$n^{1+\frac{1}{k}}\left(\frac{1}{\varepsilon}\right)^{2+\frac{1}{k}}$	**	[63]

Table A.2: Guarantees for Multiplicative Spanners. Here, $n=|V|$, $m=|E|$, $\alpha(n)$ is the inverse Ackermann function, and all weight bounds are multiplied by $W(\mathrm{MST}(G))$. Complexities not given in the paper are denoted by **.

Additive Error (β)	Size: $O(\|E'\|)$	Time $O(\cdot)$	Reference
2	$\widetilde{O}(n^{\frac{3}{2}})$	$O(n^{\frac{5}{2}}\sqrt{\log n})$	[10]
2	$n^{\frac{3}{2}}$	$n^{\frac{5}{2}}$	[10]
2	$n^{\frac{3}{2}}\log^{\frac{1}{2}} n$	$n^2\log^2 n$	[81]
2	$n^{\frac{3}{2}}$	n^2	[120]
4	$O(n^{\frac{7}{5}}\log^{\frac{4}{5}}(n))$	$n^{\frac{3}{5}}\log^{\frac{1}{5}} n+D$	[55]
4	$\widetilde{O}(n^{\frac{7}{5}})$	**	[61]
6	$n^{\frac{4}{3}}$	$mn^{\frac{2}{3}}$	[26]
6	$n^{\frac{4}{3}}\log^3 n$	$n^2\log^2 n$	[154]
8	$n^{\frac{4}{3}}$	n^2	[154]
$(1, 2k+4\ell)$	$\Gamma_k(G)+n^{1+\frac{1}{k+\ell+1}}$	$mn^{1-\frac{l}{k+l+1}}$	[26]
$(1, O(\sqrt{d(u,v)}))$	$\widetilde{O}(n^{1+\frac{3}{17}})$	**	[61]

Table A.3: Guarantees for Additive Spanners. Here, $n=|V|$, $m=|E|$. Complexities not given in the paper are denoted by **. The term D is the diameter of the graph.

(α, β)	Size: $O(\lvert E' \rvert)$	Time $O(\cdot)$	Reference
$(k-1, 2k-O(1))$	$kn^{1+\frac{1}{k}}$	$mn^{1-\frac{1}{k}}$	[95]
$(k, k-1)$	$kn^{1+\frac{1}{k}}$	km	[26]
$(1+\epsilon, 4)$	$\epsilon^{-1}n^{\frac{4}{3}}$	$mn^{\frac{2}{3}}$	[95]
$(1+\epsilon, \beta)$	$\beta n^{1+\frac{1}{k}}$	$n^{2+\frac{1}{t}}$	[95], $\beta = \beta(k, \epsilon, t)$
$(1+\epsilon, \beta')$	$\beta' n^{1+\frac{1}{k}}$	mn^{ρ}	[84], $\beta' = \beta'(k, \epsilon, \rho)$
$(1+\epsilon, \beta'')$	$kn^{1+\frac{1}{k}}$	$kmn^{\frac{1}{k}}$	[150], $\beta'' = \beta''(k, \epsilon)$

Table A.4: Guarantees for (α, β)–spanners. Here, $n = |V|$, $m = |E|$. In [84], $\rho > \frac{1}{2k}$ is required.

Spanner	(α, β)	Size: $O(\lvert E' \rvert)$	Reference
Pairwise	$(1+\varepsilon, 4)$	$n\lvert P \rvert^{\frac{1}{4}}\sqrt{\log \frac{n}{\varepsilon}}$	[73]
Pairwise	$(1, 4k)$	$n^{1+\frac{1}{2k+1}}((4k+5)\lvert P \rvert)^{\frac{k}{4k+2}}$	[73]
Pairwise	$(1, 2)$	$n\lvert P \rvert^{\frac{1}{4}}$	[116]
Pairwise	$(1, 2)$	$O(n\lvert P \rvert^{\frac{1}{3}})$	[1]
Pairwise	$(1, 2)$	$\widetilde{O}(n\lvert P \rvert^{\frac{1}{3}})$	[117]
Pairwise	$(1, 2)$	$O(n\lvert P \rvert^{\frac{1}{3}}\log^{\frac{2}{3}}(n))$	[55]
Pairwise	$(1, 4)$	$\widetilde{O}(n\lvert P \rvert^{\frac{2}{7}})$	[116]
Pairwise	$(1, 6)$	$n\lvert P \rvert^{\frac{1}{4}}$	[116]
Sourcewise	$(1, 2)$	$\widetilde{O}(n(n\lvert S \rvert)^{\frac{1}{4}})$	[117]
Sourcewise	$(1, 2)$	$O(n^{\frac{5}{4}}\lvert S \rvert^{\frac{1}{4}}\log^{\frac{3}{4}}(n))$	[55]
Sourcewise	$(1, 2)$	$\widetilde{O}(n(n\lvert S \rvert)^{\frac{1}{4}})$	[117]
Sourcewise	$(1, 4)$	$\widetilde{O}(n(n\lvert S \rvert)^{\frac{2}{9}})$	[116]
Sourcewise	$(1, 6)$	$\widetilde{O}(n(n\lvert S \rvert)^{\frac{1}{5}})$	[116]
Sourcewise	$(1, 2k), \forall k \geq 1$	$\widetilde{O}(n(n\lvert S \rvert^{k})^{1/(2k+2)})$	[130]
Sourcewise $(S \times V)$	$(1, 2k)$	$n^{1+\frac{1}{2k+1}}(k\lvert S \rvert)^{\frac{k}{2k+1}}$	[73]
Subsetwise	$(1, 2)$	$n\sqrt{\lvert S \rvert}$	[73, 83, 139]
Subsetwise	$(1, 2)$	$O(n\lvert S \rvert^{\frac{2}{3}}\log^{\frac{2}{3}}(n))$	[55]
Source-Target	$(1, 4)$	$n(\lvert S \rvert\lvert T \rvert)^{\frac{1}{4}}$	[116]
Source-Target	$(1, 2k), \forall k \geq 1$	$n(\lvert S \rvert^{k+1}\lvert T \rvert)^{1/(2k+3)}$	[116]

Table A.5: Guarantees for pairwise and subsetwise spanners. In [118], an algorithm has been provided to compute multiplicative subsetwise spanner with constant approximation ratio for planar graphs.

Fault (f, E/V)	(α, β)	Size: $O(\lvert E' \rvert)$	Time $O(\cdot)$	Reference
f, V	$(2k-1, 0)$	$f^2 k^{f+1} n^{1+\frac{1}{k}} \log^{1-\frac{1}{k}} n$	$f^2 k^{f+2} n^{3+\frac{1}{k}} \log^{1-\frac{1}{k}} n$	[62]*
f, E	$(2k-1, 0)$	$f n^{1+\frac{1}{k}}$	$f^2 k^{f+2} n^{3+\frac{1}{k}} \log^{1-\frac{1}{k}} n$	[62]*
f, V	$(2k-1, 0)$	$f^{2-\frac{1}{k}} n^{1+\frac{1}{k}} \log n$	**	[78]
f, V	$(2k-1, 0)$	$f^{1-\frac{1}{k}} n^{1+\frac{1}{k}} 2^{O(k)}$	**	[41]
f, V	$(2k-1, 0)$	$f^{1-\frac{1}{k}} n^{1+\frac{1}{k}}$	**	[44]
f, V	$(3, 2)$	$f^{\frac{4}{3}} n^{\frac{4}{3}}$	$\widetilde{O}(f^2 m)$	[17]
f, V	$(2, 1)$	$f n^{\frac{3}{2}}$	$\widetilde{O}(fm)$	[17]
f, E, V	$(1+\varepsilon, \beta)$	$f\beta(\frac{\beta}{\varepsilon})^{2f} n^{1+\frac{1}{k}} \log n$	**	[50]*
f, E, V	$(k+\varepsilon, k-1)$	$f(\frac{k-1}{\varepsilon})^{2f} n^{1+\frac{1}{k}} \log n$	**	[50]*
1, E	$(1, 2)$	$n^{\frac{5}{3}}$	**	[37]
1, E	$(1, 4)$	$n^{\frac{3}{2}}$	**	[37]
1, E	$(1, 10)$	$\widetilde{O}(n^{\frac{7}{5}})$	**	[37]*
1, E	$(1, 14)$	$n^{\frac{4}{3}}$	**	[37]
1, V	$(1, 2)$	$n^{\frac{5}{3}}$	**	[132]
1, V	$(1, 6)$	$n^{\frac{3}{2}}$	**	[132]
f, E, V	$(2k-1, 0)$	$f^{1-\frac{1}{k}} n^{1+\frac{1}{k}} 2^{O(k)}$	**	[41]
f, E, V	$(2k-1, 0)$	$f^{1-\frac{1}{k}} n^{1+\frac{1}{k}}$	**	[44]

Table A.6: Guarantees for Fault Tolerant Spanners. Here, $n = |V|$, $m = |E|$, f is the number of faults allowed, and E or V in the first column denotes if the guarantee is for edge-fault or vertex-fault tolerance. Complexities not given in the paper are denoted by **. Spanners constructed with high probability are denoted by * next to the citation.