Additive Spanners and (α, β)-Spanners*

Surender Baswana† Telikepalli Kavitha‡ Kurt Mehlhorn§ Seth Pettie¶

Abstract

An (α, β)-spanner of an unweighted graph G is a subgraph H that distorts distances in G up to a multiplicative factor of α and an additive term β. It is well known that any graph contains a (multiplicative) $(2k-1, 0)$-spanner of size $O(n^{1+1/k})$ and an (additive) $(1, 2)$-spanner of size $O(n^{3/2})$. However no other additive spanners are known to exist.

In this paper we develop a couple of new techniques for constructing (α, β)-spanners. Our first result is an additive $(1, 6)$-spanner of size $O(n^{4/3})$. The construction algorithm can be understood as an economical agent that assigns *costs* and *values* to paths in the graph, *purchasing* affordable paths and ignoring expensive ones, which are intuitively well-approximated by paths already purchased. We show that this *path buying* algorithm can be parameterized in different ways to yield other sparseness-distortion tradeoffs. Our second result addresses the problem of which (α, β)-spanners can be computed efficiently, ideally in linear time. We show that for any k, a $(k, k-1)$-spanner with size $O(kn^{1+1/k})$ can be found in linear time, and further, that in a distributed network the algorithm terminates in a constant number of rounds. Previous spanner constructions with similar performance had roughly twice the multiplicative distortion.

1 Introduction

An (α, β)-spanner of an undirected graph G is a *subgraph* H such that for all vertices u, v:

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

where δ_G is the distance in graph G. In other words, an (α, β)-spanner guarantees that for pairs of vertices far apart in G, their distance in the spanner is stretched by roughly an α factor, which would ideally be close to 1. We call a $(1, \beta)$-spanner an *additive* β-spanner. If $\beta = 0$ this definition reverts to the usual definition of a *multiplicative* α-spanner [46, 9].

Spanners (and related structures) are useful in many contexts. They are the basis of space-efficient routing tables that guarantee nearly shortest routes [4, 53, 49, 23, 24, 48], schemes for simulating synchronized protocols in unsynchronized networks [47], and parallel and distributed algorithms for computing approximate shortest paths [20, 21, 27]. A recent application of spanners is the construction of labeling schemes and distance oracles [54, 14, 50, 11], which are data structures that can report approximately accurate distances in constant time. In all of these applications the quality of the solution ultimately depends on an efficient algorithm for computing a *low distortion* sparse spanner. The main open problem in this area is to understand the inherent tradeoffs between these three measures of efficiency: distortion (α and β), sparseness, and construction time. Even ignoring construction time, there are only a handful of cases where the distortion-sparseness tradeoff is fully understood.

*Partially supported by the Future and Emerging Technologies program of the EU under contract number IST-1999-14186 (ALCOM-FT).

†Email: sbaswana@cse.iitk.ac.in
‡Email: kavitha@csa.iisc.ernet.in
§Email: mehlhorn@mpi-inf.mpg.de
¶Email: pettie@umich.edu.

Multiplicative Spanners. The early work on spanners established the basic tradeoff between sparseness and *multiplicative* distortion. If the spanner size is fixed at $O(n^{1+1/k})$ the multiplicative distortion can be no better than $\Theta(k)$ [46]. We let n and m be the number of vertices and edges in the input graph. Althöfer et al. [9] proposed a greedy algorithm for producing an $(2k-1)$-spanner whose size is at most $m_{2k+1}(n)$, where $m_g(n)$ is the maximum number of edges in a graph with girth at least g.[1] Moreover, they observed that $m_{2k+1}(n)$ is *precisely* the best possible bound for a $(2k-1)$-spanner. If one removes any edge from a graph with girth $2k+1$ the distance between its endpoints jumps from 1 to at least $2k$. Thus, the only $(2k-1)$-spanner of such a graph is the graph itself. A trivial upper bound on $m_{2k+1}(n)$ and $m_{2k+2}(n)$ is $O(n^{1+1/k})$. It has been conjectured, by Erdős [34] and others that this bound is asymptotically tight, though the conjecture has only been proved for $k = 1, 2, 3$, and 5; weaker lower bounds are known for all other k; see [56, 54]. In other words, finding the exact tradeoff between sparseness and *multiplicative* distortion is at least as hard as proving or disproving the girth conjecture.

The fastest implementations of the Althöfer et al. algorithm run in time $O(\min\{kn^{2+1/k}, mn^{1+1/k}\})$ [9, 51], though there are several more efficient $(2k-1)$-spanner constructions. Halperin and Zwick [38, 45] compute an $O(n^{1+1/k})$-size $(2k-1)$-spanner in linear time. However, unlike the algorithm of Althöfer et al., the Halperin-Zwick algorithm only works on unweighted graphs. For weighted graphs Baswana and Sen [12] give a randomized construction of such a spanner with size $O(kn^{1+1/k})$. The Baswana-Sen algorithm has since been derandomized by Roditty et al. [50].

Beyond Purely Multiplicative Distortion. The *girth* bound exactly characterizes the optimal tradeoff between sparseness and multiplicative distortion but arguments based on girth only apply to *adjacent* vertices. In unweighted graphs the girth argument could just as easily be interpreted as bounding the additive distortion, or some combination of additive and multiplicative distortion. In particular, if the girth conjecture is true we can only say that an (α, β)-spanner of size $O(n^{1+1/k})$ has $\alpha + \beta \geq 2k-1$. It is conceivable that there exist additive $(2k-2)$-spanners with size $O(n^{1+1/k})$, for any k. Before our work, however, only one such additive spanner was known. Aingworth et al. [6] (with followup work in [25, 30, 55]) showed that there exist $O(n^{3/2})$-size additive 2-spanners. On the lower bound side, Woodruff [57] recently proved that any spanner with size $O(k^{-1}n^{1+1/k})$ cannot do better than an additive distortion of $2k-2$, *independent* of whether the girth conjecture is true or not.

The current research trend is to optimize distortion as a function of the distance being approximated, rather than fixate on adjacent vertices and the girth conjecture. Elkin and Peleg [29] showed that the girth bound (on multiplicative distortion) fails to hold even for vertices at distance 2. They gave a construction for $(k-1, 2k-O(1))$-spanners with size $O(kn^{1+1/k})$, with a number of refinements for short distances. They also showed [30] that for any $k \geq 2, \epsilon > 0$, there exist $(1+\epsilon, \beta)$-spanners with size $O(\beta n^{1+1/k})$, where $\beta = k^{\log\log k - \log \epsilon}$ is independent of n. In other words, the size can be driven close to linear and the multiplicative stretch close to 1, at the cost of a large additive term in the distortion. Thorup and Zwick [55] give a sparseness-distortion tradeoff that is in some ways stronger than Elkin and Peleg's. Their $(1+\epsilon, \beta)$-spanners have size $O(kn^{1+1/k})$ and $\beta = O(\lceil 1 + 2/\epsilon \rceil^{k-2})$, where ϵ plays no role in the construction and can be chosen as a function of the distance d being approximated. For $\epsilon^{-1} = d^{1/(k-1)}$, the spanner has additive distortion $O(d^{1-1/(k-1)} + 2^k)$. That is, the multiplicative distortion tends to 1 as the distance increases, whereas the Elkin-Peleg spanners tend to $1+\epsilon$, for an ϵ chosen a priori. See Figure 1 for a summary of existing spanners constructions.

Our Results. Our first result is that every graph contains an additive 6-spanner with size $O(n^{4/3})$ and that such a spanner can be computed efficiently. This result is a far cry from a full spectrum of tradeoffs between sparseness and additive distortion. However, our approach is completely new and is generic enough to be applied in other ways. We view a spanner construction as an economic agent that assigns a *cost* and *value* to paths in the graph. Affordable paths are *purchased* (included in the spanner) and expensive ones ignored. Different cost and value combinations lead to spanners with different properties. Besides constructing a 6-spanner, our path buying algorithm can be parameterized to find an additive 2-spanner of size $O(n^{3/2})$, matching [6, 25, 30, 55], and an additive $(n^{1-3\delta})$-spanners of size $O(n^{1+\delta})$, for any constant

[1] Girth is the length of the shortest cycle. Note that since every graph has a bipartite subgraph with at least half the edges, $\frac{1}{2}m_{2k+1}(n) \leq m_{2k+2}(n) \leq m_{2k+1}(n)$.

WEIGHTED GRAPHS, MULTIPLICATIVE SPANNERS

α	**Size**	**Time**	**Notes**
$2k-1$	$\frac{1}{2}n^{1+1/k}$	$O(mn^{1+1/k})$	[9, 7]
	$\frac{1}{2}n^{1+1/k}$	$O(kn^{2+1/k})$	[51, 7]
	$O(kn^{1+1/k})$	$O(km)$ (rand.)	[12]
	$O(kn^{1+1/k})$	$O(km)$	[50]

UNWEIGHTED GRAPHS, (α, β)-SPANNERS

(α, β)	**Size**	**Time**	**Notes**
$(2k-1,\, 0)$	$n^{1+1/k}$	$O(m)$	[38]
$(k-1,\, 2k-O(1))$	$O(kn^{1+1/k})$	$O(mn^{1-1/k})$	[29]
$(k,\, k-1)$	$O(kn^{1+1/k})$	$O(km)$	**new**
$(1+\epsilon,\, 4)$	$O(\epsilon^{-1}n^{4/3})$	$O(mn^{2/3})$	[30]
$(1+\epsilon,\, \beta)$	$O(\beta n^{1+1/k})$	$O(n^{2+1/t})$	[30], $\beta = \beta(k, \epsilon, t)$
$(1+\epsilon,\, \beta')$	$O(\beta' n^{1+1/k})$	$O(mn^{\rho})$	[27], $\beta' = \beta'(k, \epsilon, \rho)$
$(1+\epsilon,\, \beta'')$	$O(kn^{1+1/k})$	$O(kmn^{1/k})$	[55], $\beta'' = \beta''(k, \epsilon)$
$(1, n^{1-2\delta})$	$O(n^{1+\delta})$	poly(n)	[15], $\delta = \Theta(1)$
$(1, n^{1-3\delta})$	$O(n^{1+\delta})$	poly(n)	**new**, $\delta = \Theta(1)$
$(1,\, 2)$	$O(n^{3/2})$	$O(m\sqrt{n})$	[30, 6, 55]
	$\tilde{O}(n^{3/2})$	$\tilde{O}(n^2)$	[25]
$(1,\, 6)$	$O(n^{4/3})$	$O(mn^{2/3})$	**new**

Figure 1: State-of-the-art in (α, β)-spanners. The parameter $k \geq 2$ is always an integer. In the $(1+\epsilon, \beta)$-Spanner of [30], β is roughly $k^{\max\{\log\log k - \log\epsilon, \log t, 3\}}$. In [27] it is required that $\rho > 1/2k$; the expression for β' here is quite complicated. In [55] $\beta'' = 2\lceil 1 + 2/\epsilon \rceil^{k-2}$; however the construction and spanner are independent of ϵ, meaning it works for all ϵ simultaneously. Some slower spanner constructions are omitted from the figure.

$\delta \in (0, 1/3)$. The latter result improves the sparseness of [15] by a polynomial factor. We can also show that graphs with high girth (or, in general, those with few edges on short cycles) contain sparse additive spanners. For example, graphs with girth greater than 4 have 4-, 8-, and 12-spanners with sizes on the order of $n^{4/3}, n^{5/4}$, and $n^{6/5}$. Other examples are given in Figure 4 in Section 2.1.

Our second result addresses those sparseness-distortion tradeoffs that can be computed by an *efficient* algorithm. We show that a $(k, k-1)$-spanner of size $O(kn^{1+1/k})$ can be constructed in $O(km)$ time. Since the decisions made by the algorithm are very local, it can easily be implemented in modern models of computation. For instance, in the cache-oblivious model [36], the PRAM model [42], or in a synchronized distributed network [45], our algorithm is close to optimal under the relevant measures. Previous spanners with equal or better distortion [29, 30, 27, 55] have construction times of the form $O(mn^{\Omega(1)})$ and the result that is most comparable to ours, Elkin and Peleg's $(k-1, 2k-O(1))$-spanner, requires time $O(mn^{1-1/k})$ to compute. If we restrict our attention to linear or near-linear time constructions, all the existing spanners with size $O(kn^{1+1/k})$ [38, 12, 50] had multiplicative distortion $2k-1$. Whereas our $(k, k-1)$-spanners can be computed in $O(k)$ rounds in a distributed network, all previous constructions with equal or better distortion required $\Omega(n^{\Omega(1)})$ rounds [32].

1.1 Related Work

Spanners are part of a large body of work on *metric embeddings* [41, 40], where one wants a mapping $\phi : S \to T$ from a given (finite) source metric[2] (S, δ_S) to a target metric (T, δ_T) that does not distort interpoint distances by too much. (Distortion here is usually defined as purely multiplicative distortion.) In our case (S, δ_S) is some unweighted graph metric and ϕ is the identity function; the problem is to find

[2]Recall that (S, δ_S) is a metric if for $u, v, w \in S$, $\delta_S(u, u) = 0$, $\delta_S(u, v) = \delta(v, u) \geq 0$, and $\delta_S(u, v) \leq \delta_S(u, w) + \delta_S(w, v)$.

a metric (T, δ_T) corresponding to a sparse subgraph. We only survey metric embeddings where the target metric is some kind of graph.

Roditty, Thorup, and Zwick [49] construct multiplicative spanners for directed graphs under the *roundtrip* metric, where the distance between two vertices is the length of the shortest cycle (not necessarily simple) that contains both. The sparseness-distortion tradeoffs in [49] are slightly worse than those obtained for undirected graphs. Bollobás et al. [15] and Coppersmith and Elkin [22] studied spanners that preserve, without distortion, the distance between some pairs of vertices. In [15] it is shown that there exist $O(n^{1+\delta})$-size spanners that preserve distances greater than $n^{1-\delta}$, and that this tradeoff is optimal. In [22] it is shown that for any set P of pairs of vertices, there is a distance preserver for P with size $O(n + |P|\sqrt{n})$ edges, and that this bound is optimal in some circumstances. In particular, if $|P| = \omega(\sqrt{n})$ then a size of $O(n)$ cannot be guaranteed. Dor, Halperin, and Zwick [25] considered *emulators*, which may contain both graph edges and additional weighted edges, and observed that there are additive 4-emulators with $O(n^{4/3})$ edges. Thorup and Zwick [55] generalized this construction to emulators with $O(kn^{1+\frac{1}{2^k-1}})$ edges and additive distortion $O(kd^{1-\frac{1}{k-1}})$, where d is the distance being approximated. One well known application of (multiplicative) spanners is in the construction of approximate distance oracles; see [54] and [14, 50, 11, 43] for more efficient distance oracles.

The sparsest spanner is a tree, but it is impossible to guarantee that a tree spanner has any non-trivial *worst case* distortion.[3] A number of weaker notions of distortion have been defined to deal with tree spanners. A *probabilistic embedding* with distortion t is a distribution over tree metrics such that $\mathbb{E}_T[\delta_T(\phi(u), \phi(v))/\delta_S(u, v)]\} \leq t$, where it is assumed that $\delta_T(\phi(u), \phi(v)) \geq \delta_S(u, v)$. Elkin et al. [28] showed that for any graph there is a probabilistic embedding into its spanning trees with distortion $O(\log^2 n \log\log n)$.[4] Fakcharoenphol et al. [35] proved that *any* metric can be probabilistically embedded in a tree metric (not necessarily a spanning tree) with distortion $O(\log n)$. These probabilistic embeddings have surprisingly diverse applications; see [8, 10, 35, 52, 58] and the references therein. The distinction between embedding into a spanning subtree vs. any tree metric was explored by Bădoiu, Indyk, and Sidiropoulos [16], who showed that the distortion of the best spanning subtree is between $\Omega(\log n/\log\log n)$ and $O(\log n)$ times the distortion of the best tree metric. They also give an algorithm to approximate the best embedding from a given metric into a tree metric.[5]

A number of recent papers look at spanners with ϵ-slack, meaning the stated distortion (a function of ϵ) may fail to hold for an ϵ fraction of the vertex pairs. Such a spanner is *gracefully degrading* if it has ϵ-slack for all ϵ. Chan, Dinitz, and Gupta [17] give a linear size, gracefully degrading spanner with multiplicative distortion $O(\log \epsilon^{-1})$. See [1, 2, 3] for standard and probabilistic embeddings into tree metrics with ϵ-slack.

For geometric graphs, where the vertices are points in $\mathbb{R}^d$, it is known that for any constants d, ϵ, there are efficiently constructible linear size $(1+\epsilon)$-spanners [37, 44]. Geometric graphs fall into a larger class of metrics with constant *doubling* dimension.[6] It was recently shown that even these metrics have $(1+\epsilon)$-spanners with size $O(n)$, for constant ϵ and dimension d [19, 18, 39].

Organization In Section 2 we introduce the path-buying technique and an algorithm for finding additive 6-spanners. In Section 2.1 we show how the path-buying algorithm can be parameterized to compute other additive spanners. Section 3 contains our linear time algorithm for constructing $(k, k-1)$-spanners and in Section 3.1 we show how it can easily be adapted to other models of computation. In Section 4 we conclude with some open problems.

2 Additive Spanners

Our construction for additive 6-spanners works in two phases, the first of which involves standard clustering techniques. In phase one we choose a collection of disjoint vertex sets $\mathcal{C} = \{C_1, C_2, \ldots, C_{n^{2/3}}\}$; each C_i is a

[3] For example, consider a cycle of n vertices.

[4] They actually show this distortion holds for adjacent vertices, which is stronger.

[5] Notice the difference between absolute vs. relative distortion. Most results give an absolute guarantee on the distortion (perhaps in expectation) whereas [16] compare the distortion of their embedding against the optimal one. See [33, 31] for other (in)approximability results for different spanner problems.

[6] A metric has doubling dimension d if the ball of radius $2r$ centered at any point can be covered by 2^d balls of radius r.

cluster with a *center* vertex that is adjacent to all other vertices in its cluster. The set H_0 (which is a subset of our spanner) consists of a radius-one spanning tree of each cluster and all edges that are incident to at least one unclustered vertex. In Section 3 we give two linear time algorithms for constructing $\mathcal{C}$ and H_0 such that $|H_0| \le n^{4/3}$. Let $\mathcal{C}(v)$ be the cluster containing v, if any, and if Z is a subgraph let $\mathcal{C}(Z)$ be the set of clusters that intersect Z.

Notice that since H_0 contains all edges incident to unclustered vertices we can focus our attention on shortest paths whose endpoints are both clustered. The objective of phase two is to show that on any shortest path $P = \langle u, \dots, u'\rangle$, where both u and u' are clustered, there exists a short path Q in the spanner from u to u' that passes through some $C^* \in \mathcal{C}(P)$. We guarantee, in particular, that the portions of Q from $\mathcal{C}(u) \rightsquigarrow C^*$ and $C^* \rightsquigarrow \mathcal{C}(u')$ are no longer than their counterparts in P. Property 2.1 formalizes this idea, and Lemma 2.2 states that any subgraph with this property is an additive 6-spanner.

Property 2.1 *A subgraph $H \supseteq H_0$ is* happy *if for any two* clustered *vertices u, u', there exists a shortest path $P = \langle u \dots, u'\rangle$ in G and a $C^* \in \mathcal{C}(P)$ such that:*

$$\delta_H(\mathcal{C}(v), C^*) \le \delta_P(\mathcal{C}(v), C^*) \quad \text{for both } v \in \{u, u'\}.$$

Lemma 2.2 *Any happy subgraph of G is also an additive 6-spanner of G.*

Proof: Let H be the happy subgraph, and u, u', P, and $C^* \in \mathcal{C}(P)$ be as in the statement of Property 2.1; see Figure 2. We can bound the distance from u to u' in H as:

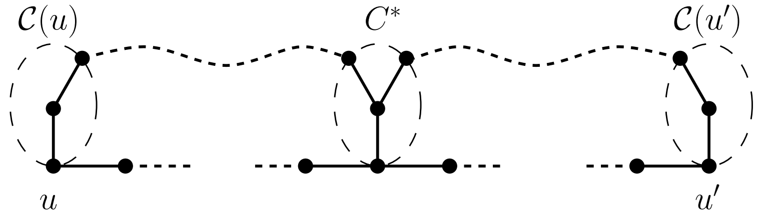

Figure 2: The clusters $\mathcal{C}(u)$, C^*, and $\mathcal{C}(u')$ indicated by ovals. The shortest inter-cluster paths in H are indicated by dashed curves.

$$\begin{aligned}
\delta_H(u,u') &\le \operatorname{diam}_H(\mathcal{C}(u)) + \delta_H(\mathcal{C}(u), C^*) + \operatorname{diam}_H(C^*) + \delta_H(C^*, \mathcal{C}(u')) + \operatorname{diam}_H(\mathcal{C}(u')) \\
&\le \delta_P(\mathcal{C}(u), C^*) + \delta_P(C^*, \mathcal{C}(u')) + 6 \\
&\le \delta_G(u,u') + 6
\end{aligned}$$

where $\operatorname{diam}_H(Z)$ represents the maximum distance between vertices in Z in the subgraph H. The second inequality follows directly from Property 2.1. □

In phase two we find a subgraph $H_0 \cup P_1 \cup P_2 \cup \cdots \cup P_p$ where $P_1, \dots, P_p$ are paths purchased by the *path buying* algorithm in Figure 3. The algorithm is parameterized by *cost* and *value* functions. It evaluates some shortest path between each pair of vertices and purchases the path if twice its value exceeds its cost. If the path is bought this will influence the cost of value of other paths.

The following cost and value functions give rise to an additive 6-spanner. In Section 2.1 we show that different costs and values lead to other sparseness-distortion tradeoffs, including a new 2-spanner of size $O(n^{3/2})$ that is quite different from previous constructions [6, 25, 30, 55].

$$\text{value}(P) = |\{\{C, C'\} \subseteq \mathcal{C}(P) \ : \ \delta_P(C, C') < \delta_H(C, C')\}|$$

$$\text{cost}(P) = |P \backslash H|$$

Note that the cost and value of a path is with respect to a subgraph H, which is our spanner under construction. The cost of a path is the number of its edges not already included in the spanner. The value function represents, roughly, how much the inter-cluster distances would be improved if P *were* included in the spanner.

Our path buying algorithm (Phase 2) is given in Figure 3. It refers to a set $\mathcal{P}$ of shortest paths between all pairs of vertices with the following restrictions. If $P \in \mathcal{P}$ then all subpaths of P are also in $\mathcal{P}$. Furthermore, for every three consecutive vertices $\langle u_1, u_2, u_3\rangle$ in a path $P \in \mathcal{P}$, if $\mathcal{C}(u_1) = \mathcal{C}(u_3)$, then u_2 is the center of $\mathcal{C}(u_1)$. This helps to reduce the cost of paths since $\langle u_1, u_2, u_3\rangle \subseteq H_0$.

$H \leftarrow H_0$	{edges chosen in clustering}
For each shortest path $P \in \mathcal{P}$	
If $2 \cdot \text{value}(P) \geq \text{cost}(P)$	{if the path is a bargain}
then $H \leftarrow H \cup P$	{then buy it!}
Return H	

Figure 3: The path buying algorithm. $\mathcal{P}$ is a set of $\binom{n}{2}$ shortest paths between all pairs of vertices.

The remainder of the proof is structured as follows. In Lemma 2.3 we argue that in the sum of values of paths purchased, the number of times any cluster pair is counted is bounded by a constant. This implies that the sum of values is $O(n^{4/3})$, since $|\mathcal{C}| = n^{2/3}$, and by our criterion for purchasing paths, that the sum of costs is also $O(n^{4/3})$. In Lemma 2.4 we relate the cost of a path to the number of clusters intersecting it. Finally, and most importantly, Lemma 2.5 shows that if any shortest path is too expensive to be purchased then the existing spanner edges already guarantee a path with additive distortion at most 6.

In the following lemmas value(P) and cost(P) represent the value and cost of P at the time it was considered by the path buying algorithm in Figure 3.

Lemma 2.3 *Let $H = H_0 \cup P_1 \cup P_2 \cup \cdots \cup P_p$, where P_i is the ith path bought in the path buying phase. Then* $\sum_{i=1}^{p} \text{value}(P_i) \leq 5\binom{|\mathcal{C}|}{2} < \frac{5}{2}n^{4/3}$.

Proof: Let $H_i = H_0 \cup P_1 \cup \cdots \cup P_i$. For any two clusters $C, C' \in \mathcal{C}$ let $P(C, C') = \{P_{j(1)}, P_{j(2)}, \ldots, P_{j(r)}\}$ be those purchased paths such that

$$\delta_{P_{j(k)}}(C, C') < \delta_{H_{j(k)-1}}(C, C') \qquad \text{for } k \in [1, r].$$

By the definition of the value function, $\sum_{i=1}^{p} \text{value}(P_i) = \sum_{\{C,C'\}\subseteq\mathcal{C}} |P(C, C')|$. Since $P_{j(1)}$ is a *shortest* path in G we have that: $\delta_{P_{j(1)}}(C, C') \leq \text{diam}_G(C) + \delta_G(C, C') + \text{diam}_G(C') \leq \delta_G(C, C') + 4$. This implies that $|P(C, C')| \leq 5$ since $\delta_G(C, C') \leq \delta_{P_{j(r)}}(C, C') < \delta_{P_{j(r-1)}}(C, C') < \cdots < \delta_{P_{j(1)}}(C, C') \leq \delta_G(C, C') + 4$. That is, after $P_{j(1)}$ is purchased the distance from C to C' can only be improved four more times. □

Lemma 2.4 *If $P \in \mathcal{P}$ then either $|\mathcal{C}(P)| = 1$ or there exists a subpath $P' \subseteq P$ such that $\mathcal{C}(P') = \mathcal{C}(P)$ and* $\text{cost}(P') \leq 2\,|\mathcal{C}(P')| - 3$.

Proof: Let $P = \langle u, \ldots, u'\rangle$ and $P' \subseteq P$ be minimal such that $\mathcal{C}(P) = \mathcal{C}(P')$. (This means that if the first or last cluster of P has two or three vertices in common with P then only the innermost one appears in P'.) The only edges in P' that might not be in $H \supseteq H_0$ are those between clustered vertices. Furthermore, if three consecutive vertices u_1, u_2, u_3 belong to the same cluster then $\langle u_1, u_2, u_3\rangle \subseteq H_0$. Thus the total number of inter-cluster edges and intra-cluster edges that are not in H are bounded by $|\mathcal{C}(P')| - 1$ and $|\mathcal{C}(P')| - 2$. □

Lemma 2.5 *The subgraph H returned by the path buying algorithm is happy.*

Proof: Let $P = \langle u, \ldots, u'\rangle \in \mathcal{P}$ be the shortest path from u to u' in G. By the statement of Property 2.1 we can dispense with several trivial cases and assume that P was not purchased in phase two, that both u and u' are clustered and that $\mathcal{C}(u) \neq \mathcal{C}(u')$. Let $P' \subseteq P$ be the subpath guaranteed by Lemma 2.4. The case when P' is included in H is also trivial. Thus we have the following inequalities:

$$2 \cdot \text{value}(P') < \text{cost}(P') \leq 2 \cdot |\mathcal{C}(P')| - 3 \tag{1}$$

where the first inequality follows from the fact that P' was not included in H and the second from Lemma 2.4. Define A as the set of cluster pairs:

$$A = \left\{ \{C_0, C_1\} \;:\; \begin{array}{l} C_0 \in \{\mathcal{C}(u), \mathcal{C}(u')\},\; C_1 \in \mathcal{C}(P')\backslash\{C_0\} \\ \text{and } \;\delta_{P'}(C_0, C_1) < \delta_H(C_0, C_1) \end{array} \right\}.$$

The cluster pairs counted in A are also counted in value(P') so $|A| \leq$ value(P'). By the inequalities of Eqn. 1 value$(P') \leq |\mathcal{C}(P')| - 2$. Notice that the maximum number of cluster pairs counted by A is $2|\mathcal{C}(P')| - 3$. This means that for at least $|\mathcal{C}(P')| - 1$ of these cluster pairs, their distance in the spanner is no worse than their distance in P'. By the pigeonhole principle there must be some cluster $C^* \in \mathcal{C}(P') = \mathcal{C}(P)$ satisfying both

$$\delta_H(\mathcal{C}(u), C^*) \leq \delta_{P'}(\mathcal{C}(u'), C^*)$$

and

$$\delta_H(C^*, \mathcal{C}(u')) \leq \delta_{P'}(C^*, \mathcal{C}(u')).$$

Since $\mathcal{C}(P) = \mathcal{C}(P')$ it also follows that P has this property. □

Lemma 2.6 *If H is the subgraph purchased by the path buying algorithm then $|H| < 6 \cdot n^{4/3}$.*

Proof: One can easily see that $|H| = |H_0| + \sum_i \text{cost}(P_i)$. By construction we have $|H_0| \leq n^{4/3}$. It follows from Lemma 2.3 that $\sum_i \text{cost}(P_i) \leq 2 \cdot \sum_i \text{value}(P_i) < 5 \cdot n^{4/3}$. □

Theorem 2.7 *There exists an additive 6-spanner of any graph with size $O(n^{4/3})$.*

Proof: Follows from Lemmas 2.2, 2.5, and 2.6. □

2.1 Parameterizing the Path Buying Algorithm

Our 6-spanner construction can be generalized to larger additive distortion. However, we can only guarantee that a β-spanner ($\beta > 6$) has $o(n^{4/3})$ edges if the graph satisfies additional requirements. Let $\Gamma_k(G)$ be the number of edges in G that lie on some cycle with length at most $2k$. Theorem 2.8 generalizes Theorem 2.7 and provides an efficient construction algorithm.

Theorem 2.8 *For any graph G and integer parameters $k \geq 1$ and $\ell \in [0, k]$, there exists an additive $(2k + 4\ell)$-spanner with size $O(\Gamma_k(G) + n^{1+\frac{1}{k+\ell+1}})$. Furthermore, the spanner can be constructed in time $O(mn^{1-\frac{\ell}{k+\ell+1}})$.*

Theorem 2.8 says that if the graph is not *too* far from having girth greater than $2k$ then we can construct a number of additive spanners with different size-distortion tradeoffs. Notice that when $k = 1$, $\Gamma_1(G) = 0$ since all graphs have girth at least 3. As special cases, Theorem 2.8 gives 2- and 6-spanners with size $O(n^{3/2})$ and $O(n^{4/3})$. (This 2-spanner construction is slower and more complicated than earlier ones but it does illustrate the applicability of the path buying algorithm.) Figure 4 lists some of the corollaries of Theorem 2.8 for graphs with girth greater than 4 and greater than 6.

The spanners provided by Theorem 2.8 are constructed with a path buying algorithm, with the following differences. First, it uses a generalized clustering scheme that produces clusters of two radii: k and ℓ. Second, it uses a value function that measures how well the spanner under construction approximates the distance between cluster pairs C, C', where C has radius ℓ and C' radius k.

Before running the path buying algorithm we compute an appropriate clustering and initial set of edges $H_{k,\ell}$, which plays the same role as H_0 in our 6-spanner. This clustering procedure that produces $H_{k,\ell}$ is very similar to the randomized clustering procedure presented later in Section 3, though the analysis of these two schemes is sufficiently different to justify two expositions. We begin by selecting vertex sets V_ℓ and V_k where V_ℓ is a random sample of V of size $n^{1-\ell\epsilon}$ and V_k is a random sample of V_ℓ of size $n^{1-k\epsilon}$, where ϵ will be chosen later. We find two clusterings $\mathcal{C}_\ell, \mathcal{C}_k$, where $\mathcal{C}_i$ consists of a set of disjoint subsets of V. Each $C \in \mathcal{C}_i$ is centered at some vertex in V_i and a vertex v is contained in some $C \in \mathcal{C}_i$ if and only if $\delta(v, V_i) \leq i$; in particular, the distance from v to the center of its cluster is at most i. $H_{k,\ell}$ consists of a radius-i spanning tree of each $C \in \mathcal{C}_i$ and $i \in \{k, \ell\}$, as well as *every* edge incident to a vertex that does not appear in both clusterings $\mathcal{C}_\ell$ and $\mathcal{C}_k$. The number of edges contributed by the spanning trees is only $2n$ so we are mainly concerned with the expected number of the remaining edges.

Lemma 2.9 *Given a graph G with m edges, the subgraph $H_{k,\ell}$ can be constructed in $O(m)$ time and $\mathbb{E}[|H_{k,\ell}|] = O(\Gamma_k(G) + n^{1+\epsilon})$.*

Add. Dist.	Spanner Size	Construction Time	Notes
2	$O(n^{3/2})$	$O(mn)$	$k=1, \ell=0$, cf. [6, 25, 30, 55]
6	$O(n^{4/3})$	$O(mn^{2/3})$	$k=\ell=1$
GRAPHS WITH GIRTH > 4			
0	$O(n^{3/2})$	—	[56, 54]
4	$O(n^{4/3})$	$O(mn)$	$k=2, \ell=0$
8	$O(n^{5/4})$	$O(mn^{3/4})$	$k=2, \ell=1$
12	$O(n^{6/5})$	$O(mn^{3/5})$	$k=\ell=2$
GRAPHS WITH GIRTH > 6			
0	$O(n^{4/3})$	—	[56, 54]
6	$O(n^{5/4})$	$O(mn)$	$k=3, \ell=0$
10	$O(n^{6/5})$	$O(mn^{4/5})$	$k=3, \ell=1$
14	$O(n^{7/6})$	$O(mn^{2/3})$	$k=3, \ell=2$
18	$O(n^{8/7})$	$O(mn^{4/7})$	$k=\ell=3$

Figure 4: Additive spanners derived from Theorem 2.8. See [56] for constructions of graphs with girth > 4 and > 6, and [54] for references to earlier constructions of high-girth graphs.

Proof: To simplify the analysis we imagine constructing $H_{k,\ell}$ by selecting vertex sets $V = V_0 \supseteq V_1 \supseteq \cdots \supseteq V_k$, where V_i is a random sample of V_{i-1} of size $n^{1-i\epsilon}$. We construct the clusterings $\mathcal{C}_0, \ldots, \mathcal{C}_k$ iteratively. Let $c \in C \in \mathcal{C}_{i-1}$, where $c \in V_{i-1}$ is the center of a cluster C. If c also appears in V_i then every vertex incident to C appears in some cluster in $\mathcal{C}_i$. In particular, c's cluster in $\mathcal{C}_i$ consists of C and some subset of the vertices incident to C that join C. (A vertex may be eligible to join multiple clusters and can choose any one.) Suppose that we decide to include all edges in $H_{k,\ell}$ that have at least one endpoint unclustered in any of $\mathcal{C}_0, \ldots, \mathcal{C}_k$. Let v be some vertex that has appeared in all the clusterings $\mathcal{C}_0, \ldots, \mathcal{C}_{i-1}$ and consider the effect on v of randomly choosing the subset V_i. Suppose v is incident to s clusters in $\mathcal{C}_{i-1}$, say $C_1, \ldots, C_s$. The probability that v is included in $\mathcal{C}_i$ is at least[7] $1-(1-n^{-\epsilon})^s$. The effect of v *not* appearing in $\mathcal{C}_i$ is to include in $H_{k,\ell}$ all edges incident to v, which could be significantly larger than s and n^ϵ. Let $\{C_1, \ldots, C_{s'}\}$ be those clusters connected to v by at most one edge (this would be exactly one edge, except for the one cluster that actually contains v.) Every edge (v,w) connecting v to a cluster C appearing in $\{C_{s'+1}, \ldots, C_s\}$ lies on a cycle of length at most $2i \le 2k$: the one consisting of (v,w), a path from w to $w' \in C$ passing through the center of C, and the edge (w', v). Such a w' exists because v is connected to C by at least two edges. If v is unclustered in $\mathcal{C}_i$ then the contribution of the edges from v to $C_{s'+1}, \ldots, C_s$ is counted in the $\Gamma_k(G)$ term. Excluding these edges, the expected number of edges contributed by v is at most $(s'-1) \cdot \Pr[v \text{ does not appear in } \mathcal{C}_i] \le (s'-1)(1-n^{-\epsilon})^s < n^\epsilon$.[8] □

We now turn to the proof of Theorem 2.8.

Proof: (of Theorem 2.8) First notice that the only edges missing from $H_{k,\ell}$ are those with both endpoints present in both clusterings $\mathcal{C}_\ell, \mathcal{C}_k$, so we can restrict our attention to shortest paths between clustered vertices. Let $\mathcal{C}_i(v)$ be the $\mathcal{C}_i$ cluster containing v and $\mathcal{C}_i(Z)$ be the set of clusters in $\mathcal{C}_i$ intersecting the subgraph Z. After running the path buying algorithm with appropriate cost and value functions we end up with a spanner H that is *happy* under a new definition of happiness.

Property 2.10 *A subgraph H of G that contains $H_{k,\ell}$ is* happy *if for any $u, u' \in V_\ell$, there is a shortest path P (w.r.t. G) from u to u' and a cluster $C^* \in \mathcal{C}_k(P)$ such that: $\delta_H(u, C^*) \le \delta_P(u, C^*)$ and $\delta_H(u', C^*) \le \delta_P(u', C^*)$.*

[7] This would be the *exact* probability if V_i were selected from V_{i-1} by sampling each element with probability $n^{-\epsilon}$. Selecting V_i as a random subset of V_{i-1} with size $n^{1-i\epsilon}$ only improves v's chance of being clustered in $\mathcal{C}_i$.

[8] For the last inequality, observe that $s'(1-1/x)^{s'} \le x(s'/x)e^{-s'/x} \le x$, for any positive x.

It follows that any happy subgraph H is also an additive $(2k+4\ell)$-spanner. To see this, consider a shortest path between any two vertices v, v' and let u and u' be the center of the $\mathcal{C}_\ell$ clusters containing v and v', respectively. Let P be the shortest path between u and u'. By the happiness of H there exists a cluster $C^* \in \mathcal{C}_k(P)$ such that

$$\delta_H(u,u') \leq \delta_H(u,C^*) + \mathrm{diam}(C^*) + \delta_H(C^*,u') \leq \delta(u,u') + 2k.$$

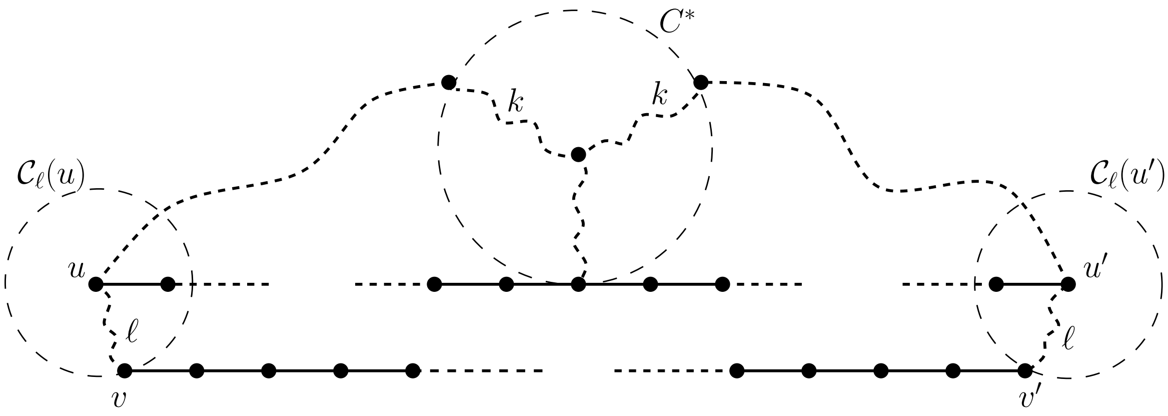

Figure 5: The clusters $\mathcal{C}_\ell(u), \mathcal{C}_\ell(u')$, and $C^* \in \mathcal{C}_k$, are indicated by ovals. Dashed curves represent paths in the spanner H. The shortest paths in G connecting v to v' and u to u' are indicated.

Using this bound on the distance from u to u' in H, we can bound $\delta_H(v,v')$ as:

$$\delta_H(v,v') \;\leq\; \delta_H(v,u) + \delta_H(v',u') + \delta_H(u,u') \;\leq\; 2\ell + \delta(u,u') + 2k \;\leq\; \delta(v,v') + 2k + 4\ell.$$

Let $\mathcal{P}$ be a set of shortest paths between all pairs of vertices. We only consider shortest paths in $\mathcal{P}$ and insist that it satisfy two properties. First, any subpath of a path in $\mathcal{P}$ is also in $\mathcal{P}$. Second, if $\langle u_0, u_1, \ldots, u_{2k}\rangle \in \mathcal{P}$ and $\mathcal{C}_k(u_0) = \mathcal{C}_k(u_{2k})$, then u_k is the center of $\mathcal{C}_k(u_0)$ and $\langle u_0, \ldots, u_{2k}\rangle$ is contained in the spanning tree of $\mathcal{C}_k(u_0)$ included in $H_{k,\ell}$. Let $P(u,u') \in \mathcal{P}$ be the shortest path between u and u'.

We use the same cost function as before: $\mathrm{cost}(P) = |P \backslash H|$, where H now represents the spanner under construction. However, the value function is *only* defined on paths connecting vertices in V_ℓ. Letting $P = P(u,u')$, where $u, u' \in V_\ell$, we define value(P) as:

$$\mathrm{value}(P) \;=\; |\{(v,C) \,:\, v \in \{u,u'\},\ C \in \mathcal{C}_k(P),\ \text{ and }\ \delta_P(v,C) < \delta_H(v,C)\}|\,.$$

The path buying algorithm below has a few differences from the 6-spanner algorithm. We only consider shortest paths between vertices in V_ℓ and the criterion for purchasing a path P is slightly stronger: $2k \cdot \mathrm{value}(P)$ (rather than $2 \cdot \mathrm{value}(P)$) must be at least cost$(P)$.

$H \leftarrow H_{k,\ell}$	{edges chosen in clustering}
For each $\{u,u'\} \subseteq V_\ell$ let $P = P(u,u')$	
If $2k \cdot \mathrm{value}(P) \;\geq\; \mathrm{cost}(P)$	{if the path is a bargain}
then $H \leftarrow H \cup P$	{then buy it!}
Return H	

Figure 6: Constructing an additive $(2k+4\ell)$-spanner.

Let $P_1, \ldots, P_p$ be the paths purchased. It follows that the size of H is exactly $|H_{k,\ell}|$ plus $\sum_{1\leq i\leq p} \mathrm{cost}(P_i) \leq 2k\sum_{1\leq i\leq p} \mathrm{value}(P_i) \leq 2\mu k\,|V_\ell \times V_k| = 2\mu k n^{2-(k+\ell)\epsilon}$, where μ is the maximum number of times that any pair (v,C), where $v \in V_\ell, C \in \mathcal{C}_k$ could be counted in $\sum_i \mathrm{value}(P_i)$. One can see that $\mu = 2k+1$. When (v,C) is first counted, say when P_i is purchased, we have $\delta_{P_i}(v,C) \leq \delta(v,C) + \mathrm{diam}(C) = \delta(v,C) + 2k$. After

P_i is purchased the distance between v and C in H can only be improved $2k$ more times. Thus, the size of H is $O(\Gamma_k(G) + n^{1+\epsilon} + k^2 n^{2-(k+\ell)\epsilon})$. Setting $n^\epsilon = (k^2 n)^{1/(k+\ell+1)}$, the total size is $O(\Gamma_k(G) + n^{1+1/(k+\ell+1)})$.

It only remains to show that H is happy and in particular, that if a path $P = P(u, u')$ is not purchased then $\delta_H(u, u') \leq \delta(u, u') + 2k$. We first bound cost$(P)$ then show that if P is not purchased, there must be some $C^* \in \mathcal{C}_k(P)$ satisfying Property 2.10

Consider any cluster $C \in \mathcal{C}_k(P)$ and let z, z' be the first and last vertices in P that belong to C. An edge is internal to C if it lies on the subpath $\langle z, \ldots, z'\rangle$. (Note that not all vertices between z and z' are necessarily in C.) It follows from our choice of shortest paths $\mathcal{P}$ that either $|\langle z, \ldots, z'\rangle| \leq 2k - 1$ or $\langle z, \ldots, z'\rangle \subseteq H_{k,\ell} \subseteq H$. Thus, the total number of internal edges in $P \backslash H$ is at most $(2k-1)\,|\mathcal{C}_k(P)|$ and the number of remaining edges in $P \backslash H$ is at most $|\mathcal{C}_k(P)| - 1$. In total we have cost$(P) \leq 2k\,|\mathcal{C}_k(P)| - 1$. If we fail to purchase P then $2k \cdot \text{value}(P) \leq \text{cost}(P) - 1 \leq 2k\,|\mathcal{C}_k(P)| - 2$, implying that value$(P) \leq |\mathcal{C}_k(P)| - 1$. Since there are $2\,|\mathcal{C}_k(P)|$ cluster pairs of the form (v, C) where $v \in \{u, u'\}$ and $C \in \mathcal{C}_k(P)$, our bound on value(P) implies that for at least one $C^* \in \mathcal{C}_k(P)$, $\delta_H(u, C^*) \leq \delta_P(u, C^*)$ and $\delta_H(u', C^*) \leq \delta_P(u', C^*)$. Therefore H is an additive $(2k + 4\ell)$-spanner.

We now turn to an efficient construction algorithm. Recall from Lemma 2.9 that $H_{k,\ell}$ can be found in linear time. In the path buying phase it is rather time consuming to maintain the actual distance in H between pairs $(v, C) \in V_\ell \times \mathcal{C}_k$, which would be required to compute the value function exactly. Instead, we keep an upper bound $\hat{\delta}_H(v, C)$, which is the minimum distance between v and C in a path already purchased. We also ignore the cost function and consider the valid upper bound cost$(P) \leq 2k\,|\mathcal{C}_k(P)| - 1$. (Clearly these changes do not affect the correctness of the algorithm.) At the beginning of the path buying phase we construct, in $O(m\,|\mathcal{C}_\ell|)$ time, a shortest path tree originating at each vertex in V_ℓ. In such a tree a *relevant* vertex is one in V_ℓ or one that has at least two V_ℓ vertices in different subtrees, that is, a branching vertex in the subtree connecting V_ℓ nodes. For $u \in V_\ell$ let R_u be the set of relevant vertices in u's shortest path tree. We consider, in non-decreasing order by length, the shortest paths between pairs (u, u') where $u \in V_\ell$ and $u' \in R_u$. For a path $P = \langle u_0, u_1, \ldots, u_q\rangle$, where $u_0 \in V_\ell$, let $\nu(u_0, u_q) = \left|\left\{(u_0, C) \,:\, C \in \mathcal{C}_k(P) \text{ and } \delta_P(u_0, C) < \hat{\delta}_H(u_0, C)\right\}\right|$. We assume that after P is considered we have computed $\nu(u_0, u_i)$ and $|\mathcal{C}_k(\langle u_0, \ldots, u_i\rangle)|$, for all relevant vertices $u_i \in R_{u_0}$. When we consider an extension of P, say $P' = \langle u_0, \ldots, u_q, \ldots, u_{q+r}\rangle$, where u_{q+r} is the first relevant vertex on the extension, we can easily update $|\mathcal{C}_k(P')|$ and $\nu(u_0, u_{q+r})$ in just $O(k + r)$ time. If $u_{q+r} \in V_\ell$ then we need to decide whether to purchase P'. We check if $\nu(u_0, u_{q+r}) + \nu(u_{q+r}, u_0) \geq |\mathcal{C}_k(P')|$, which serves the same purpose as checking the inequality $2k \cdot \text{value}(P') \geq \text{cost}(P')$. If the inequality holds we buy P' and update $\nu(u, u')$ where $u \in \{u_0, u_{q+r}\}$ and $u' \in R_u$. This amounts to checking, for each pair (u_0, C) counted in $\nu(u_0, u_{q+r})$ and each path $P'' = \langle u_0, \ldots, u'\rangle$, whether $\delta_{P'}(u_0, C) \leq \delta_{P''}(u_0, C)$. If so, the pair (u_0, C) should no longer be counted in $\nu(u_0, u')$, if it was counted there at all. The total time for these updates is $O(k \cdot |\mathcal{C}_\ell|^2 \cdot |\mathcal{C}_k|) = O(n^{3 - \frac{2\ell + k}{k+\ell+1}})$ and the total time for performing breadth first searches is $O(m \cdot |\mathcal{C}_\ell|) = O(mn^{1 - \frac{\ell}{k+\ell+1}})$. We can safely assume that $m \geq n^{1 + \frac{1}{k+\ell+1}}$; if not we could simply return the original graph as a trivial additive 0-spanner. For m above this threshold the two time bounds are both $O(mn^{1 - \frac{\ell}{k+\ell+1}})$. □

The first obstacle to improving Theorem 2.8 is dealing with graphs with lots of 3- and 4-cycles. It is strange that short cycles should impede the discovery of more additive spanners since high-girth graphs are the most difficult instances when optimizing for *multiplicative* distortion. The recent lower bounds of Woodruff [57] provide some circumstantial evidence that short cycles really do complicate the problem. His hard instances are actually composed entirely of complete bipartite graphs, where each edge appears on a huge number of 4-cycles.

Theorem 2.8 generalizes the 6-spanner construction by considering clusterings with wider radii. Another direction for generalization is to consider more hops between clusters. Theorem 2.11 follows from a small adjustment to the 6-spanner algorithm.

Theorem 2.11 *Every graph on n vertices contains an additive $O(n^{1-3\epsilon})$-spanner with size $O(n^{1+\epsilon})$, for any constant $\epsilon \in (0, \frac{1}{3})$.*

Proof: The algorithm is nearly identical to the 6-spanner construction. We find a radius-1 clustering $\mathcal{C}$ containing $n^{1-\epsilon}$ clusters, and let H_0 contain all edges incident to at least one unclustered vertex and a spanning tree of each cluster. We run the path buying algorithm from Figure 3 using the same cost function but a different value function:

$$\text{value}(P) = n^{-1+3\epsilon} \cdot |\{\{C, C'\} \subseteq \mathcal{C}(P) \;:\; \delta_P(C, C') < \delta_H(C, C')\}|\,.$$

The rationale for this value function is simple. If there are $n^{1-\epsilon}$ clusters and our desired spanner size is $O(n^{1+\epsilon})$, each cluster pair can only afford to be charged the cost of buying $O(n^{-1+3\epsilon})$ edges, that is, a small fraction of a single edge. Let P be some shortest path and P' and P'' be the prefix and suffix of P containing $n^{1-3\epsilon}$ distinct clusters. If P is not purchased then, by the same reasoning used before, there must be three not necessarily distinct clusters C', C^*, C'' where $C' \in \mathcal{C}(P'), C'' \in \mathcal{C}(P'')$, and $C^* \in \mathcal{C}(P)$ such that $\delta_H(C', C^*) \leq \delta_P(C', C^*)$ and $\delta_H(C'', C^*) \leq \delta_P(C'', C^*)$. That is, the subpath of P connecting C' to C'' is approximated by H to within an additive distortion of 6. To get from the first vertex of P to C' and from C'' to the last vertex of P we use any standard multiplicative $O(\epsilon^{-1})$-spanner with size $O(n^{1+\epsilon})$. The total additive distortion is therefore $6 + O(\epsilon^{-1}(\text{cost}(P') + \text{cost}(P''))) = O(n^{1-3\epsilon})$. The size of H_0 and the multiplicative spanner is $O(n^{1+\epsilon})$ and the number of edges included by the path buying algorithm is $\sum_i \text{cost}(P_i) \leq 2\sum_i \text{value}(P_i) = O(n^{-1+3\epsilon} \cdot \binom{|\mathcal{C}|}{2}) = O(n^{1+\epsilon})$. □

Theorem 2.11 is not particularly impressive, but, again, it illustrates the flexible nature of the path buying technique. The previous best additive spanner with size $O(n^{1+\epsilon})$ had distortion roughly $O(n^{1-2\epsilon})$ [15] and was substantially more complicated to construct.

3 A Simple $(k, k-1)$-Spanner in Linear Time

In this section we will first show how to construct a $(2k-1, 0)$-spanner of size $O(kn^{1+1/k})$ in $O(km)$ deterministic time. Then we will extend this method to construct a $(k, k-1)$ spanner of size $O(kn^{1+1/k})$ in $O(km)$ deterministic time.

The input graph is $G = (V, E)$. A *cluster* is simply a set of vertices and a *clustering* is a set of disjoint clusters. A vertex is *(un)clustered* in a clustering $\mathcal{C}$ if it appears (does not appear) in some cluster of $\mathcal{C}$. In a clustering $\mathcal{C}$, for any clustered vertex u, denote by $\mathcal{C}(u)$ the cluster of $\mathcal{C}$ that contains u. For clusters C and C', let $E(C, C') = (C \times C') \cap E(G)$ be the set of edges between C and C'. Let $E(v, C)$ be the set of edges between the vertex v and vertices in C. A vertex v is adjacent to a cluster C if $E(v, C) \neq \emptyset$. In a similar manner, two clusters C and C' are adjacent to each other if $E(C, C') \neq \emptyset$.

Our constructions in this section are based on a set of $k+1$ clusterings, $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k$, where $\mathcal{C}_0 = \{\{v\} : v \in V(G)\}$, $\mathcal{C}_k = \emptyset$, and $|\mathcal{C}_i| \leq n^{1-i/k}$. Below, we give two methods for constructing appropriate sequences of clusterings. The edge set of our $(2k-1)$-spanner S is defined by the following two rules:

Rule R1. For each cluster $C \in \mathcal{C}_i$, there exists a tree in S that spans C and has radius[9] at most i.

Rule R2. For each vertex v that is unclustered in $\mathcal{C}_i$ and each cluster $C \in \mathcal{C}_{i-1}$ adjacent to v, some edge from $E(v, C)$ appears in S.

The construction of Theorem 3.1 is slightly weaker than that of [38]; however it is the starting point for our $(k, k-1)$-spanner.

Theorem 3.1 *A $(2k-1)$-spanner of size $O(kn^{1+1/k})$ can be constructed in $O(km)$ deterministic time.*

Proof: We first prove that Rules R1 and R2 give a $(2k-1)$-spanner; we then prove the size and time bounds. Let (u, v) be an arbitrary edge in the original graph. If $\delta_S(u, v) \leq (2k-1)\delta_G(u, v)$ then S is clearly a $(2k-1)$-spanner. Let ℓ be minimum such that either u or v was unclustered in $\mathcal{C}_\ell$ and without loss of generality let the unclustered vertex be u. By Rule R2 there must be an edge in S from u to $\mathcal{C}_{\ell-1}(v)$; call this edge (u, w). By Rule R1 there must be a path in S from w to v of length at most $2(\ell-1)$, twice the radius of $\mathcal{C}_{\ell-1}(v)$. Since $\ell \leq k$ it follows immediately that $\delta_S(u, v) \leq 2k-1$.

Given the clustering $\mathcal{C}_i$ we show how to compute $\mathcal{C}_{i+1}$ such that the number of edges added to the spanner due to Rules R1 and R2 are at most n and $n^{1+1/k}$, respectively (This construction is a simplified version of one described by Elkin [26].) Initially $\mathcal{C}_{i+1} = \emptyset$. We define the priority of a cluster $C \in \mathcal{C}_i$ to be the

[9] Recall that maximum distance between any two vertices in a subgraph is at most twice the radius of that subgraph.

number of adjacent vertices that are unclustered with respect to $\mathcal{C}_{i+1}$. We repeatedly choose a cluster $C \in \mathcal{C}_i$ with maximum priority. If priority$(C) \geq n^{(i+1)/k}$ we add a new cluster to $\mathcal{C}_{i+1}$ consisting of C and all unclustered vertices adjacent to C. (If C has radius i then the cluster added to $\mathcal{C}_{i+1}$ clearly has radius $i+1$.) It follows that $|\mathcal{C}_{i+1}| \leq n^{1-(i+1)/k}$ and that the number of edges included in the spanner due to Rule R2 is $\sum_{C \in \mathcal{C}_i}$ priority(C), which is at most $|\mathcal{C}_i|\,(n^{(i+1)/k} - 1) < n^{1+1/k}$. The number of edges added due to Rule R1 is at most the number of clustered vertices in $\mathcal{C}_{i+1}$, i.e. at most n. The clustering $\mathcal{C}_{i+1}$ can easily be generated in linear time using a priority queue consisting of n buckets. □

The randomized construction of [13] constructs the clusterings in an even simpler manner. Rather than carefully selecting clusters from $\mathcal{C}_i$ for inclusion in $\mathcal{C}_{i+1}$, they randomly sample all clusters from $\mathcal{C}_i$ with probability $n^{-1/k}$. The alternative $(2k-1)$-spanner construction based on this method is given in Figure 7. With this algorithm the *expected* size of the spanner is $O(kn^{1+1/k})$.

> Initially $S = \emptyset$ and $\mathcal{C}_0 = \{\{v\} : v \in V(G)\}$
>
> For i from 1 to k
>
> - Let $\mathcal{C}_i$ be sampled from $\mathcal{C}_{i-1}$ with prob. $n^{-1/k}$ (If $i = k$ let $\mathcal{C}_k = \emptyset$)
> - For each vertex v which does not belong to any cluster in $\mathcal{C}_i$ do (concurrently):
>   - (R1) If v is adjacent to some $C \in \mathcal{C}_i$, add v to C and add some edge of $E(v, C)$ to S.
>   - (R2) Otherwise, add to S some edge from $E(v, C)$, for each $C \in \mathcal{C}_{i-1}$ adjacent to v.
>
> Return the $(2k-1)$-spanner S

Figure 7: A randomized $(2k-1)$-spanner construction.

We next improve our $(2k-1)$-spanner construction to obtain a $(k, k-1)$-spanner. All the edges that we put in the $(2k-1)$-spanner S are of the form $E(v, C)$. Now let us also add edges of the form $E(C, C')$, which will consist of edges from clusters of one clustering to clusters of another clustering.

Rule R3. For each i with $0 \leq i \leq k-1$ and for each pair of adjacent clusters C, C' with $C \in \mathcal{C}_i$ and $C' \in \mathcal{C}_{k-1-i}$, some edge from $E(C, C')$ appears in S.

Rule R4. For each $i \geq \lceil k/2 \rceil$ and each pair of adjacent clusters C, C' with $C \in \mathcal{C}_i$ and $C' \in \mathcal{C}_{i-1}$, some edge from $E(C, C')$ appears in S.

The number of edges included due to Rule R3 is bounded by $n^{1-i/k} n^{1-(k-1-i)/k} = n^{1+1/k}$, for each i. Similarly, at most $n^{1-i/k} n^{1-(i-1)/k} = n^{2-2i/k+1/k}$ edges are included due to R4, which is at most $n^{1+1/k}$ since $i \geq \lceil k/2 \rceil$. Our entire $(k, k-1)$-spanner construction is given in Figure 8. It consists of just those edges included by Rules R1–R4.

(R1–2)	Compute a $(2k-1)$-spanner S with our construction.
(R3)	Add to S one edge from $E(C, C')$ for each adjacent pair $C \in \mathcal{C}_i$ and $C' \in \mathcal{C}_{k-1-i}$, for i from 0 to $k-1$.
(R4)	Add to S one edge from $E(C, C')$ for each adjacent pair $C \in \mathcal{C}_i$, and $C' \in \mathcal{C}_{i-1}$, for i from $\lceil k/2 \rceil$ to $k-1$.

Figure 8: A simple linear time algorithm for constructing a $(k, k-1)$-spanner.

Implementing Rules R3 and R4 takes linear time for any fixed i. Once it is proved that Rules R1–R4 yield a $(k, k-1)$-spanner we can conclude with the following theorem.

Theorem 3.2 *A $(k, k-1)$-spanner of size $O(kn^{1+1/k})$ can be computed in $O(km)$ deterministic time.*

We now show that S is a $(k, k-1)$-spanner. Let $t = \lfloor k/2 \rfloor$. We need some more definitions. Call a vertex i-(un)clustered if it appears (does not appear) in clustering $\mathcal{C}_i$. The *center* of a cluster $C \in \mathcal{C}_i$ is a vertex $c \in C$ such that the distance from c to any other vertex in C is at most i, the radius of C. If v is i-clustered let $c_i(v)$ be the center of $\mathcal{C}_i(v)$. In analyzing the stretch of spanner S, we shall make extensive use of the following notations.

- $f_i(v) = \begin{cases} v & \text{if } v \text{ is } i\text{-unclustered} \\ c_i(v) & \text{if } v \text{ is } i\text{-clustered} \end{cases}$ $\qquad r_i(v) = \begin{cases} 0 & \text{if } v \text{ is } i\text{-unclustered} \\ i & \text{if } v \text{ is } i\text{-clustered} \end{cases}$

  It is easy to observe that $\delta_S(v, f_i(v)) \leq r_i(v)$.

- $\sigma(j) = \begin{cases} \sum_{i=0}^{j} 2i & : \ j \geq 0 \\ \sum_{i=0}^{|j|-1} 2i & : \ j < 0 \end{cases}$

  It is easy to observe that the equalities $\sigma(-j) = \sigma(j-1)$ and $\sigma(j) - 2j = \sigma(j-1)$ hold for all j.

We shall now state three simple Lemmas which follow from the rules R1–R4 used for constructing the spanner. Lemma 3.3 is based on R2 and is already proved as part of Theorem 3.1.

Lemma 3.3 *If v is ℓ-unclustered for $\ell \leq k-1$, then for each edge $(u, v) \in E$,*

$$\delta_S(f_{\ell-1}(u), f_\ell(v)) \leq 2\ell - 1 - r_{\ell-1}(u)$$

The following Lemma uses the rule R3 in a crucial manner.

Lemma 3.4 *If v is ℓ-clustered, then the following inequality holds for each edge $(u, v) \in E$.*

$$\delta_S(f_{k-1-\ell}(u), f_\ell(v)) \leq 2k - 2\ell - 1 + r_\ell(v) - r_{k-1-\ell}(u)$$

Proof: If u is $(k-1-\ell)$-unclustered then it follows from rule R2 that there is a path of length at most $2k-2\ell-3$ in S between u and v. Since there is a path of length $\leq r_\ell$ from v to $c_\ell(v)$ in S, the inequality holds. The other case is when u is $(k-1-\ell)$-clustered. By Rule R3 there is an edge in S between $\mathcal{C}_{k-1-\ell}(u)$ and $\mathcal{C}_\ell(v)$. This gives us a path of length k between $c_{k-1-\ell}(u)$ and $c_\ell(v)$. Also note that $r_\ell(v) - r_{k-1-\ell}(u) = -(k-1-2\ell)$. Therefore, the inequality $\delta_S(f_{k-1-\ell}(u), f_\ell(v)) \leq 2k - 2\ell - 1 + r_\ell(v) - r_{k-1-\ell}(u)$ holds in this case as well. □

The following Lemma uses the rule R4 in a crucial manner.

Lemma 3.5 *For each edge $(u, v) \in E$ and integer $\ell \geq \lceil \frac{k}{2} \rceil$, the following inequality holds.*

$$\delta_S(f_{\ell-1}(u), f_\ell(v)) \leq 2\ell - 1 + r_\ell(v) - r_{\ell-1}(u)$$

Proof: Depending upon whether u and v appear in clusterings $\mathcal{C}_{\ell-1}$ and $\mathcal{C}_\ell$, respectively, we have four cases: both are unclustered, both are clustered, u is clustered but v is not and vice-versa. For the first case, that is $u \in \mathcal{C}_{\ell-1}$ and $v \in \mathcal{C}_\ell$, it follows from rule R4 that there is a spanner path between $f_{\ell-1}(u)$ and $f_\ell(v)$ of length 2ℓ. This fact combined with the observation that $r_\ell(v) = \ell$ and $r_{\ell-1}(u) = \ell - 1$ imply the inequality. The proof of the remaining three cases follow from Lemma 3.3 in a straight forward manner. □

We are now ready to prove that S is indeed a $(k, k-1)$-spanner. Our analysis will be based on Lemmas 3.3,3.4,3.5. We first analyze the case when k is an odd positive integer.

Analysis of the stretch for odd value of k

For the case when k is odd, we shall prove the following Theorem whose simple corollary would imply that S is $(k, k-1)$-spanner. Note that $k = 2t + 1$ since k is odd.

Theorem 3.6 *Let $\langle u = v_0, v_1, \cdots v_q = v \rangle$ be a path between any $u, v \in V$ in the given graph, and S be the spanner computed by our algorithm for an odd integer k. Then for each $0 \leq i \leq q$ and integer $\ell \in [0, 2t]$,*

$$\delta_S(v_0, f_\ell(v_i)) \leq ki + r_\ell(v_i) + \sigma(t - \ell)$$

Proof: We shall prove the statement of the theorem by induction on $i \geq 0$.
Base Case : $(i = 0)$ It is easy to observe that $\delta(v_0, f_\ell(v_0)) = r_\ell(v_0)$. Moreover $\sigma()$ is always non-negative, hence the base case is true.
Induction step : Let us suppose that the statement is true for all integers up to $i-1$. We shall prove the validity of the statement for i.

- **For $\ell > t$:**
  Consider the path from v_0 to $f_\ell(v_i)$ formed by concatenating the two shortest paths : $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ and $f_{\ell-1}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner. Using the induction hypothesis, it follows that the length of the shortest path $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ is $k(i-1) + r_{\ell-1}(v_{i-1}) + \sigma(t - \ell + 1)$. Moreover, it follows from Lemma 3.5 that there is a path in the spanner from $f_{\ell-1}(v_{i-1})$ to $f_\ell(v_i)$ consisting of at-most $2\ell - 1 + r_\ell(v_i) - r_{\ell-1}(v_{i-1})$ edges. Hence we can infer that

$$\begin{aligned}
\delta_S(v_0, f_\ell(v_i)) &\leq k(i-1) + r_\ell(v_i) + 2\ell - 1 + \sigma(t-\ell+1) \\
&= ki + r_\ell(v_i) + 2(\ell - t - 1) + \sigma(t - \ell + 1) \qquad \{\text{substituting } k = 2t+1\} \\
&= ki + r_\ell(v_i) - 2(t - \ell + 1) + \sigma(t-\ell+1) = ki + r_\ell(v_i) + \sigma(t - \ell)
\end{aligned}$$

- **For $\ell \leq t$:** There are two cases here.
  v_i is ℓ-unclustered : In this case, consider the path formed by concatenating the shortest paths $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ and $f_{\ell-1}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner. Using the induction hypothesis, it follows that the length of the path $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ is at most $k(i-1) + r_{\ell-1}(v_{i-1}) + \sigma(t-\ell+1)$. Moreover, since v_i is ℓ-unclustered, it follows from Lemma 3.3 that the path $f_{\ell-1}(v_{i-1}) \rightsquigarrow v_i$ is of length at most $2\ell - 1 - r_{\ell-1}(v_{i-1})$. Hence, we can infer that

$$\begin{aligned}
\delta_S(v_0, f_\ell(v_i)) &\leq k(i-1) + r_\ell(v_i) + 2\ell - 1 + \sigma(t - \ell + 1) \qquad \{\text{note that } r_\ell(v_i) = 0\} \\
&= ki + r_\ell(v_i) - 2(t - \ell + 1) + \sigma(t - \ell + 1) \qquad \{\text{substituting } k = 2t+1\} \\
&= ki + r_\ell(v_i) + \sigma(t - \ell)
\end{aligned}$$

  v_i is ℓ-clustered : In this case, Lemma 3.4 implies that there is a path $f_{k-1-\ell}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner with length at most $2k - 2\ell - 1 + r_\ell(v_i) - r_{k-1-\ell}(v_{i-1})$. Moreover, using the induction hypothesis the shortest path from v_0 to $f_{k-1-\ell}(v_{i-1})$ in the spanner is of length $k(i-1) + r_{k-\ell-1}(v_{i-1}) + \sigma(t - k + \ell + 1)$. Hence we can infer that

$$\begin{aligned}
\delta_S(v_0, f_\ell(v_i)) &\leq ki + r_\ell(v_i) + k - 1 - 2\ell + \sigma(-(t - \ell)) \qquad \{\text{substituting } k = 2t+1\} \\
&= ki + r_\ell(v_i) + 2(t - \ell) + \sigma(-(t-\ell)) \\
&= ki + r_\ell(v_i) + 2(t-\ell) + \sigma(t - \ell - 1) \qquad \{\text{ since } \sigma(-j) = \sigma(j-1) \ \ \forall j\} \\
&= ki + r_\ell(v_i) + \sigma(t - \ell)
\end{aligned}$$

□

Corollary 3.7 *The spanner computed by our algorithm is a $(k, k-1)$-spanner for any odd value of k.*

Proof: Let $\langle u = v_0, v_1, \cdots v_q = v\rangle$ be any path of length q between any two vertices $u, v \in V$ in the graph. If v is t-clustered then it follows from Theorem 3.6 that there is a path of length $kq + t$ from u to $c_t(v)$. Together with the path from $c_t(v)$ to v of length at most t, we have a path of length $kq + 2t = kq + k - 1$ from u to v in S. If v is t-*un*clustered then $f_t(v) = v$ and $r_t(v) = 0$, implying a path of length kq from u to v. Hence the spanner S is indeed a $(k, k-1)$-spanner for any odd value of k. □

Analysis of the stretch for even value of k

We shall now prove the following theorem for the case when k is even. Note that $k = 2t$ since k is even.

Theorem 3.8 *Let $\langle u = v_0, v_1, \cdots v_q = v\rangle$ be a path between any $u, v \in V$ in the given graph, and S be the spanner computed by our algorithm for an even integer k. Then for each $0 \leq i \leq q$ and integer $\ell \in [0, 2t-1]$,*

$$\delta_S(v_0, f_\ell(v_i)) \leq ki + r_\ell(v_i) + \sigma(t - \ell) + \ell - t - (i + t - \ell) \bmod 2$$

Proof: We shall prove the statement of theorem by induction on $i \geq 0$.
Base Case : $(i = 0)$ It is easy to observe that $\delta(v_0, f_\ell(v_0)) = r_\ell(v_0)$. Moreover $\sigma(j) - j - (j \bmod 2)$ is always non-negative, hence the base case is true.
Induction step : Let us suppose that the statement is true for all integers up to $i - 1$. We shall prove the validity of the statement for i.

- **For** $\ell \geq t$ **:**
Consider the path from v_0 to $f_\ell(v_i)$ formed by concatenating the two shortest paths : $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ and $f_{\ell-1}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner. It follows from the induction hypothesis that the shortest path from from v_0 to $f_{\ell-1}(v_{i-1})$ in the spanner is of length $k(i-1) + r_{\ell-1}(v_{i-1}) + \sigma(t - \ell + 1) + \ell - 1 - t - (i + t - \ell - 2) \bmod 2$. Moreover using Lemma 3.5, there is a path in the spanner from $f_{\ell-1}(v_{i-1})$ to $f_\ell(v_i)$ of length at most $2\ell - 1 + r_\ell(v_i) - r_{\ell-1}(v_i - 1)$. Hence we can infer that

$$\begin{aligned} \delta_S(v_0, f_\ell(v_i)) &\leq k(i-1) + r_\ell(v_i) + \sigma(t - \ell + 1) + \ell - 1 - t - (i + t - \ell) \bmod 2 + 2\ell - 1 \\ &\leq ki + r_\ell(v_i) + \sigma(t - \ell + 1) - 2(t - \ell + 1) + \ell - t - (i + t - \ell) \bmod 2 \\ &= ki + r_\ell(v_i) + \sigma(t - \ell) + \ell - t - (i + t - \ell) \bmod 2 \end{aligned}$$

- **For** $\ell < t$ **:** There are two cases here.
v_i is ℓ-unclustered : In this case, consider the path formed by concatenating the shortest paths $v_0 \rightsquigarrow f_{\ell-1}(v_{i-1})$ and $f_{\ell-1}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner. Using the induction hypothesis, it follows that there is a path in the spanner from v_0 to $f_{\ell-1}(v_{i-1})$ of length at most $k(i-1) + r_{\ell-1}(v_{i-1}) + \sigma(t - \ell + 1) + \ell - 1 - t - (i + t - \ell - 2) \bmod 2$. Moreover, since v_i is ℓ-unclustered, it follows from Lemma 3.3 that the path $f_{\ell-1}(v_{i-1}) \rightsquigarrow v_i$ is of length at most $2\ell - 1 + r_\ell(v_i) - r_{\ell-1}(v_{i-1})$. Hence we can infer that

$$\begin{aligned} \delta_S(v_0, f_\ell(v_i)) &\leq k(i-1) + r_\ell(v_i) + \sigma(t - \ell + 1) + \ell - 1 - t + 2\ell - 1 - (i + t - \ell) \bmod 2 \\ &\leq ki + r_\ell(v_i) + \sigma(t - \ell + 1) - 2(t - \ell + 1) + \ell - t - (i + t - \ell) \bmod 2 \\ &= ki + r_\ell(v_i) + \sigma(t - \ell) + \ell - t - (i + t - \ell) \bmod 2 \end{aligned}$$

v_i is ℓ-clustered : In this case, Lemma 3.4 implies that there is a path $f_{k-1-\ell}(v_{i-1}) \rightsquigarrow f_\ell(v_i)$ in the spanner with length at most $2k - 2\ell - 1 + r_\ell(v_i) - r_{k-1-\ell}(v_{i-1})$. Moreover, using the induction hypothesis the shortest path from v_0 to $f_{k-1-\ell}(v_{i-1})$ in the spanner is of length at most $k(i-1) + r_{k-1-\ell}(v_{i-1}) + \sigma(t - k + \ell + 1) + k - \ell - 1 - t - (i + t + \ell) \bmod 2$. Hence there is a path from v_0 to $f_\ell(v_i)$ in the spanner passing through $f_{k-1-\ell}(v_{i-1})$ with length

$$\begin{aligned} \delta_S(v_0, f_\ell(v_i)) &\leq k(i-1) + r_\ell(v_i) + \sigma(t - k + \ell + 1) + 3k - 3\ell - 2 - t - (i + t + \ell) \bmod 2 \\ &= ki + r_\ell(v_i) + \sigma(-(t - \ell - 1)) + (4t - 4\ell - 2) + \ell - t - (i + t - \ell) \bmod 2 \\ &= ki + r_\ell(v_i) + \sigma(t - \ell - 2) + (4t - 4\ell - 2) + \ell - t - (i + t - \ell) \bmod 2 \\ &= ki + r_\ell(v_i) + \sigma(t - \ell) + \ell - t - (i + t - \ell) \bmod 2 \end{aligned}$$

□

Corollary 3.9 *The spanner computed by our algorithm is a $(k, k-1)$-spanner for any even value of k.*

Proof: Let $\langle u = v_0, v_1, \cdots, v_q = v \rangle$ be the shortest path between u and v in the given graph. There are two cases :

- q **is odd :** Consider the shortest path from $u = v_0$ to v in the spanner passing through $f_t(v)$. Theorem 3.8 implies that there is a path in the spanner S from u to $f_t(v)$ of length at most $kq + r_t(v) - 1$. Thus there is a path in S from u to v of length $kq + 2r_t(v) - 1$. Since $r_t(v) \leq t = k/2$, therefore,

$$\delta_S(u, v) \leq kq + 2.k/2 - 1 = kq + k - 1$$

- q **is even :** Consider the shortest path from $u = v_0$ to v in the spanner passing through $f_{t-1}(v)$. Theorem 3.8 implies that there is a path in the spanner S from u to $f_{t-1}(v)$ of length at most $kq + r_{t-1}(v)$. Hence there is a path in S from u to v passing through $f_{t-1}(v)$ with length $kq + 2r_{t-1}(v) = kq + k - 2$. Thus

$$\delta_S(u, v) \leq kq + k - 2$$

Hence the spanner is indeed a $(k, k-1)$-spanner for even k too. $\square$

Theorems 3.6 and 3.8 together imply that S is a $(k, k-1)$-spanner for any positive integer k. Thus Theorem 3.2 is completely proved.

3.1 Implementation in other models of computations

Our algorithm (Figure 8) for $(k, k-1)$-spanners can be adapted quite easily to other models of computation. The complexity of each of these algorithms is close to optimal under relevant measures.

- In the external memory model [5] and the cache oblivious model [36], a $(k, k-1)$-spanner of $O(kn^{1+1/k})$ size can be computed using the same number of I/O operations as that of sorting km items. Sorting is one of the most primitive tasks in both models and optimal algorithms are known.
- In the CRCW PRAM model [42], a $(k, k-1)$-spanner of expected size $O(kn^{1+1/k})$ can be computed in $O(k \log^* n)$ time and $O(km)$ work. The algorithm employs primitive parallel subroutines like computing the smallest element, semisorting, and multiset hashing.
- In the synchronous distributed model [45], a $(k, k-1)$-spanner of expected $O(kn^{1+1/k})$ size can be computed in $O(k)$ rounds with total message volume $O(k^2 m)$.

In order to convey to the reader the ease with which the algorithm is adapted in these models, we provide below the execution of the $(k, k-1)$-spanner construction in distributed environment. Adaptation of the algorithm in the external-memory and CRCW PRAM environment as mentioned above is similar to the adaptation of $(2k-1)$-spanner algorithm of Baswana and Sen [12] in these models.

3.2 Distributed algorithm for $(k, k-1)$-spanner

Rules R3 and R4 of our algorithm (Figure 8) can be executed in $O(k)$ rounds of communication in a distributed network, and, using the randomized algorithm from Figure 7, Rules R1 and R2 can also be executed in $O(k)$ rounds. The main result in this section is Theorem 3.10. Before we prove it, let us elaborate a little on our model of distributed computation. In a synchronized distributed network the nodes of the network solve a problem by exchanging messages in discrete *rounds*. In each round one message may be sent across each link in each direction. We are interested in three measures: the number of rounds, the maximum length of any message sent (measured in units of $O(\log n)$ bits) and the total length of all messages sent. Clearly any protocol requiring R rounds, maximum message length L and total volume V can be converted to one with parameters RL/U, U, and V, for any any $U \geq 1$. That is, Theorem 3.10 can be adapted to any synchronized network with a fixed maximum message length.

Theorem 3.10 *In a synchronized distributed network G, a $(k, k-1)$-spanner of G whose expected size is $O(kn^{1+1/k})$, can be constructed in $O(k)$ rounds of communication. The total message volume is $O(k^2 m)$ and the maximum message length is $O(n^{1/2+1/2k})$.*

Proof: We compute the clusters $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k = \emptyset$ using the randomized algorithm from Figure 7. Each vertex is the center of its cluster in $\mathcal{C}_0 = \{\{v\} \, : \, v \in V(G)\}$. With probability $n^{-1/k}$ each center in $\mathcal{C}_i$ declares itself to also be a center in $\mathcal{C}_{i+1}$. These random choices are made *before* the first round of communication.

After $\mathcal{C}_i$ is computed, every vertex tells its neighbors whether it is clustered in $\mathcal{C}_i$ and if it is, the identity of its center in $\mathcal{C}_i$ and the highest $j \geq i$ for which that center is also a center in $\mathcal{C}_j$. For each vertex w that has a neighbor v already clustered in $\mathcal{C}_{i+1}$, w declares (w, v) to be a spanner edge (Rule R1) and declares its center in $\mathcal{C}_{i+1}$ to be that of v. Every vertex w that did not become clustered in $\mathcal{C}_{i+1}$ declares one edge from $E(w, C)$ to be in the spanner (Rule R2), for each $C \in \mathcal{C}_i$ adjacent to w. Rules R1 and R2 require $k-1$ rounds of communication, plus one more to let clustered vertices in $\mathcal{C}_{k-1}$ inform their neighbors of this fact. Each message sent so far has unit length.

Once $\mathcal{C}_0, \ldots, \mathcal{C}_{k-1}$ are computed, we implement Rules R3 and R4. Consider Rule R3, a fixed $i \geq 0$, and a fixed cluster $C \in \mathcal{C}_{(k-1)/2-i}$.[10] Rule R1 has created a tree T of spanner edges rooted at the center of C. This tree is used to inform the center of C of all incident clusters in $\mathcal{C}_{(k-1)/2+i}$, and for each such cluster,

[10] If k is odd then i is an integer. For even k, i is a half-integer.

one connecting edge. Once the center decides which edges to select for Rule R3 it broadcasts its choices back through T. The number of rounds for Rule R3 is clearly $O(k)$. The maximum necessary message length (for fixed i) is $|\mathcal{C}_{(k-1)/2+i}|$ since duplicate edges connecting the same clusters can be ignored. With high probability $|\mathcal{C}_{(k-1)/2+i}| = O(n^{1/2+1/2k-i/k})$. Summing over $i \geq 0$, the maximum message length is bounded by $O(n^{1/2+1/2k})$. For even k, i is always at least $1/2$, so in this case the maximum message length is $O(\sqrt{n})$. The total message volume for Rule R3 is $O(k^2 m)$ since each edge can participate for $k/2$ values of i and for each, contribute $O(k)$ units of message volume. The implementation and analysis of Rule R4 is very similar to R3.

□

4 Conclusion

The main existential question in the field of spanners is whether, for any given size bound $O(n^{1+\epsilon})$, there exist additive $\beta(\epsilon)$-spanners and if not, which additive spanners do exist? For $\epsilon < 1/3$ our additive spanners are the best known, but they have additive distortion that depends heavily on n. In this paper we introduced a general construction technique that might help to resolve the question of additive spanners for general graphs.

In Section 3 we addressed the problem of computationally efficient spanner constructions and gave partial answers to two problems of practical significance: what is the highest quality spanner that can be constructed in linear time? and which spanners can be constructed distributively in $O(1)$ rounds? It seems implausible that any additive or $(1+\epsilon, \beta)$-spanners admit such efficient constructions.

One promising direction for future research is to develop approximate distance oracles for unweighted graphs whose distortion improves with distance. The ultimate goal would be to have oracles with constant additive distortion, though any improvement over the purely multiplicative distortion of [54, 13, 43, 11] would be a start. For example, there is no known $(3-\epsilon, \beta)$-distance oracle with size $O(n^{3/2})$ whose query time is reasonably fast.

Acknowledgment. We would like to thank Uri Zwick, Mikkel Thorup, and Michael Elkin for some helpful comments on an earlier draft of this paper.

References

[1] I. Abraham, Y. Bartal, H. T.-H. Chan, K. Dhamdhere, A. Gupta, J. M. Kleinberg, O. Neiman, and A. Slivkins. Metric embeddings with relaxed guarantees. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 83–100, 2005.

[2] I. Abraham, Y. Bartal, and O. Neiman. Advances in metric embedding theory. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 271–286, 2006.

[3] I. Abraham, Y. Bartal, and O. Neiman. Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages ??–??, 2007.

[4] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: upper bounds. In *Proc. 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 217–224, 2006.

[5] A. Aggarwal and J. S. Vitter. The input/output complexity of sorting and related problems. *Comm. ACM*, 31(9):1116–1127, 1988.

[6] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

[7] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002.

[8] N. Alon, R. M. Karp, D. Peleg, and D. B. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

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

[10] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 184–193, 1996.

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

[12] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *J. Random Structures and Algs.* to appear. Extended abstract published in Proc. 30th Annual Int'l Colloq. on Automata, Languages and Programming.

[13] S. Baswana and S. Sen. A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Proc. 30th Ann. Intl. Colloq. on Automata, Languages and Programming (ICALP)*, 2003.

[14] S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in $O(n^2 \log n)$ time. In *Proc. 15th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 271–280, 2004.

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

[16] M. Bădoiu, P. Indyk, and A. Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[17] H. T.-H. Chan, M. Dinitz, and A. Gupta. Spanners with slack. In *Proc. 14th Annual European Symposium on Algorithms (ESA)*, pages 196–207, 2006.

[18] H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–78, 2006.

[19] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 762–771, 2005.

[20] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM J. Comput.*, 28:210–236, 1998.

[21] E. Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest-paths. *J. ACM*, 47:132–166, 2000.

[22] D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 660–669, 2005.

[23] L. J. Cowen. Compact routing with minimum stretch. *J. Algor.*, 28:170–183, 2001.

[24] L. J. Cowen and C. G. Wagner. Compact roundtrip routing in directed networks. *J. Algor.*, 50(1):79–95, 2004.

[25] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

[26] M. Elkin. Private communication. 2004.

[27] M. Elkin. Computing almost shortest paths. *Transactions on Algorithms*, 1(2):283–323, 2005.

[28] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng. Lower-stretch spanning trees. In *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 494–503, 2005.

[29] M. Elkin and D. Peleg. $(1+\epsilon, \beta)$-Spanner constructions for general graphs. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, 2001.

[30] M. Elkin and D. Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631, 2004.

[31] M. Elkin and D. Peleg. Approximating k-spanner problems for $k \geq 2$. *Theoretical Computer Science*, 337(1–3):249–277, 2005.

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

[33] Y. Emek and D. Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 261–270, 2004.

[34] P. Erdős. Extremal problems in graph theory. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963), pages 29–36. Publ. House Czechoslovak Acad. Sci., Prague*, 1963.

[35] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[36] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *Proc. 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–298, 1999.

[37] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002.

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

[39] S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.

[40] P. Indyk. Algorithmic aspects of low-distortion geometric embeddings, 2001. Tutorial, 42nd Annual IEEE Symposium on Foundations of Computer Science, URL: http://theory.lcs.mit.edu/~indyk/tut.html.

[41] P. Indyk and J. Matousek. Low-distortion embedding of finite metric spaces. In *Handbook of Discrete and Computational Geometry, Second Edition*. CRC Press, 2004.

[42] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In *Handbook of Computer Science*, pages 869–942. MIT Press/Elsevier, 1990.

[43] M. Mendel and A. Naor. Ramsey partitions and proximity data structures. In *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages ??–??, 2006.

[44] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[45] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[46] D. Peleg and A. A. Schaffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

[47] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18:740–747, 1989.

[48] D. Peleg and E. Upfal. A trade-off between space amd efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

[49] L. Roditty, M. Thorup, and U. Zwick. Roundtrip spanners and roundtrip routing in directed graphs. In *Proc. 13th Ann. ACM-SIAM Symposium On Discrete Algorithms (SODA)*, pages 844–851, 2002.

[50] L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proc. 32nd Int'l Colloq. on Automata, Languages, and Programming (ICALP)*, pages 261–272, 2005.

[51] L. Roditty and U. Zwick. On dynamic shortest paths problems. In *Proc. 12th Annual European Symposium on Algorithms (ESA)*, pages 580–591, 2004.

[52] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2004.

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

[54] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[55] M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 802–809, 2006.

[56] 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.

[57] D. Woodruff. Lower bounds for additive spanners, emulators, and more. In *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages ??–??, 2006.

[58] J. Xiao, L. Liu, L. Xia, and T. Jiang. Fast elimination of redundant linear equations and reconstruction of recombination-free Mendelian inheritance on a pedigree. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages ??–??, 2007.