New Constructions of (α, β)-Spanners and Purely Additive Spanners*

Surender Baswana   Telikepalli Kavitha   Kurt Mehlhorn   Seth Pettie†

Abstract

An (α, β)-spanner of an unweighted graph G is a subgraph H that approximates distances in G in the following sense. For any two vertices u, v: $\delta_H(u, v) \leq \alpha\delta_G(u, v) + \beta$, where δ_G is the distance w.r.t. G. It is well known that there exist (multiplicative) $(2k-1, 0)$-spanners of size $O(n^{1+1/k})$ and that there exist (purely additive) $(1, 2)$-spanners of size $O(n^{3/2})$. However no other $(1, O(1))$-spanners are known to exist.

In this paper we develop a couple new techniques for constructing (α, β)-spanners. The first result is a purely additive $(1, 6)$-spanner of size $O(n^{4/3})$. Our 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. This general approach should lead to new spanner constructions.

The second result is a truly simple linear time construction of $(k, k-1)$-spanners with size $O(n^{1+1/k})$. In a distributed network the algorithm terminates in a *constant* number of rounds and has expected size $O(n^{1+1/k})$. The new idea here is primarily in the analysis of the construction. We show that a few simple and local rules for picking spanner edges induce seemingly coordinated global behavior.

1 Introduction

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

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

When $\beta = 0$ this definition reverts to the multiplicative α-spanner [PS89, PU89a]. In an (α, β)-spanner a single edge may be stretched by as much as $\alpha + \beta$. However the average stretch in a long path is as low as α. We call a spanner *additive* if $\alpha = 1$ and *purely additive* if $\alpha = 1$ and $\beta = O(1)$.

Spanners (and related structures) are useful in many contexts. They are the basis of space-efficient routing tables that guarantee nearly shortest routes [TZ01b, RTZ02, Cow01, CW04, PU89b], schemes for simulating synchronized protocols in unsynchronized networks [PU89a], parallel and distributed algorithms for computing approximate shortest paths [Coh98, Coh00, Elk01], and algorithms for constructing so-called approximate distance oracles [TZ01a, BS04].

Spanner constructions are to be evaluated along three worst-case measures: *approximation quality*, *space*, and *construction time*. One aims for small values of α and β, small size of the spanner, and efficient construction, ideally a linear time construction. Furthermore, since spanners have important applications in distributed networks it is desirable to have the spanner construction *itself* be distributed.

Erdős [Erd63] conjectured that there exist graphs with $\Omega(n^{1+1/k})$ edges and girth $2k+2$, where n is the number of vertices, k is an integer constant, and *girth* is the length of the shortest cycle. Removing any edge from such a graph increases the distance between its endpoints from 1 to at least $2k+1$, which implies that any (α, β)-spanner with $\alpha + \beta \leq 2k$ must have space $\Omega(n^{1+1/k})$. The girth conjecture is settled for only $k = 1, 2, 3$, and 5 [Wen91].

1.1 Comparison with previous results

Althöfer et al. [ADD+93] were the first to discover an $O(n^{1+1/k})$-size $(2k-1, 0)$-spanner, which is optimal assuming the girth conjecture. Their construction took $O(mn^{1+1/k})$ time, where m is the number of edges. This time bound was improved several times, most recently by Baswana and Sen [BS03], who gave a randomized linear time $(2k-1, 0)$-spanner construction. Both [ADD+93, BS03] work on weighted graphs. Better α-β tradeoffs are known for unweighted graphs. Elkin and Peleg [EP01] exhibit

*Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT). Address: Max Planck Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany. Email: {sbaswana,kavitha,mehlhorn,pettie}@mpi-sb.mpg.de.

†Supported by an Alexander von Humboldt postdoctoral fellowship.

(α, β)	Size	Time	Notes
$(2k-1,\, 0)$	$O(n^{1+1/k})$	$O(m)$	[BS03],[HZ96]
$(k-1,\, 2k-O(1))$	$O(n^{1+1/k})$	$O(mn^{1-1/k})$	[EP01]
$(1+\epsilon,\, 4)$	$O(\epsilon^{-1}n^{4/3})$	$O(mn^{2/3})$	[EP01]
$(1+\epsilon,\, \beta(\delta,\epsilon))$	$O(\beta n^{1+\delta})$	$O(mn^{\delta})$	[EP01, Elk01], $\delta < 1/2$
$(1,\, 2)$	$O(n^{3/2})$	$O(m\sqrt{n})$	[EP01],[ACIM99]
	$\tilde{O}(n^{3/2})$	$\tilde{O}(n^2)$	[DHZ00]
$(1,\, 2^{\delta^{-1}} n^{(1-\delta)(\lceil\delta^{-1}\rceil-2)/(\lceil\delta^{-1}\rceil-1)})$	$O(n^{1+\delta})$	poly(n)	[BCE03], $\delta < 1/2$
$(1,\, n^{1/3})$	$O(n^{4/3})$	poly(n)	[BCE03]
$(k,\, k-1)$	$O(n^{1+1/k})$	$O(m)$	**this paper**
$(1,\, 6)$	$O(n^{4/3})$	$O(mn)$	**this paper**
$(1, n^{1-3\delta})$	$O(n^{1+\delta})$	poly(n)	**this paper,** $\delta < 1/3$

Figure 1: State-of-the-art in (α, β)-spanners. Some slower constructions are omitted. The parameters k, δ, ϵ are positive; k is an integer and both δ, ϵ are reals.

a $(k-1, 2k-O(1))$-spanner[1] of size $O(n^{1+1/k})$ and Baswana and Sen [BS04] give a linear time constructible $(2,1)$-spanner of size $O(n^{3/2})$.

The first — and to date, only — purely additive spanner was discovered by Aingworth et al. [ACIM99]. They showed that every graph has a polynomial-time constructible $(1,2)$-spanner with size $\tilde{O}(n^{3/2})$. This construction was slightly improved by Dor et al. [DHZ00] and Elkin and Peleg [EP01]. Bollobás et al. [BCE03] showed that there exist a spectrum of additive spanners where the error depends on n. For space bound $O(n^{1+1/k})$ the [BCE03] construction gives a $(1, 2^k n^{1-2/k})$-spanner. (However when k is not an integer the additive stretch is slightly more than $2^k n^{1-2/k}$.) Elkin and Peleg [EP01] proved that for any superlinear size bound there exist *almost* purely additive spanners. In particular, there are $(1+\epsilon, \beta)$-spanners of size $O(\beta n^{1+\delta})$, where $\beta(\delta, \epsilon)$ is independent of n but grows superpolynomially in δ^{-1} and ϵ^{-1}. In [EP01] it is also shown that there are $(1+\epsilon, 4)$-spanners of size $O(\epsilon^{-1}n^{4/3})$. Elkin and Zhang [Elk01, EZ04] went on to develop a distributed $(1+\epsilon, \beta)$-spanner construction that takes $O(n^{\delta})$ rounds and $O(mn^{\delta})$ communication.

We remark that all the above constructions take polynomial time and that only the [BS03, BS04] constructions take linear time. An unpublished algorithm of Halperin and Zwick [HZ96] also takes linear time.

Figure 1 summarizes our results and puts them into context.

Our Results. In Section 2 we present a construction for $(1,6)$-spanners of size $O(n^{4/3})$, which is the first purely additive spanner of size $o(n^{3/2})$. In general terms our algorithm defines *cost* and *value* functions over paths in the graph, which are w.r.t. our spanner under construction. We examine each of the $\binom{n}{2}$ shortest paths in turn. If the value of the path exceeds its cost we buy the path (include it in the spanner). We show that if the algorithm refuses to buy a path then the existing spanner already approximates the distance between its endpoints, to an additive error of 6. We are optimistic that this framework, based on costs, values, and affordable paths, could lead to further purely additive spanners. We are already able to show that our method gives substantially better additive spanners. By adjusting the path valuation function (and other parameters) our $(1,6)$-spanner construction produces $(1, n^{1-3\delta})$-spanners of size $O(n^{1+\delta})$, for any $\delta < 1/3$. This is a polynomial improvement over the [BCE03] construction, whose additive stretch is never better than $2^{\delta^{-1}} n^{1-2\delta}$.

In Section 3, we give a simple linear-time construction for $(k, k-1)$-spanners of size $O(n^{1+1/k})$, which, assuming the girth conjecture, cannot be unilaterally improved. Our construction uses only local adjacency information and computes *no distances or shortest paths* whatsoever. In a synchronized distributed network the algorithm terminates in a *constant* number of rounds ($O(k)$), has linear communication complexity, and produces a spanner whose expected size is $O(n^{1+1/k})$. This distributed implementation makes our algorithm well suited to the applications cited earlier. Perhaps the most interesting feature of our $(k, k-1)$-spanner construction is how it guarantees good approximate distances, even for distant vertices, using only simple and loosely coordinated rules for deciding which edges to put in the

[1] With numerous refinements for odd k, even k, and longer distances.

spanner.

1.2 Variations on Spanners Spanners can be generalized to directed graphs by defining $\delta(u,v)$ to be the *roundtrip* distance from u to u via v. Roundtrip spanners were studied by Cowen and Wagner [CW04] and Roditty et al. [RTZ02]. An (α,β)-*Steiner spanner* is just like an (α,β)-spanner with two differences: it need not be a subgraph of the original graph and it can have *weighted* edges, even if the original graph is unweighted. (Cohen's [Coh00] *hop sets* are a variation on Steiner spanners.) Dor et al. [DHZ00] constructed a $(1,4)$-Steiner spanner with size $O(n^{4/3})$. Bollobás et al. [BCE03] defined a D-(Steiner) preserver as a (not necessarily) subgraph that preserves *exact* distances for all pairs of vertices at distance at least D in the original graph. They proved tight bounds of $\Theta(n^2/D)$ on the size of D-(Steiner) preservers, with slightly better results for *directed* Steiner preservers. Thorup and Zwick [TZ01a] introduced the idea of an approximate distance oracle, a compact data structure of size $O(n^{1+1/k})$ that answers distance queries in constant time to within a multiplicative error of $2k-1$. Baswana and Sen [BS04] gave faster constructions of distance oracles in unweighted graphs. As a natural byproduct, the constructions of [TZ01a, BS04] compute $(2k-1,0)$-spanners of size $O(n^{1+1/k})$. One obvious use for spanners is computing (α,β)-approximate shortest paths in sub-linear time (simply run BFS on an (α,β)-spanner rather than the original graph.) This method can be improved when solving certain *all-pairs* approximate shortest path problems; see [DHZ00, CZ01, BGS04].

2 A New Purely Additive Spanner

Our construction for $(1,6)$-spanners works in two phases, the first of which involves standard 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 *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 incident to unclustered vertices. In Section 3 we give two linear time algorithms for constructing $\mathcal{C}$ and H_0 such that $|H_0| \le n^{4/3} + n$. Let $\mathcal{C}(v)$ be the cluster containing v, if any, and if D is a subgraph let $\mathcal{C}(D)$ be the set of clusters incident to any vertex in D.

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 $P = \langle u_0, \ldots, u_q\rangle$, where both u_0 and u_q are clustered, there exists a short path Q in the spanner from u_0 to u_q that passes through some $C \in \mathcal{C}(P)$. We guarantee, in particular, that the portions of Q from $\mathcal{C}(u_0) \rightsquigarrow C$ and $C \rightsquigarrow \mathcal{C}(u_q)$ are no longer than their counterparts in P. Property 2.1 formalizes this idea, and Lemma 2.1 states that any subgraph with this property is a $(1,6)$-spanner.

PROPERTY 2.1. *A subgraph* $H \supseteq H_0$ *is* contented *if for any two* clustered *vertices* u_0, u_q*, there exists a shortest path* $P = \langle u_0, \ldots, u_q\rangle$ *in* G *and a* $C \in \mathcal{C}(P)$ *such that:*

$$\delta_H(\mathcal{C}(u_i), C) \le \delta_P(\mathcal{C}(u_i), C) \quad \textit{for } i = 0 \textit{ and } i = q$$

LEMMA 2.1. *Any contented subgraph of* G *is also a* $(1,6)$*-spanner of* G.

Proof. Let H be the contented subgraph, u_0 and u_q be two arbitrary vertices and let $P = \langle u_0, \ldots, u_q\rangle$ be the shortest path between them. We can assume w.l.o.g. that both u_0 and u_q are clustered; otherwise one of (u_0, u_1) or (u_{q-1}, u_q) are in $H_0 \subseteq H$ and we could instead consider a path shorter than P. So, assuming u_0 and u_q are clustered, let $C \in \mathcal{C}(P)$ be the cluster guaranteed by the contentedness of H — see Figure 2. We can bound the distance from u_0 to u_q in H as:

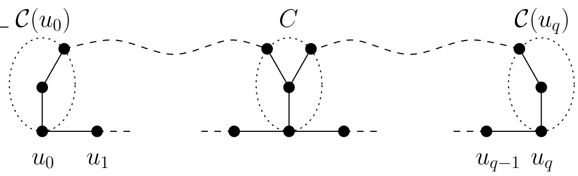

Figure 2: The clusters $\mathcal{C}(u_0)$, C, and $\mathcal{C}(u_q)$ indicated by ovals. The shortest inter-cluster paths in H are indicated by dashed curves.

$$\begin{aligned}
&\delta_H(u_0, u_q) \\
&\quad\le \operatorname{diam}_H(\mathcal{C}(u_0)) + \delta_H(\mathcal{C}(u_0), C) + \operatorname{diam}_H(C) \\
&\qquad + \delta_H(C, \mathcal{C}(u_q)) + \operatorname{diam}_H(\mathcal{C}(u_q)) \\
&\quad\le 2 + \delta_P(\mathcal{C}(u_0), C) + 2 + \delta_P(C, \mathcal{C}(u_q)) + 2 \\
&\quad\le \delta_G(u_0, u_q) + 6
\end{aligned}$$

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

In phase two we find a subgraph $H_0 \cup P_1 \cup P_2 \cup \cdots \cup P_k$ where $P_1, \ldots, P_k$ are the *purchased* shortest paths. The phase two algorithm is very simple. We evaluate some shortest path between each pair of vertices, and, based on certain evolving *cost* and *value* functions, we purchase the path if twice its value exceeds its cost. See Figure 3.

The following cost and value functions are defined with respect to an edge set H, which is our spanner under construction.

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

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

Notice in the definition of value(D) that $\delta_D(C, C') = \infty$ unless both C and C' intersect D.

In other words, 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 algorithm for Phase 2 is given in Figure 3. Let $\mathcal{P}$ be a collection 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 $\langle u_1, u_2, u_3 \rangle \subseteq P \in \mathcal{P}$, 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 number of non-H_0 edges in $\mathcal{P}$ since $\langle u_1, u_2, u_3 \rangle \subseteq H_0$.)

$H := H_0$ {edges chosen in clustering}
For each path $P \in \mathcal{P}$
If $2 \cdot \text{value}(P) \geq \text{cost}(P)$
then $H := H \cup P$ {purchase the path P}
Return H

Figure 3: The Phase 2 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 2.2 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 2.3 we relate the cost of a path to the number of clusters intersecting it. Finally, and most importantly, Lemma 2.4 shows that if any shortest path is too expensive to be purchased then the existing spanner edges already guarantee a path with additive stretch 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 in Line 3 of Figure 3.

Lemma 2.2. $\sum_{i \geq 1} \text{value}(P_i) \leq 5\binom{|\mathcal{C}|}{2} < \frac{5}{2} n^{4/3}$

Proof. Let $H_i = H_0 \cup P_1 \cup \cdots \cup P_i$ be the spanner after purchasing i shortest paths. Let C, C' be two arbitrary clusters and let $P(C, C') = \{P_{j_1}, P_{j_2}, \ldots, P_{j_r}\}$ be those paths such that

$$\delta_{P_{j_i}}(C, C') < \delta_{H_{j_i - 1}}(C, C')$$

By the definition of the value function $\sum_{i \geq 1} \text{value}(P_i) = \sum_{\{C,C'\} \subseteq \mathcal{C}} |P(C, C')|$. 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 $\delta_G(C, C') \leq |P_{j_r}| < |P_{j_{r-1}}| < \cdots < |P_{j_1}| \leq \delta_G(C, C') + 4$.

Lemma 2.3. *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 $P = \langle u_0, \ldots, u_q \rangle$ and let $C_0, C_1 \in \mathcal{C}(P)$ be the (distinct) first and last clusters intersecting P. Define P' as the minimal subpath of P such that $\mathcal{C}(P') = \mathcal{C}(P)$. (This means that if C_0 or C_1 have 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 $H \supseteq H_0$ are bounded by $|\mathcal{C}(P')| - 1$ and $|\mathcal{C}(P')| - 2$.

Lemma 2.4. *H is contented.*

Proof. Let $P = \langle u_0, \ldots, u_q \rangle \in \mathcal{P}$ be the shortest path from u_0 to u_q 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_0 and u_q are clustered and that $\mathcal{C}(u_0) \neq \mathcal{C}(u_q)$. Let $P' \subseteq P$ be the subpath guaranteed by Lemma 2.3. The case when P' is included in H is also trivial. Thus we have the following inequalities:

$$(2.1) \qquad 2 \cdot \text{value}(P') < \text{cost}(P') \leq 2 \cdot |\mathcal{C}(P')| - 3$$

where the first inequality follows from the fact that P' was not included in H and the second from Lemma 2.3. Define A as the set of cluster pairs:

$$A = \left\{ \{C, C'\} \ : \ \begin{array}{l} C \in \{\mathcal{C}(u_0), \mathcal{C}(u_q)\}, \ C' \in \mathcal{C}(P) \\ \text{and } \ \delta_{P'}(C, C') < \delta_H(C, C') \end{array} \right\}$$

By the definition of A we have $|A| \leq \text{value}(P')$ and by the inequalities of Eqn. 2.1 $\text{value}(P') \leq |\mathcal{C}(P')| - 2$. 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

$$\delta_H(\mathcal{C}(u_0), C'') \leq \delta_{P'}(\mathcal{C}(u_0), C'')$$

and

$$\delta_H(C'', \mathcal{C}(u_q)) \leq \delta_{P'}(C'', \mathcal{C}(u_q))$$

Lemma 2.5. $|H| < 6 \cdot n^{4/3} + n$

Proof. One can easily see that $|H| = |H_0| + \sum_{i \geq 1} \mathrm{cost}(P_i)$. By construction we have $|H_0| < n^{4/3} + n$. It follows from Lemma 2.2 that: $\sum_i \mathrm{cost}(P_i) \leq 2 \cdot \sum_i \mathrm{value}(P_i) < 5 \cdot n^{4/3}$

Theorem 2.1. *Given any graph on n vertices and m edges, a $\min\{(1,6), (1+\epsilon, 4)\}$-spanner of size $O(\epsilon^{-1} n^{4/3})$ can be constructed in $O(mn)$ time.*

Proof. Lemmas 2.1, 2.4, and 2.5 establish that H is a $(1,6)$-spanner of the desired size. By augmenting H with a shortest path between every two clusters C, C' such that $\delta_G(C, C') = O(1/\epsilon)$ we obtain a $(1+\epsilon, 4)$-spanner with $O(\epsilon^{-1} n^{4/3})$ edges. (This is roughly the same construction given in [EP01].) We now turn to an efficient construction algorithm. Phase one (clustering) takes linear time. We construct the set of shortest paths $\mathcal{P}$ using n BFS computations, which takes $O(mn)$ time. Phase two, the way it is stated in Figure 3, is much too slow. The bottleneck is evaluating value(P) and maintaining the exact distances in H between all pairs of clusters. We maintain instead the following upperbound $\hat{\delta}$ on the inter-cluster distances:

$$\hat{\delta}(C, C') = \min_i \left\{ \delta_{P_i}(C, C') \ : \ \begin{array}{l} P_i = \langle u_0, \ldots \rangle \text{ and} \\ C = \mathcal{C}(u_0) \end{array} \right\}$$

where i ranges over all paths already included in H. Rather than evaluate the value function, which is very expensive, we refer to value*. Let $P = \langle u_0, \ldots \rangle$. When $\mathcal{C}(u_0)$ does not exist value$^*(P) = 0$. In general,

$$\begin{aligned} &\mathrm{value}^*(P) \\ &= \left| \{ C \in \mathcal{C}(P) \ : \ \delta_P(\mathcal{C}(u_0), C) < \hat{\delta}(\mathcal{C}(u_0), C) \} \right| \end{aligned}$$

The value$^*(\langle u_0, \ldots, u_q \rangle)$ function can be evaluated in $O(q)$ time and in $O(1)$ time if value$^*(\langle u_0, \ldots, u_{q-1} \rangle)$ is given. (Notice that value* is sensitive to the direction of the path.) In Phase 2, the criterion for adding P to H is $4 \cdot \mathrm{value}^*(P) \geq \mathrm{cost}(P)$ rather than $2 \cdot \mathrm{value}(P) \geq \mathrm{cost}(P)$. (This modified algorithm guarantees that $|H| < 11 n^{4/3}$ rather than $6n^{4/3}$. The proof that this algorithm returns a contented subgraph is similar to that given above. The proof concludes by showing that if $P = \langle u_0, \ldots, u_q \rangle$ is not selected for inclusion in H then both value$^*(\langle u_0, \ldots, u_q \rangle)$ and value$^*(\langle u_q, \ldots, u_0 \rangle)$ are strictly less than $|\mathcal{C}(P)|/2$, which implies by the pigeonhole principle that $\delta_P(C, C') \geq \hat{\delta}(C, C') \geq \delta_H(C, C')$ for both $C \in \{\mathcal{C}(u_0), \mathcal{C}(u_q)\}$ and some $C' \in \mathcal{C}(P)$.) We can implement the modified algorithm in $O(n^2)$ time by performing a DFS traversal of the BFS tree rooted at each vertex. The value$^*(P)$ function can be evaluated in constant time because P is an extension of a previously considered shortest path, and if P is to be included in H, updating the $\hat{\delta}$ function can be done in $O(\mathrm{value}^*(P))$ time, which is $O(n^{4/3})$ overall.

Our $(1,6)$-spanner construction can be tweaked to give an additive $(1, n^{1-3\delta})$-spanner of size $O(n^{1+\delta})$, for any $\delta < 1/3$. Rather than choosing $n^{2/3}$ clusters we choose $n^{1-\delta}$. We use the same cost function but substitute value$'(P) = n^{3\delta-1} \cdot \mathrm{value}(P)$ for the value function. The Phase two construction algorithm is the same, except that H is initialized to include all edges chosen in Phase one (H_0) plus any muliplicative spanner of size $O(n^{1+\delta})$. One can easily show the size of the resulting spanner to be $O(|H_0| + n^{3\delta-1} \cdot \binom{\mathcal{C}}{2}) = O(n^{1+\delta})$. Proving the additive stretch is somewhat more complicated. Let $P = \langle u, \ldots, v \rangle$ be a path not purchased by the algorithm and let P' (resp. P'') be the prefix of P (suffix of P) containing exactly $n^{1-3\delta}$ clusters. We show the existence of three not necessarily distinct clusters $C' \in \mathcal{C}(P'), C \in \mathcal{C}(P), C'' \in \mathcal{C}(P'')$ such that

$$\begin{array}{rcl} \delta_H(u, C') & \leq & \delta_P(u, C') + O(n^{1-3\delta}) \\ \delta_H(C', C) & \leq & \delta_P(C', C) \\ \delta_H(C, C'') & \leq & \delta_P(C, C'') \\ \delta_H(C'', v) & \leq & \delta_P(C'', v) + O(n^{1-3\delta}) \end{array}$$

where the first and last inequalities follow because both cost(P') and cost(P'') are $O(n^{1-3\delta})$ and H includes a multiplicative spanner.

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

Before presenting our $(k, k-1)$-spanner construction we show how to construct $(2k-1, 0)$-spanners in deterministic linear time.

The input graph is $G = (V, E)$. For vertex sets C and C', define $E(C, C') = (C \times C') \cap E(G)$ as the set of edges from C to C'. If C is a vertex it is treated as a singleton set. A *cluster* is simply a set of vertices and a *clustering* is a set of clusters.

A vertex is *(un)clustered* in $\mathcal{C}$ if it appears (does not appear) in some cluster of $\mathcal{C}$. For a clustered vertex u, denote by $\mathcal{C}(u)$ *any* cluster containing u in $\mathcal{C}$. The diameter (resp., radius) of a subgraph is the maximum distance (half the maximum distance) between any two vertices in that subgraph. A vertex is adjacent to a cluster if there is an edge connecting it to a vertex in the cluster.

Our constructions in this section are based on a set of $k+1$ clusterings, $\{\{v\} : v \in V(G)\} = \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k = \emptyset$, where $|\mathcal{C}_i| \leq n^{1-i/k}$. 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 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 [HZ96]; however it is the starting point for our $(k, k-1)$-spanner.

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

Proof. (Sketch) 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 w.l.o.g. suppose it is u. By Rule R2 there must be an edge in S from u to $\mathcal{C}_{\ell-1}(v)$ — call it (u, w) — and 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 $n^{1+1/k}$, resp. (This construction is a simplified version of one described by Elkin [Elk04].) Initially $\mathcal{C}_{i+1} = \emptyset$. We define the priority of a cluster $C \in \mathcal{C}_i$ to be the number of adjacent vertices that are unclustered w.r.t. $\mathcal{C}_{i+1}$. 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 [BS03] constructs the clusterings in an even simpler manner. Rather than carefully select clusters from $\mathcal{C}_i$ for inclusion in $\mathcal{C}_{i+1}$, we 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 4. With this algorithm the *expected* size of the spanner is $O(kn^{1+1/k})$.

We next improve our $(2k-1)$-spanner construction to obtain a $(k, k-1)$-spanner. Until now all spanner edges connected unclustered vertices to clusters. We will now add edges connecting 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 $i \geq k/2$ 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 $i \geq k/2$. Our entire $(k, k-1)$-spanner construction is given in Figure 5. It consists of just those edges included by Rules R1–R4.

(R1–2)	Compute a $(2k-1)$-spanner S (det. or randomized 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 1 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 5: 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

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

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. To simplify the exposition we assume that k is odd, and let $t = (k-1)/2$. The case of even k is similar and will be treated in the full version of this paper. 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 $\mathcal{C}_i(v)$.

Let us first indicate our overall proof strategy. In analyzing the stretch of a shortest path $\langle u_0, u_1, \ldots, u_q\rangle$ we imagine a $(k+1) \times (q+1)$ matrix where the columns correspond to vertices and the rows to clusterings. The (i, j)th matrix entry is marked if u_j is i-clustered. Clearly $\mathcal{C}_k$'s row is totally unmarked and $\mathcal{C}_0$'s row is totally marked. However the rest of the array can be arbitrary. A particularly easy case is when all of the u_i's are t-clustered. We obtain a path from u_0 to u_q via $c_t(u_0), c_t(u_1), \ldots, c_t(u_q)$, which, in the array representation, is represented as a straight line through $\mathcal{C}_t$'s row. In general we have to use clusters with radius larger than t, though to establish a multiplicative stretch of k we cannot do this too often. We show, using an inductive argument, that there always exists a spanner path of the proper length that, in the array representation, is composed of a sequence of zig-zags like the one depicted in Figure 7. We achieve an overall multiplicative stretch of k by perfectly balancing *detours* and *shortcuts* corresponding to the diagonals above and below $\mathcal{C}_t$'s row.

The proof makes extensive use of the following notation.

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

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

It follows from the definitions that $\delta_S(v, f_i(v)) \leq r_i(v)$. Let us omit the subscript S and refer to δ_S as δ. We will prove the following theorem by induction for any path $\langle u_0, ..., u_q\rangle$ in G.

THEOREM 3.3. $\delta(u_0, f_t(u_i)) \leq ki + r_t(u_i)$

Observe that Theorem 3.3 immediately implies that S is a $(k, k-1)$-spanner. If u_q is t-clustered then there is a path of length $kq+t$ from u_0 to $c_t(u_q)$. Together with the path from $c_t(u_q)$ to u_q of length at most t, we have a path of length $kq + 2t = kq + k - 1$ from u_0 to u_q in S. If u_q is t-*un*clustered then $f_t(u_q) = u_q$ and $r_t(u_q) = 0$, implying a path of length kq from u_0 to u_q.

We now prove Theorem 3.3. The statement is true for $i = 0$ since $\delta(u_0, f_t(u_0)) \leq r_t(u_0)$. For the induction step, assume $i > 0$ and $\delta(u_0, f_t(u_s)) \leq ks + r_t(u_s)$ for all $s < i$. We distinguish cases according to which of u_{i-1} and u_i are t-clustered. If u_{i-1} is t-unclustered then we have $f_t(u_{i-1}) = u_{i-1}$ and $r_t(u_{i-1}) = 0$, which means that $\delta(u_0, u_{i-1}) \leq k(i-1)$. Since u_{i-1} is t-unclustered it also follows that $\delta(u_{i-1}, u_i) \leq 2t - 1$ — see the proof of Theorem 3.1. Hence,

$$\begin{aligned} &\delta(u_0, f_t(u_i)) \\ &\quad\leq\ \delta(u_0, u_{i-1}) + \delta(u_{i-1}, u_i) + \delta(u_i, f_t(u_i)) \\ &\quad\leq\ k(i-1) + (2t-1) + r_t(u_i) \\ &\quad\leq\ ki + r_t(u_i) \end{aligned}$$

Similarly, if both u_{i-1} and u_i are t-clustered then, by Rule R3, S contains an edge between $\mathcal{C}_t(u_{i-1})$ and $\mathcal{C}_t(u_i)$ since $t = k-1-t$. So there is a path of length $2t+1 = k$ between $c_t(u_{i-1})$ and $c_t(u_i)$ in S. Again we have $\delta(u_0, c_t(u_i)) \le ki + t$ using the induction hypothesis that $\delta(u_0, c_t(u_{i-1})) \le k(i-1) + t$.

We now come to the final and most interesting case: u_{i-1} is t-clustered and u_i is t-unclustered. Consider the $(k+1) \times (q+1)$ table M where rows represent clusterings and columns represent the vertices $u_0, \ldots, u_q$. The entries of M are 0 or 1 where $M[\ell, j] = 0$ means that vertex u_j is ℓ-unclustered and $M[\ell, j] = 1$ means that vertex u_j is ℓ-clustered, where $0 \le \ell \le k$ and $0 \le j \le q$. Note that row 0 of M consists of only 1's since each vertex is a singleton cluster in $\mathcal{C}_0$. We are considering the case that $M[t, i-1] = 1$ and $M[t, i] = 0$.

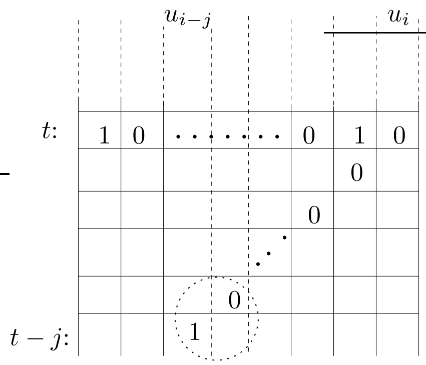

Figure 6: The Table M. The circled $0-1$ pattern is very useful to us.

We want to claim the existence of a vertex u_{i-j} such that $M[t-j, i-j] = 1$ while $M[t, i], M[t-1, i-1], \ldots, M[t-j+1, i-j+1]$ are all 0. While following the diagonal sequence of 0's in the table M starting from the location $M[i, j]$, if we do not reach the starting vertex u_0, then we have to meet such a u_{i-j} since the bottom row of the table M consists of all 1's. If we reach the zeroth column before finding a 1 we have proved our induction step for i because every vertex between u_0 and u_i was ℓ-unclustered for some $\ell \le t$. So without loss of generality, let us assume that such a u_{i-j} exists. Then the following lemma holds.

Lemma 3.1. *$\delta(c_{t-j}(u_{i-j}), u_i) \le kj - (t-j) - \sigma(j)$, where $\sigma(j) = \sum_{l=1}^{j} 2l$.*

(Proofs of Lemmas 3.1, 3.2, 3.3 are omitted.) Note that the above lemma says that we have a path from u_i to $c_{t-j}(u_{i-j})$ that is *shorter*, by $t - j + \sigma(j)$, than we were already prepared to pay for. We will use these savings to pay for a path from u_0 to $c_{t-j}(u_{i-j})$ that is longer than we could ordinarily afford. A remarkable feature of our analysis is that we amortize over paths of length $\Omega(k^2)$ yet the spanner construction itself never makes decisions within this wide a horizon.

We now need to exhibit a path in S from $c_{t-j}(u_{i-j})$ to u_0 whose length is no more than $k(i-j) + (t-j) + \sigma(j)$. We will first show a path in S from $c_{t-j}(u_{i-j})$ to $f_{t+j}(u_{i-j-1})$, using Rules R1–3, then show another path from $f_{t+j}(u_{i-j-1})$ to $f_t(u_{i-2j-1})$, which uses Rules R1–2 and R4 (see Figure 7). Finally, we invoke the induction hypothesis for $i - 2j - 1$.

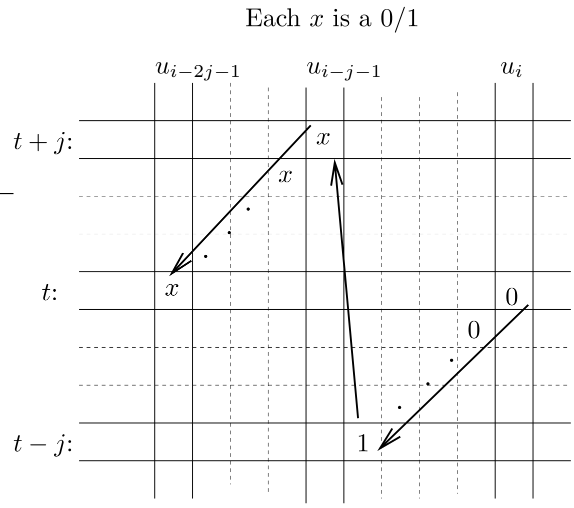

Figure 7: The sub-paths that make up our path from u_i to u_0 in S.

Lemma 3.2. *Let $(u, v) \in E$ be an edge and suppose that v is $(t-j)$-clustered. Then*

$$\delta(f_{t+j}(u), c_{t-j}(v)) \;\le\; k + 2j + (t-j) - r_{t+j}(u)$$

Lemma 3.3. *Let $\langle u_s, \ldots, u_{s+j} \rangle$ be a path in the graph G. Then*

$$\begin{aligned} &\delta(f_t(u_s), f_{t+j}(u_{s+j})) \\ &\quad \le \; kj + r_{t+j}(u_{s+j}) - r_t(u_s) + \sigma(j-1) \end{aligned}$$

Using Lemmas 3.2 and 3.3 it is now easy to complete our proof. We have the desired vertex u_{i-j}. Lemma 3.1 provides us with a path of length at most $kj - (t-j) - \sigma(j)$ between u_i and $c_{t-j}(u_{i-j})$ in S, Lemma 3.2 provides us with a path of length at most $k + 2j + (t-j) - r_{t+j}(u)$ between $c_{t-j}(u_{i-j})$ and $f_{t+j}(u_{i-j-1})$ in S, and Lemma 3.3 with $s = i - 2j - 1$ provides us with a path of length at most $kj + r_{t+j}(u_{i-j-1}) - r_t(u_s) + \sigma(j-1)$ from $f_t(u_s)$ to

$f_{t+j}(u_{i-j-1})$. In summary:

$$
\begin{aligned}
&(3.2) \quad \delta(c_{t-j}(u_{i-j}), u_i) \le kj - (t-j) - \sigma(j) \\
&(3.3) \quad \delta(f_{t+j}(u_{i-j-1}), c_{t-j}(u_{i-j})) \\
&\qquad \le k + 2j + (t-j) - r_{t+j}(u_{i-j-1}) \\
&(3.4) \quad \delta(f_t(u_{i-2j-1}), f_{t+j}(u_{i-j-1})) \\
&\qquad \le kj + r_{t+j}(u_{i-j-1}) - r_t(u_{i-2j-1}) + \sigma(j-1)
\end{aligned}
$$

So, adding Inequalities 3.2–3.4 and using $\sigma(j-1) + 2j = \sigma(j)$, we get that there is a path of length at most $k(2j+1) - r_t(s)$ from $f_t(u_s)$ to u_i where $s = i-2j-1$. We know from the induction hypothesis that there is a path of length at most $ks + r_t(s)$ from u_0 to $f_t(u_s)$. So, we have a path of length at most ki from u_0 to u_i, which proves our induction step because we are in the case where $f_t(u_i) = u_i$ and $r_t(u_i) = 0$.

Note that we have made the assumption that $i - 2j - 1 \ge 0$, which is always true when $i \ge k$. However, we did not explicitly consider the possibility that $u_{i-j} = u_0$, or that $u_{i-j-1} = u_0$, or that before we came to u_{i-2j-1} we met the starting vertex. It is easy to check that these three cases can be handled as corollaries to, respectively, Lemmas 3.1, 3.2, and 3.3.

3.1 Implementation in a Distributed Network

Rules R3 and R4 of our algorithm (Figure 5) can be executed in $O(k)$ rounds of communication in a distributed network, and, using the randomized algorithm from Figure 4, Rules R1 and R2 can also be executed in $O(k)$ rounds. See the appendix for a detailed proof of Theorem 3.4.

THEOREM 3.4. *In a synchronized distributed network G, a $(k, k-1)$-spanner of G with expected size $O(kn^{1+1/k})$ can be constructed in $O(k)$ rounds of communication. The total message volume is $O(k^2 m)$ and the maximum message length is $O(n^{1/2+1/2k})$.*

4 Further Research

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

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 purely additive or $(1+\epsilon, \beta)$-spanners admit such efficient constructions.

Lastly we would like to highlight an issue relating to Erdős's *girth conjecture* [Erd63]. We are able to show, by applying the algorithm of Figure 4 with altered sampling probabilities, that there exists a $o(n^{1+1/k})$-sized $(2k-1)$-spanner for any graph with *minimum degree* $\tilde{\Omega}(n^{1/k})$. That is, the girth bound does *not* apply to such graphs. In general, what is the optimal size α-spanner for arbitrary graphs of degree at least d?

Acknowledgment. We would like to thank Uri Zwick, Mikkel Thorup, and Michael Elkin for catching errors in an earlier manuscript.

References

[ACIM99] 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.

[ADD+93] 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.

[BCE03] 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.

[BGS04] S. Baswana, V. Goyal, and S. Sen. All-pairs nearly 2-approximate shortest paths in $o(n^2 \text{polylog} n)$ time. manuscript, 2004.

[BS03] 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 *30th Ann. Intl. Colloq. on Automata, Languages and Programming (ICALP)*, 2003.

[BS04] 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*, 2004.

[Coh98] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM J. Comput.*, 28:210–236, 1998.

[Coh00] E. Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest-paths. *J. ACM*, 47:132–166, 2000.

[Cow01] L. J. Cowen. Compact routing with minimum stretch. *J. Algor.*, 28:170–183, 2001.

[CW04] L. J. Cowen and C. G. Wagner. Compact roundtrip routing in directed networks. *J. Algor.*, 50(1):79–95, 2004.

[CZ01] E. Cohen and U. Zwick. All-pairs small-stretch paths. *J. Algor.*, 38:335–353, 2001.

[DHZ00] D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

[Elk01] M. Elkin. Computing almost shortest paths. In *20th ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, pages 53–62, 2001.

[Elk04] M. Elkin. Private communication. 2004.

[EP01] M. Elkin and D. Peleg. $(1+\epsilon,\beta)$-Spanner constructions for general graphs. In *ACM Symposium on Theory of Computing (STOC)*, 2001.

[Erd63] 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.

[EZ04] M. Elkin and J. Zhang. Efficient algorithms for constructing $(1+\epsilon,\beta)$-spanners in the distributed and streaming models. In *Proc. 23rd Ann. Symp. on Principles of Distributed Computing*, pages 160–168, 2004.

[HZ96] S. Halperin and U. Zwick. Unpublished result. 1996.

[PS89] D. Peleg and A. A. Schaffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

[PU89a] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18:740–747, 1989.

[PU89b] D. Peleg and E. Upfal. A trade-off between space amd efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

[RTZ02] L. Roditty, M. Thorup, and U. Zwick. Roundtrip spanners and roundtrip routing in directed graphs. In *Proc. 13th Ann. ACM-SIAM Symp. On Discrete Algorithms*, pages 844–851, 2002.

[TZ01a] M. Thorup and U. Zwick. Approximate distance oracles. In *Proc. 33rd Ann. ACM Symp. on Theory of Computing*, pages 183–192, 2001.

[TZ01b] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 1–10, 2001.

[Wen91] 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.

Appendix: Proof of Theorem 3.4

Before proving Theorem 3.4 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 $\lceil RL/U \rceil$, U, and V, for any any $U \geq 1$. That is, Theorem 3.4 can be adapted to any synchronized network with a fixed maximum message length.

Theorem 3.4 In a synchronized distributed network G, a $(k, k-1)$-spanner of G with expected size $O(kn^{1+1/k})$ 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. (Sketch) We compute the clusters $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k = \emptyset$ using the randomized algorithm from Figure 4. 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 $C \in \mathcal{C}_{(k-1)/2-i}$.[2] 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, 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 contributes k units of message volume for each of $k/2$ values of i. The implementation and analysis of Rule R4 is very similar to R3.

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