Small Stretch Pairwise Spanners and Approximate D-Preservers*

Telikepalli Kavitha and Nithin M. Varma

Tata Institute of Fundamental Research, India. kavitha@tcs.tifr.res.in
Pennsylvania State University, USA. nithvarma@gmail.com

Abstract. Let $G = (V, E)$ be an undirected unweighted graph on n vertices. A subgraph H of G is called a purely-additive spanner of G with stretch β if for each $(u, v) \in V \times V$, the u-v distance in H is at most $\delta_G(u, v) + \beta$. We currently know sparse purely-additive spanners with $\beta = O(1)$ only for $\beta = 2, 4, 6$. When $\beta = 2$, the size of the spanner is $O(n^{3/2})$, when $\beta = 4$, the size of the spanner is $O(n^{1.4} \log^{0.2} n)$, and when $\beta = 6$, the size of the spanner is $O(n^{4/3})$.

The following is a natural relaxation of the above problem: we care for only certain distances, these are captured by the set $\mathcal{P} \subseteq V \times V$ and the problem is to construct a sparse subgraph H (also called a $\mathcal{P}$-spanner) where for every $(u, v) \in \mathcal{P}$, the u-v distance in H is at most $\delta_G(u, v)+\beta$. In this paper we show algorithms to construct the following for $\beta = 2$:

- a $\mathcal{P}$-spanner of size $\tilde{O}(n|\mathcal{P}|^{1/3})$ for any $\mathcal{P} \subseteq V \times V$
- a $\mathcal{P}$-spanner of size $\tilde{O}(n|\mathcal{P}|^{1/4})$ when $\mathcal{P} = S \times V$ where $S \subseteq V$.

Our $(S \times V)$-spanner with additive stretch 2 leads to a simple deterministic construction of a purely-additive spanner with stretch 4 and size $O(n^{1.4} \log^{0.2} n)$. We also consider a variant of the $\mathcal{P}$-spanner problem where the set $\mathcal{P}$ is implicitly given via a distance threshold D. That is, $\mathcal{P} = \{(u, v) : \delta_G(u, v) \geq D\}$. We refer to such a $\mathcal{P}$-spanner as an *approximate D-preserver*.

A D-preserver is a subgraph where distances $\geq D$ are *exactly* preserved and D-preservers of size $O(n^2/D)$ are known. For a given $D \in \mathbb{Z}^+$ and any integer $k \geq 1$, we construct an $\tilde{O}(n^{3/2}/D^{k/(2k+2)})$-sized subgraph where distances $\geq D$ are approximated with an additive stretch of $4k$. In particular, when $k = \lfloor \log D \rfloor$, this subgraph has size $\tilde{O}(n \cdot \sqrt{n/D})$.

* A preliminary version of this paper appeared in ICALP 2013 [22]. This work was done when N. M. Varma was at TIFR.

1 Introduction

Let $G = (V, E)$ be an undirected unweighted graph on n vertices. A subgraph H of G is called a spanner of G with multiplicative stretch α and additive stretch β if we have

$$\delta_H(u, v) \ \leq \ \alpha \cdot \delta_G(u, v) \ + \ \beta \ \text{ for all pairs } (u, v) \in V \times V,$$

where $\delta_G(u, v)$ is the u-v distance in G and $\delta_H(u, v)$ is the u-v distance in H. The main goal in graph spanner problems is to compute the sparsest possible subgraph for given values of stretch.

Graph spanners were introduced by Peleg and Schaffer [27] in 1989. The problem of constructing sparse spanners with small stretch has been widely studied in weighted and unweighted graphs during the last 25 years [5, 6, 9, 16–18, 20, 23, 27, 29, 31, 32, 34, 36] and this has become a fundamental problem in graph algorithms. There are several applications that involve spanners: these include algorithms for approximate shortest paths [2, 11, 17], approximate distance oracles [33, 7, 4], labeling schemes [26, 19], network design [28], and routing [3, 13, 14].

Regarding the sparsity of spanners, it is known that every graph on n vertices admits a spanner of size $O(n^{1+1/k})$ and multiplicative stretch $2k-1$ [20, 6, 31, 32]. For unweighted graphs, one seeks spanners where the stretch is purely-additive, i.e., $\alpha = 1$. Aingworth et al. [1] (with subsequent work by Dor et al. [16] and Elkin and Peleg in [18]) showed the first purely-additive spanner with stretch 2 and size $O(n^{3/2})$. Baswana et al. [5] showed a purely-additive spanner with stretch 6 and size $O(n^{4/3})$. Very recently, Chechik [9] gave a randomized algorithm to construct a purely-additive spanner of expected size $O(n^{1.4}\log^{0.2} n)$ and stretch 4. Spanners with additive stretch 2, 4, and 6 are the only sparse purely-additive spanners with $O(1)$ stretch that are currently known.

The current best trade-offs between sparsity and additive stretch are from [9, 29]: it was shown in [9] that for any $\delta \in [3/17, 1/3)$, there is a subgraph of size $\tilde{O}(n^{1+\delta})$ and additive stretch $\tilde{O}(\sqrt{n^{1-3\delta}})$ and it was shown in [29] that for any $\delta \geq 0$, there is an $O(n^{1+\delta})$-sized spanner with additive stretch $\tilde{O}(n^{9/16-7\delta/8})$. Regarding lower bounds, Woodruff [35] showed that for every $k = O\left(\frac{\ln n}{\ln \ln n}\right)$, any additive spanner of G with stretch $2k-1$ has $\Omega(\frac{1}{k} n^{1+1/k})$ edges.

In this paper we study the following variant of the purely-additive spanner problem. Here we are given a set $\mathcal{P} \subseteq V \times V$ and the problem is to compute a sparse subgraph H where

$$\delta_H(u, v) \leq \delta_G(u, v) + \beta \ \text{ for all pairs } (u, v) \in \mathcal{P}.$$

For pairs not in $\mathcal{P}$, the stretch in H can be arbitrary. Such a subgraph H is called a $\mathcal{P}$-spanner/*pairwise spanner* of G with additive stretch β. Pairwise spanners are relevant in settings where only certain distances are of interest to us and we seek a sparse subgraph H where these distances are approximated within a small stretch. A natural setting where only certain distances are of interest to us is the case when there is a set $S \subseteq V$ of source vertices and $\mathcal{P} = S \times V$. Such pairwise spanners are also called *sourcewise spanners*.

Coppersmith and Elkin [12] in 2005 were the first to study sparse pairwise spanners. Given a graph $G = (V, E)$ along with $\mathcal{P} \subseteq V \times V$, they studied the problem of constructing a sparse subgraph H such that the u-v distance for each $(u, v) \in \mathcal{P}$ was *exactly* preserved in H, i.e., $d_H(u, v) = d_G(u, v)$ for all pairs $(u, v) \in \mathcal{P}$. They called such subgraphs *pairwise preservers* or $\mathcal{P}$-preservers. They gave algorithms to construct $\mathcal{P}$-preservers with $O(\min\{n\sqrt{|\mathcal{P}|},\ n + \sqrt{n}|\mathcal{P}|\})$ edges for any graph on n vertices. They also posed the problem

of computing sparser subgraphs where distances in $\mathcal{P}$ are *approximately* preserved, namely the problem of computing sparse $\mathcal{P}$-spanners.

This question was answered by Cygan et al. [15] who gave algorithms to construct sparse pairwise spanners with additive stretch $4k$ and sparse sourcewise spanners with additive stretch $2k$, for all integers $k \geq 1$. For any $\mathcal{P} \subseteq V \times V$ and $k \in \mathbb{Z}^+$, they showed a $\mathcal{P}$-spanner with stretch $4k$ and size $O(n^{1+1/t}(k|\mathcal{P}|)^{k/2t})$ and for any $S \subseteq V$, they showed an $(S \times V)$-spanner with stretch $2k$ and size $O(n^{1+1/t}(k|S|)^{k/t})$, where $t = 2k+1$. Also known is an $(S \times S)$-spanner of size $O(n\sqrt{|S|})$ and additive stretch 2 [29, 15].

The first problem that we consider here is that of constructing a sparse $\mathcal{P}$-spanner with additive stretch 2. A prime motivation in studying the problem of $\mathcal{P}$-spanners is to find subgraphs that are strictly sparser than all-pairs spanners for the same value of stretch. Note that the $O(n^{3/2})$ size bound for the all-pairs additive stretch 2 spanner is tight [30, 35]. We show the following result here.

Theorem 1. *Given $G = (V, E)$ and $\mathcal{P} \subseteq V \times V$, a $\mathcal{P}$-spanner of size $O(n \cdot (|\mathcal{P}| \log n)^{1/3})$ and additive stretch 2 can be constructed in polynomial time, where $|V| = n$.*

There is a large range of values of $|\mathcal{P}|$ such that the $\mathcal{P}$-spanner from Theorem 1 is sparser than the all-pairs additive stretch 2 spanner and the $\mathcal{P}$-preserver. Ignoring log-factors, the size of our $\mathcal{P}$-spanner is $o(n^{3/2})$ when $|\mathcal{P}|$ is $o(n^{3/2})$ and it is $o(\min\{n\sqrt{|\mathcal{P}|},\ n + \sqrt{n}|\mathcal{P}|\})$ when $|\mathcal{P}|$ is $\omega(n^{3/4})$.

For an appropriate setting of the set $\mathcal{P}$ in our algorithm to construct the above $\mathcal{P}$-spanner, we get Theorem 2. Note that the size of an $(S \times S)$-spanner with additive stretch 2 is $O(n\sqrt{|S|}) = O(n|\mathcal{P}|^{1/4})$ and Theorem 2 shows a bound of $\tilde{O}(n|\mathcal{P}|^{1/4})$ for an $(S \times V)$-spanner with additive stretch 2.

Theorem 2. *Given $G = (V, E)$ and $S \subseteq V$, an $(S \times V)$-spanner of size $O(n \cdot (n|S| \log n)^{1/4})$ and additive stretch 2 can be constructed in polynomial time, where $|V| = n$.*

Ignoring log-factors, the sourcewise spanner of Theorem 2 has size $o(n^{3/2})$ when $|S|$ is $o(n)$. Also, for $\mathcal{P} = S \times V$, the bound of $\tilde{O}(n|\mathcal{P}|^{1/4})$ is always $o(\min\{n\sqrt{|\mathcal{P}|},\ n + \sqrt{n}|\mathcal{P}|\})$.

For another appropriate setting of the set $\mathcal{P}$ in our algorithm to construct the $\mathcal{P}$-spanner in Theorem 1, we get a simple *deterministic* construction of an all-pairs additive spanner with stretch 4 and size $O(n^{1.4}\log^{0.2} n)$. Thus our construction improves upon the construction in [9] which shows the same bound on the expected number of edges in the all-pairs additive stretch 4 spanner.

Very recently, Parter [25] generalized the result in Theorem 2 to show for any $k \geq 1$, sourcewise spanners with additive stretch $2k$ and size $\tilde{O}(n^{1+1/r}|S|^{k/r})$, where $r = 2k+2$. Parter [25] also showed a lower bound of $\Omega(n|S|^{1/k}/k)$ on the size of a sourcewise spanner with additive stretch $2(k-1)$, for any integer $k \geq 1$. The following $\mathcal{P}$-spanners were very recently shown in [21]: for additive stretch 4, a $\mathcal{P}$-spanner of size $\tilde{O}(n \cdot |\mathcal{P}|^{2/7})$ and a sourcewise spanner of size $\tilde{O}(n \cdot (n|S|)^{2/9})$; for additive stretch 6, a $\mathcal{P}$-spanner of size $O(n \cdot |\mathcal{P}|^{1/4})$ and a sourcewise spanner of size $O(n \cdot (n|S|)^{1/5})$.

Approximate D-preservers. We next study a variant of the $\mathcal{P}$-spanner problem where the set $\mathcal{P}$ is not specified explicitly as a part of the input. Instead, the set $\mathcal{P}$ is described implicitly via a *distance threshold* D. Here along with the input graph G, we are given $D \in \mathbb{Z}^+$ and the set $\mathcal{P}$ is $\{(u, v) : \delta_G(u, v) \geq D\}$.

The problem of computing sparse D-*preservers*, where the problem is to compute a sparse $\mathcal{P}$-preserver for the set $\mathcal{P} = \{(u, v) : \delta_G(u, v) \geq D\}$, was studied by Bollobás et al. [8]. The

motivation for studying D-preservers was the following result of Elkin and Peleg [18]: for any $\epsilon > 0$ and $k \in \mathbb{Z}^+$, there exists a value $f(\epsilon, k)$ such that there is a subgraph H of G with $O(n^{1+1/k})$ edges where $\delta_H(u,v) \leq (1+\epsilon) \cdot \delta_G(u,v)$ for all (u,v) with $\delta_G(u,v) \geq f(\epsilon,k)$. This motivated the question studied in [8] of whether there is a sparse subgraph where all large distances can be exactly preserved and they showed such a D-preserver of size $O(n^2/D)$.

Here we consider the question of whether there is a subgraph sparser than the D-preserver where distances $\geq D$ can be approximated within a small additive stretch. We will use the term *approximate D-preserver* to refer to a $\mathcal{P}$-spanner for $\mathcal{P} = \{(u,v) : \delta_G(u,v) \geq D\}$. We show the following result here.

Theorem 3. *Given $G = (V,E)$ and $D \in \mathbb{Z}^+$, for any integer $k \geq 1$, an approximate D-preserver of size $O(\sqrt{k} n^{3/2} \cdot (\log n/D^k)^{1/(2k+2)})$ and additive stretch $4k$ can be computed in polynomial time.*

This result shows a trade-off between stretch and sparsity in an approximate D-preserver. In particular, when $k = \lfloor \log D \rfloor$ (for $D \geq 2$), we obtain an approximate D-preserver of size $\tilde{O}(n\sqrt{n/D})$ and additive stretch $4\lfloor \log D \rfloor$. For instance, by setting $k = \lfloor \log \sqrt{n} \rfloor$, we get a subgraph of size $\tilde{O}(n^{5/4})$ where all distances that are at least $\sqrt{n}$ are approximated with an additive stretch of $2\lfloor \log n \rfloor$. Note that an additive stretch of $4 \log D$ is exponentially small in the distance (which is $\geq D$) that we seek to approximate here.

1.1 Our Techniques

Our algorithms have two phases: the first phase partitions the vertex set into *clusters*. Since the input graph may be assumed to be sufficiently dense (otherwise there is no need to sparsify it), there are several vertices with degree at least some threshold h. The neighbors of such high-degree vertices are natural candidates for forming diameter 2 subgraphs or *clusters* as we call them. We can afford to include in our post-clustering subgraph all the edges incident on vertices that are *unclustered*, since the number of unclustered neighbors of any vertex will be less than h.

Clustering along with BFS (breadth-first-search) trees rooted at all *cluster centers* yields an all-pairs additive stretch 2 spanner of size $O(n^{3/2})$. Since we seek sparser subgraphs, we cannot afford to include BFS trees rooted at all cluster centers. We mark every pairwise shortest path ρ as either *cheap* or *expensive* depending on the number of edges in ρ that are missing in the post-clustering subgraph.

We select a few clusters so that the selected clusters intersect every expensive shortest path and BFS trees rooted at the centers of the selected clusters get included in our spanner. We show that this will approximate all expensive shortest paths with an additive stretch of 2. If ρ is a cheap shortest path, then most edges of ρ are already present in the post-clustering subgraph. The second phase of our algorithms, called "path-buying", has to decide which of these cheap shortest paths should get included in our current subgraph.

In our $\mathcal{P}$-spanner with additive stretch 2, if the u-v shortest path is cheap where $(u,v) \in \mathcal{P}$, then we buy the u-v shortest path. When $\mathcal{P} = S \times V$, if there is a cheap shortest path between $s \in S$ and vertex v where v is in cluster C, then we buy only one cheap shortest path between s and C. In the spanner with additive stretch 4, if there is a cheap shortest path between any pair of clusters, then we buy only one cheap shortest path between this pair of clusters.

In our approximate D-preserver algorithm, we buy only "affordable" cheap paths where a path is affordable if it improves the distance between many (vertex, cluster) pairs. If the u-v shortest path (where $\delta_G(u,v) \geq D$) is not affordable, then we show that there has to be

another u-v path that is not much longer than $\delta_G(u,v)$ and this path is more affordable. For a given parameter k, we find an affordable u-v path over k trials and this allows us to bound the additive stretch here by $4k$. Such a method was earlier used in the algorithms in [15] to show trade-offs between size and sparsity for pairwise spanners and sourcewise spanners.

The path-buying technique was first used in the additive stretch 6 spanner algorithm [5]. The algorithms in [5, 15] use clustering along with path-buying. The algorithm in [9] to construct an all-pairs additive stretch 4 spanner first uses randomization to pick a small sample of vertices for rooting BFS trees and then to cover the neighborhood of high-degree vertices with high probability; finally some selected shortest paths are added to get a stretch 4 spanner.

Organization of the paper. Section 2 describes the clustering step and the selection of some cluster centers to root BFS trees. Section 3 describes our algorithms to construct sparse $\mathcal{P}$-spanners (for arbitrary $\mathcal{P} \subseteq V \times V$ and for $\mathcal{P} = S \times V$) with additive stretch 2. The all-pairs additive stretch 4 spanner algorithm is given in Section 3.1. Our algorithm for approximate D-preservers with additive stretch 4 is given in Section 4.1 and for the general case, our algorithm is given in Section 4.2.

2 Clustering and BFS Trees

Given a graph $G = (V, E)$, a clustering of G is a collection $\{C_1, \ldots, C_\lambda\}$ of disjoint subsets of the vertex set V. The vertices in $U = V \setminus \cup_i C_i$ will be called *unclustered*. Corresponding to each cluster C_i, there is a vertex called the *cluster center* of C_i, denoted by center(C_i), with the following property: in the graph G, center(C_i) is a common neighbor of every vertex in C_i. Thus for each i, the set $C_i \cup \{\text{center}(C_i)\}$ of vertices has radius 1 in G; so each C_i is a diameter 2 subgraph. We will use the symbol $C(v)$ to denote the cluster to which a clustered vertex v belongs.

Given a graph G and an integer h, where $1 \leq h \leq n$, the following procedure constructs a clustering $\mathcal{C}_h = \{C_1, \ldots, C_\lambda\}$ and returns $\langle \mathcal{C}_h, U \rangle$, where U is the set of unclustered vertices.

- Initially all vertices are unclustered and $\mathcal{C}_h = \emptyset$.
- While there exists a vertex v with at least h unclustered neighbors do
  - let C be the set of unclustered neighbors of v; set center$(C) = v$.
  - mark all vertices in C as clustered; $\mathcal{C}_h = \mathcal{C}_h \cup \{C\}$.
- Let U denote the set of unclustered vertices. Return $\langle \mathcal{C}_h, U \rangle$.

It is easy to see that every pair of clusters is disjoint and each cluster C is a collection of at least h vertices. So $|\mathcal{C}_h| \leq n/h$. Associated with $\mathcal{C}_h$ is a post-clustering subgraph G_h that consists of all the edges incident on unclustered vertices as well as the edges (v, c), where v is a clustered vertex and $c = \text{center}(C(v))$.

Lemma 1. *The number of edges in G_h is $O(nh)$.*

Proof. The termination condition of the clustering procedure is the existence of no vertex with h or more unclustered neighbors. Hence when this procedure terminates, every vertex has less than h unclustered neighbors. Thus the total number of edges with at least one unclustered endpoint is less than nh.

The total number of edges (v, c), where v is a clustered vertex and c is the center of cluster $C(v)$, is at most n since each cluster has a unique center – thus there are at most n such pairs (v, c). □

The following quantity associated with a path will be useful to us in our algorithms.

Definition 1. *For any path ρ in G, let $\mathsf{cost}(\rho)$ denote the number of edges of ρ that are not present in the post-clustering subgraph G_h.*

The following lemma gives a lower bound on the number of distinct clusters that are incident on a shortest path ρ whose cost is t or more. We say a cluster C *intersects* a shortest path ρ if C and ρ have at least one vertex in common.

Lemma 2. *Let ρ be a shortest path in G with $\mathsf{cost}(\rho) \geq t$. Then there are at least $t/3$ distinct clusters of $\mathcal{C}_h$ that intersect ρ.*

Proof. Since all the edges incident on unclustered vertices are present in G_h, the number of clustered vertices in any path is at least as large as its cost. Since the diameter of a cluster is at most 2, a cluster can have at most three vertices incident on a shortest path. Since $\mathsf{cost}(\rho) \geq t$, the number of distinct clusters that intersect ρ is at least $t/3$. □

For each pair of vertices u, v in G, we will fix an arbitrary u-v shortest path in G as *the* u-v shortest path. Let $\mathcal{R} = \{\rho_1, \ldots, \rho_{\binom{n}{2}}\}$ be the set of all $\binom{n}{2}$ pairwise shortest paths in G.

Definition 2. *A path $\rho \in \mathcal{R}$ is said to be expensive if $\mathsf{cost}(\rho) \geq (n \ln n)/h^2$. A path in $\mathcal{R}$ is cheap if it is not expensive.*

The following subroutine computes a subset S_h of $\mathcal{C}_h$ such that every expensive shortest path intersects at least one cluster in S_h. Let $Q \subseteq \mathcal{R}$ be the set of expensive paths in $\mathcal{R}$. Initially all paths in Q are *uncovered* and when a cluster that interests path $p \in Q$ gets added to S_h, then p is *covered*. At least one uncovered path in Q will get covered in each iteration, so this subroutine runs for at most $|Q|$ iterations. Lemma 3 shows that the number of clusters in S_h is $O(h)$.

- Initially all expensive paths are uncovered; let $S_h = \emptyset$.
- While there exists an uncovered expensive path do
  - let $C \in \mathcal{C}_h$ intersect the largest number of uncovered expensive paths
  - $S_h = S_h \cup \{C\}$
  - mark all uncovered expensive paths intersecting C as covered.

Note that the above algorithm is the greedy set cover algorithm, where our universe is Q and corresponding to each cluster C, there is a set $S(C)$ is our collection – the set $S(C)$ is the set of those paths in Q that C intersects. It follows from the dual-fitting analysis of the greedy set cover algorithm [24, 10] that $|S_h|$ is at most $H_q \cdot \mathsf{OPT}_f$, where OPT_f is the value of the optimal fractional set cover and H_q is the q-th Harmonic number, $q = |Q| \leq \binom{n}{2}$.

Lemma 3. *The set S_h has size $O(h)$.*

Proof. Recall that every expensive path has cost at least t, where $t = (n \ln n)/h^2$. Therefore each such path has at least $t/3$ clusters intersecting it (by Lemma 2). Thus the fractional solution that selects each cluster with value $3/t$ covers every expensive path, so this is a feasible fractional set cover. Hence $\mathsf{OPT}_f \leq |\mathcal{C}_h| \cdot 3/t$, which is at most $(n/h)(3/t)$. Thus $|S_h|$ is $O(\ln n \cdot n/(ht))$; by substituting the value of $t = (n \ln n)/h^2$, we get $|S_h|$ is $O(h)$. □

For any vertex u, let $T(u)$ denote the edge set of the BFS tree in G rooted at u. Lemma 4 shows the role of S_h in approximating all expensive paths with an additive stretch of 2.

Lemma 4. *Let $\rho \in \mathcal{R}$ be the u-v shortest path in G. If ρ is expensive, then $\bigcup_{C \in S_h} T(\text{center}(C))$ has a u-v path of length at most $\delta_G(u,v)+2$.*

Proof. The shortest path ρ is expensive. Hence S_h contains some cluster C that intersects ρ. Let a be a vertex common to C and ρ; let r denote center(C). As a and r are neighbors in G, we have $\delta_G(r,u) \le \delta_G(a,u)+1$ and $\delta_G(r,v) \le \delta_G(a,v)+1$ (see Fig. 1).

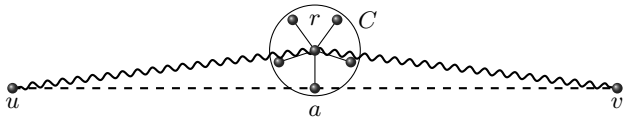

Fig. 1: *The r-u and r-v shortest paths in $T(r)$ (wavy)*

The BFS tree rooted at r contains an r-u path of length $\delta_G(r,u)$ and an r-v path of length $\delta_G(r,v)$. So there is a u-v path of length at most $\delta_G(r,u)+\delta_G(r,v) \le \delta_G(a,u)+\delta_G(a,v)+2$ in $T(r)$. Thus $\bigcup_{C \in S_h} T(\text{center}(C))$ has a u-v path of length at most $\delta_G(u,v)+2$. □

Summarizing, we have $\bigcup_{C \in S_h} T(\text{center}(C))$, which is an $O(nh)$-sized subset of edges, that approximates the u-v distance with an additive stretch of 2, for all those pairs (u,v) whose shortest path has at least $(n \ln n)/h^2$ edges missing in the post-clustering subgraph G_h.

3 Algorithms for sparse $\mathcal{P}$-spanners with additive stretch 2

In this section we prove Theorems 1 and 2 stated in Section 1. The input to the $\mathcal{P}$-spanner algorithm is an undirected unweighted graph $G=(V,E)$ on n vertices and a set $\mathcal{P} \subseteq V \times V$.

Algorithm $\mathcal{P}$-spanner (Input: $G=(V,E)$ and $\mathcal{P} \subseteq V \times V$)

(1.1) Call the clustering subroutine with parameter $h = \lceil (|\mathcal{P}| \ln n)^{1/3} \rceil$. Let H denote the post-clustering subgraph.

(1.2) Select $O(h)$ clusters to form the set S_h consisting of clusters that intersect all expensive shortest paths. Add $\bigcup_{C \in S_h} T(\text{center}(C))$ to the edge set of H.

(2) For each $(u,v) \in \mathcal{P}$ do:

- Let ρ be the u-v shortest path. If $\mathsf{cost}(\rho) < (n \ln n)/h^2$ then add ρ to H.

When step (2) says "add ρ to H", we mean add to the current subgraph H those edges of ρ that are not present in this H – note that the number of such edges in ρ is at most $\mathsf{cost}(\rho)$. The above algorithm computes a subgraph H of G and Lemma 5 bounds the size and additive stretch of H.

Lemma 5. *The subgraph H computed above is a $\mathcal{P}$-spanner of G with size $O(n(|\mathcal{P}| \log n)^{1/3})$ and additive stretch 2.*

Proof. Consider any $(u,v) \in \mathcal{P}$ and let $\rho \in \mathcal{R}$ be the shortest u-v path. If ρ is expensive, then step (1.2) of our $\mathcal{P}$-spanner algorithm ensures that $\delta_H(u,v) \le \delta_G(u,v)+2$ (see Lemma 4). Otherwise ρ is cheap, i.e., its cost is less than $(n \ln n)/h^2$, and ρ gets added to H in step (2), so $\delta_H(u,v) = \delta_G(u,v)$. This shows that H is a $\mathcal{P}$-spanner of G with additive stretch 2.

We now bound the size of H. We know that the size of H after steps (1.1) and (1.2) is $O(nh)$ (by Lemmas 1 and 3). Since $h = \lceil (|\mathcal{P}| \ln n)^{1/3} \rceil$, this is $O(n(|\mathcal{P}| \log n)^{1/3})$. The total number of edges added in step (2) is at most $|\mathcal{P}| \cdot (n \ln n)/h^2$. As $h = \lceil (|\mathcal{P}| \ln n)^{1/3} \rceil$, this is also $O(n(|\mathcal{P}| \log n)^{1/3})$. □

This finishes the proof of Theorem 1 stated in Section 1. We now consider the case when $\mathcal{P} = S \times V$, i.e., our input is $G = (V, E)$ and a set $S \subseteq V$. Our problem here is to compute a sparse subgraph H that approximates all s-v distances, where $s \in S$ and $v \in V$, with an additive stretch of 2. By calling our algorithm for sparse $\mathcal{P}$-spanners here, we get an $(S \times V)$-spanner of size $O(n \cdot (n|S| \log n)^{1/3})$. We seek a sparser subgraph here. Recall that Theorem 2 promises a sourcewise spanner of size $O(n \cdot (n|S| \log n)^{1/4})$.

We will now exploit the following fact about our $\mathcal{P}$-spanner: for any pair $(a, b) \in \mathcal{P}$, if the a-b shortest path is cheap, then the a-b distance is *exactly* preserved in the subgraph H. Additionally, the description of the set $\mathcal{P}$ is not required in steps (1.1) and (1.2) of our algorithm – these steps only need to know the size of $\mathcal{P}$.

We now determine the set $\mathcal{P}$ more carefully so that $|\mathcal{P}|$ is much smaller than $|S| \cdot n$. We run the clustering step for an appropriate parameter h and then determine the set $\mathcal{P}$ by initializing $\mathcal{P}$ to $\emptyset$ and then adding pairs to $\mathcal{P}$ as follows. For each $s \in S$ and $C \in \mathcal{C}_h$ we do:

(i) check if there is any vertex $u \in C$ such that the s-u shortest path is cheap
(ii) if so then let $v \in C$ be such that the s-v shortest path is cheap and there is no $x \in C$ with $\delta_G(s, x) < \delta_G(s, v)$ and the s-x shortest path is also cheap; add (s, v) to $\mathcal{P}$.

Step (ii) says that among all vertices $u \in C$ such that the s-u shortest path is cheap, v is a vertex that is closest to s (since there is no vertex closer to s in C with a cheap shortest path to s). For each $(s, C) \in S \times \mathcal{C}_h$, at most one pair of vertices (s, v) where $v \in C$, can belong to $\mathcal{P}$. Thus $|\mathcal{P}| \leq |S|n/h$. In order to claim $h \geq (|\mathcal{P}| \ln n)^{1/3}$, we will set h as follows:

$$h \geq (|S|n/h \cdot \ln n)^{1/3} \qquad \Longrightarrow \qquad h \geq (|S|n \ln n)^{1/4}.$$

We are now ready to present our $(S \times V)$-spanner with additive stretch 2.

1. Run steps (1.1) and (1.2) of our $\mathcal{P}$-spanner algorithm with $h = \lceil (|S|n \ln n)^{1/4} \rceil$.
2. Now determine the set $\mathcal{P}$ as described in steps (i) and (ii) above. Run step (2) of our $\mathcal{P}$-spanner algorithm using this set $\mathcal{P}$.

Let H denote the subgraph returned by the above algorithm. The size of H is $O(nh)$ since $(|\mathcal{P}| \ln n)^{1/3} \leq h$. Thus the size of H is $O(n \cdot (|S|n \log n)^{1/4})$. The following lemma bounds the stretch in H for pairs in $S \times V$. Thus Theorem 2 stated in Section 1 follows.

Lemma 6. *For every pair $(s, v) \in S \times V$, we have $\delta_H(s, v) \leq \delta_G(s, v) + 2$.*

Proof. Let (s, v) be any pair in $S \times V$. We first consider the case when v is clustered. Let ρ be the s-v shortest path. If ρ is expensive, then by step (1.2) of the $\mathcal{P}$-spanner algorithm, we have $\delta_H(s, v) \leq \delta_G(s, v) + 2$. So let us assume ρ is cheap. If $(s, v) \notin \mathcal{P}$, then there has to be another vertex $v' \in C(v)$ such that $\delta_G(s, v') \leq \delta_G(s, v)$ and $(s, v') \in \mathcal{P}$ (see Fig. 2). Since $(s, v') \in \mathcal{P}$, we bought the s-v' shortest path in our algorithm. Hence we have

$$\delta_H(s, v) \;\leq\; \delta_H(s, v') + 2 \;=\; \delta_G(s, v') + 2 \;\leq\; \delta_G(s, v) + 2.$$

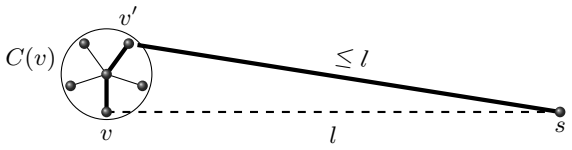

Fig. 2: *Path in H (thick) from s to v via v' of length at most $l + 2$*

We are now left with the case when v is unclustered. If every vertex on the s-v shortest path is unclustered, then this entire path is present in the post-clustering subgraph which implies $\delta_H(s, v) = \delta_G(s, v)$. So let us assume that the s-v shortest path has some clustered vertices. Let u be the clustered vertex on this path that is closest to v. So the u-v subpath of the s-v shortest path is present in H since every vertex on this subpath (other than u) is unclustered. We have already seen that $\delta_H(s, u) \leq \delta_G(s, u) + 2$. Hence we have

$$\delta_H(s, v) \;\leq\; \delta_H(s, u) + \delta_H(u, v) \;\leq\; \delta_G(s, u) + 2 + \delta_G(u, v) \;=\; \delta_G(s, v) + 2.$$

This finishes the proof of the lemma. □

3.1 A purely-additive all-pairs spanner

We now use our $\mathcal{P}$-spanner algorithm to construct a spanner with additive stretch 4 and size $O(n^{1.4} \log^{0.2} n)$. Our approach is the same as in our $(S \times V)$-spanner algorithm. We run the clustering step for an appropriate parameter h and then determine the set $\mathcal{P}$ as described below. For every pair of clusters (C_1, C_2) we do:

(i) check if there is any pair $(a, b) \in C_1 \times C_2$ such that the a-b shortest path is cheap.
(ii) if so then let $(u, v) \in C_1 \times C_2$ be such that the u-v shortest path is cheap and there is no $(x, y) \in C_1 \times C_2$ with a cheap x-y shortest path and $\delta_G(x, y) < \delta_G(u, v)$; add (u, v) to $\mathcal{P}$.

For each pair of clusters (C_1, C_2), at most one pair $(u, v) \in C_1 \times C_2$ belongs to our set $\mathcal{P}$. Thus $|\mathcal{P}| \leq (n/h)^2$. In order to claim $h \geq (|\mathcal{P}| \ln n)^{1/3}$, we will set h as follows:

$$h \geq (n^2/h^2 \cdot \ln n)^{1/3} \qquad \Longrightarrow \qquad h \geq n^{2/5} \ln^{1/5} n.$$

We are now ready to present our algorithm for the all-pairs spanner with additive stretch 4.

1. Run steps (1.1) and (1.2) of our $\mathcal{P}$-spanner algorithm with $h = \lceil n^{0.4} \ln^{0.2} n \rceil$.
2. Now determine the set $\mathcal{P}$ as described in steps (i) and (ii) above. Run step (2) of our $\mathcal{P}$-spanner algorithm using this set $\mathcal{P}$.

Let H denote the subgraph returned by the above algorithm. The size of H is $O(nh)$, which is $O(n^{1.4} \log^{0.2} n)$. The following lemma bounds the stretch in H for all pairs of vertices.

Lemma 7. *For every pair $(a, b) \in V \times V$, we have $\delta_H(a, b) \leq \delta_G(a, b) + 4$.*

Proof. Let (a, b) be any pair of vertices and let ρ be the shortest a-b path. If all vertices on this path are unclustered, then $\delta_H(a, b) = \delta_G(a, b)$ and we are done. Otherwise, let u and v be clustered vertices on ρ that are closest to a and b, respectively. We will now show that $\delta_H(u, v) \leq \delta_G(u, v) + 4$. This immediately implies that $\delta_H(a, b) \leq \delta_G(a, b) + 4$, since the a-u subpath of ρ and the v-b subpath of ρ are present in the post-clustering subgraph.

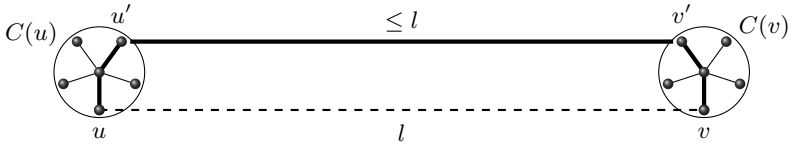

Fig. 3: *Path in H (thick) from u to v (via u' and v') of length at most $l + 4$*

Let p be the u-v shortest path. If p is expensive, then $\delta_H(u, v) \leq \delta_G(u, v)+2$ (by step (1.2) of the $\mathcal{P}$-spanner algorithm). So suppose p is cheap. This means a path $p' \in \mathcal{R}$ between some $u' \in C(u)$ and $v' \in C(v)$ was bought in step (2) of the $\mathcal{P}$-spanner algorithm and $|p'| \leq |p|$ (see Fig. 3). So we have

$$\delta_H(u, v) \;\leq\; \delta_H(u, u') + \delta_H(u', v') + \delta_H(v', v) \;\leq\; 2 + |p'| + 2 \;\leq\; |p| + 4.$$

Thus $\delta_H(u, v) \leq \delta_G(u, v) + 4$ and this finishes the proof of the lemma. □

We have thus shown the following theorem.

Theorem 4. *For any undirected unweighted $G = (V, E)$, a purely-additive spanner of size $O(n^{1.4} \log^{0.2} n)$ and stretch 4 can be constructed in polynomial time, where $|V| = n$.*

4 Approximate D-preservers

Here we are given an undirected unweighted graph $G = (V, E)$ on n vertices along with a distance threshold $D \in \mathbb{Z}^+$. Our problem is to compute a sparse $\mathcal{P}$-spanner for the set $\mathcal{P} = \{(u, v) \in V \times V : \delta_G(u, v) \geq D\}$. Recall that we refer to such a $\mathcal{P}$-spanner as an approximate D-preserver.

In this section we show Theorem 3 from Section 1. In Section 4.1 we show an approximate D-preserver with additive stretch 4 and size $O(n^{3/2} \cdot (\log n/D)^{1/4})$, which is Theorem 3 for the case $k = 1$. The case $k = 1$ is simpler and several of the ideas used in the general case are seen here. We then prove Theorem 3 for the general case in Section 4.2.

4.1 An approximate D-preserver with additive stretch 4

The algorithm here is similar to the $\mathcal{P}$-spanner algorithm from Section 3. In the first phase here, for an appropriate parameter h, we call the clustering subroutine and select $O(h)$ cluster centers and add the BFS trees rooted at these cluster centers to the post-clustering subgraph.

The second phase is *path-buying*, where for each path $\rho \in \mathcal{R}$ with $|\rho| \geq D$, we have to decide whether to buy ρ or not. To help us make this decision, along with the function $\mathsf{cost}(\rho)$ (Definition 1), we also use the function $\mathsf{value}_H(\rho)$ defined below.

Let $\delta_X(u, v)$ be the least length of a u-v path using only edges in X, where X is a subgraph or a subset of E. For a vertex u and cluster C, let $\delta_X(u, C) = \min\{\delta_X(u, v) : v \in C\}$.

Definition 3. *For any path ρ in G and subgraph H of G, let $\mathsf{value}_H(\rho)$ be the number of pairs $(u, C) \in V \times \mathcal{C}_h$ such that both u and C are incident on ρ and $\delta_\rho(u, C) < \delta_H(u, C)$.*

Thus $\mathsf{value}_H(\rho)$ is the number of (vertex, cluster) pairs that are incident on ρ and whose distance in H improves after adding ρ to H. We now present our algorithm for constructing a sparse approximate D-preserver with additive stretch 4.

1. Run steps (1.1) and (1.2) of the $\mathcal{P}$-spanner algorithm with $h = \lceil \sqrt{n} \cdot (\ln n/D)^{1/4} \rceil$. Let H be the resulting subgraph.
2. For each $\rho \in \mathcal{R}$ with $|\rho| \geq D$ do: add ρ to H if $\mathsf{cost}(\rho) \leq \mathsf{value}_H(\rho) \cdot \sqrt{\ln n/D}$.

Let H_ρ be the current subgraph when $\rho \in \mathcal{R}$ with $|\rho| \geq D$ gets considered in the path-buying step. Call ρ *affordable* if $\mathsf{cost}(\rho)$ is at most $\mathsf{value}_{H_\rho}(\rho) \cdot \sqrt{\ln n/D}$, otherwise ρ is unaffordable. We will prove in Lemma 8 that if ρ is unaffordable, then H_ρ (and thus the final subgraph H) has a path with additive stretch 4 between the endpoints of ρ.

Lemma 8. *If $(u, v) \in V \times V$ has $\delta_G(u, v) \geq D$, then $\delta_H(u, v) \leq \delta_G(u, v) + 4$.*

Proof. Consider a pair (u, v) such that $\delta_G(u, v) \geq D$. Let $\rho \in \mathcal{R}$ be the shortest u-v path. Since $h = \lceil \sqrt{n} \cdot (\ln n/D)^{1/4} \rceil$, if $\mathsf{cost}(\rho) \geq \sqrt{D \ln n}$, then ρ is an *expensive* path, i.e., its cost is at least $(n \ln n)/h^2$ and so $\delta_H(u, v) \leq \delta_G(u, v) + 2$ (by Lemma 4).

Suppose ρ is *cheap* and we did not buy ρ because it was not affordable. In other words,

$$\sqrt{D \ln n} > \mathsf{cost}(\rho) > \mathsf{value}_{H_\rho}(\rho) \cdot \sqrt{\frac{\ln n}{D}}, \tag{1}$$

where the first inequality follows from the fact that ρ is cheap. It follows from (1) that $\mathsf{value}_{H_\rho}(\rho) < D$.

Since $\mathsf{cost}(\rho)$ is positive, there are clustered vertices on ρ. Let x and y be the first and last clustered vertices on ρ, respectively. Let C_1 and C_2 be the clusters to which x and y respectively belong (see Fig. 4). Note that the u-x subpath of ρ and the y-v subpath of ρ are present in the post-clustering subgraph.

We claim there has to be a vertex r on ρ such that $\delta_{H_\rho}(r, C_1) \leq \delta_\rho(r, C_1)$ and $\delta_{H_\rho}(r, C_2) \leq \delta_\rho(r, C_2)$. Otherwise for each vertex w on ρ, we would have either $\delta_{H_\rho}(w, C_1) > \delta_\rho(w, C_1)$ or $\delta_{H_\rho}(w, C_2) > \delta_\rho(w, C_2)$; then $\mathsf{value}_{H_\rho}(\rho) \geq D$ since there are at least D vertices on ρ. This contradicts $\mathsf{value}_{H_\rho}(\rho) < D$ from (1).

We now use the vertex r to show a u-v path of length at most $\delta_G(u, v) + 4$ in H_ρ. If r lies on the x-y subpath of ρ, then there is a C_1-r-C_2 path in H_ρ of length at most $\delta_\rho(r, C_1) + \delta_\rho(r, C_2) \leq \delta_\rho(x, y)$ (see Fig. 4). We obtain the desired u-v path by concatenating the u-x subpath of ρ, the shortest x-y path in H_ρ, and the y-v subpath of ρ. Since there is an x-y path in H_ρ of length at most $\delta_\rho(x, y) + 4$, the length of this u-v path is at most $|\rho| + 4$.

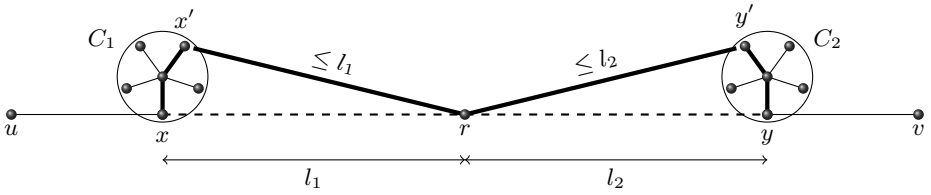

Fig. 4: *An x-y path in H_ρ (thick) via x', r and y' of length at most $l_1 + l_2 + 4$*

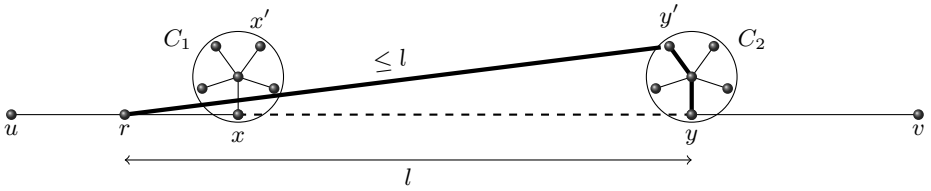

Fig. 5: *An r-y path in H_ρ (thick) via y' of length at most $l+2$*

If r is on the u-x subpath, then there is a u-v path in H_ρ obtained by concatenating the u-r subpath of ρ, the shortest r-y path in H_ρ, and the y-v subpath of ρ (see Fig. 5). Since there is an r-C_2 path in H_ρ of length $\delta_\rho(r, C_2)$, we have a u-v path in H_ρ of length at most $|\rho|+2$. The case when r is on the y-v subpath is symmetric. Thus in all cases we have shown there is a u-v path of length at most $\delta_G(u,v)+4$ in H_ρ, thus in the final subgraph H. □

We now bound the size of H in Lemma 9. Thus Theorem 3 stated in Section 1 follows for the case $k=1$.

Lemma 9. *The final subgraph H has size $O(n^{3/2} \cdot (\log n/D)^{1/4})$.*

Proof. The number of edges added in the first phase of our algorithm (steps (1.1) and (1.2) of the $\mathcal{P}$-spanner algorithm) is $O(nh)$, where $h = \lceil \sqrt{n} \cdot (\ln n/D)^{1/4} \rceil$. We now need to bound the number of edges added in the path-buying step. This is at most $\sum_\rho \mathsf{cost}(\rho)$, where the sum is over all the paths $\rho \in \mathcal{R}$ that were bought in the path-buying step in our algorithm. Since any path that gets bought satisfies our path-buying condition, we have

$$\sum_\rho \mathsf{cost}(\rho) \;\leq\; \sqrt{\frac{\ln n}{D}} \cdot \sum_\rho \mathsf{value}_{H_\rho}(\rho), \tag{2}$$

where the summations are over the paths that we bought. We bound $\sum_\rho \mathsf{value}_{H_\rho}(\rho)$ now. Let us say a (vertex, cluster) pair (v, C) *supports* path ρ if (v, C) contributes positively to $\mathsf{value}_{H_\rho}(\rho)$, i.e., v and C are incident on ρ and $\delta_\rho(v, C) < \delta_{H_\rho}(v, C)$. So for the paths ρ bought,

$$\sum_\rho \mathsf{value}_{H_\rho}(\rho) = \sum_\rho \text{number of pairs } (v,C) \text{ that supported } \rho \tag{3}$$

$$= \sum_{(v,C)} \text{number of paths bought that } (v,C) \text{ supported.} \tag{4}$$

A pair (v, C) could have supported at most 3 paths that were bought in our algorithm. This is because if $\delta_G(v, C) = d$, then for any $p \in \mathcal{R}$ such that v and C are incident on p, we have $\delta_p(v, C) \leq d+2$; otherwise the v-C subpath in p can be made shorter and thus p cannot be a shortest path between its endpoints. So the first time (v, C) supported a path that got bought, the v-C distance in the current subgraph became at most $d+2$, and after that the v-C distance could have improved at most twice. Hence the right side of (4) is at most $\sum_{(v,C)} 3 = 3|V||\mathcal{C}_h| \leq 3n^2/h$. Using (2) now, we have

$$\sum_\rho \mathsf{cost}(\rho) \leq \sqrt{\frac{\ln n}{D}} \cdot \frac{3n^2}{h} \leq 3n^{3/2} \cdot (\ln n/D)^{1/4}.$$

Thus the size of the final subgraph H is $O(n^{3/2} \cdot (\log n/D)^{1/4})$. □

4.2 An approximate D-preserver with stretch $4k$

In this section we prove Theorem 3 for $k \geq 1$. The algorithm here is similar to the algorithm in Section 4.1, however there is one difference – we allow not just paths in $\mathcal{R}$ to be bought but *approximate shortest paths* may be bought as well. The path-buying condition here uses the function $\mathsf{value}_H(.)$ (Definition 3) as well as a function $\mathsf{cost}_H(.)$ defined below.

Definition 4. *The cost of a path ρ with respect to subgraph H, denoted by $\mathsf{cost}_H(\rho)$, is the number of edges in ρ that are not present in H.*

The input here is $G = (V, E)$ along with $D, k \in \mathbb{Z}^+$. The first phase of this algorithm is similar to the first phase of our $\mathcal{P}$-spanner algorithm. For an appropriate parameter h, we call the clustering subroutine and select $O(h)$ clusters and add the BFS trees rooted at these cluster centers to the post-clustering subgraph. In the second phase of our algorithm we take care of pairs (u, v) whose distance in G is at least D and the u-v shortest path is cheap. Our algorithm is presented below and then the subroutine NextPath is described.

1. Run steps (1.1) and (1.2) of our $\mathcal{P}$-spanner algorithm with $h = \lceil \sqrt{nk} \cdot (\ln n / D^k)^{1/(2k+2)} \rceil$. Let H be the resulting subgraph.
2. For every $(u, v) \in V \times V$ such that $\delta_G(u, v) \geq D$ and the shortest u-v path is *cheap* do:
   (a) initialize p to the shortest u-v path
   (b) while $\mathsf{cost}_H(p) > 6 \cdot \mathsf{value}_H(p) \cdot (\ln n / D^k)^{1/(k+1)}$ do: $p = \textsc{NextPath}(p)$
   (c) add p to H.

Let (u, v) be a pair of vertices whose distance in G is at least D and the u-v shortest path is cheap, i.e., this path has less than $(n \ln n)/h^2 = (1/k) \cdot (D \ln n)^{k/(k+1)}$ edges missing in the post-clustering subgraph. Let ρ be the u-v shortest path and let H_ρ be the current subgraph when the pair (u, v) gets considered in step (2) of our algorithm. If ρ is *unaffordable*, i.e., $\mathsf{cost}_{H_\rho}(\rho) > 6 \cdot \mathsf{value}_{H_\rho}(\rho) \cdot (\ln n / D^k)^{1/(k+1)}$, then we come up with another u-v path via NextPath and test if this is affordable. We compute candidate u-v paths by calling NextPath repeatedly till we find an affordable one. We describe the subroutine NextPath below.

The subroutine $\textsc{NextPath}(p)$. The input to this subroutine is a u-v path p and this subroutine returns another u-v path p', where the path p' will satisfy Lemma 10.

We will maintain the invariant that any cluster C has at most 3 vertices in common with p. This invariant is true at the onset when p is the shortest u-v path ρ and we will ensure that this invariant holds for $\textsc{NextPath}(p)$ as well.

Let $\gamma = \mathsf{cost}_{H_\rho}(p)$ and $t = (D \ln n)^{1/(k+1)}$. Let L (similarly, R) denote the longest prefix (resp., suffix) of p that contains $\lfloor \gamma / 2t \rfloor$ edges that are *not* present in H_ρ. Let A (similarly, B) denote the set of clusters intersecting L (resp., R).

Claim. There are clusters $C_1 \in A$ and $C_2 \in B$ and a vertex r on p such that

$$\delta_{H_\rho}(r, C_1) \leq \delta_p(r, C_1) \quad \text{and} \quad \delta_{H_\rho}(r, C_2) \leq \delta_p(r, C_2).$$

For now we assume the above claim and show the construction of p' from p. Then we will prove the claim. The construction of p' from p is analogous to the construction in Lemma 8. Let $x \in C_1$ be the vertex that is closest to u on p and let $y \in C_2$ be the vertex that is closest to v on p. We have three cases depending on the position of r on p.

(i) If r is on the x-y subpath of p, then p' is the concatenation of the u-x subpath of p, the shortest x-y path in H_ρ, and the y-v subpath of p (see Fig. 4). Note that in this case the above claim guarantees an x-y path in H_ρ of length at most $\delta_p(x,y)+4$.

(ii) If r lies on the u-x subpath, then p' is the concatenation of the u-r subpath of p, the shortest r-y path in H_ρ, and the y-v subpath of p (see Fig. 5). Note that in this case the above claim guarantees an r-y path in H_ρ of length at most $\delta_p(r,y)+2$.

(iii) If r lies on the y-v subpath, then p' is the concatenation of the u-x subpath of p, the shortest x-r path in H_ρ, and the r-v subpath of p. Note that in this case the above claim guarantees an x-r path in H_ρ of length at most $\delta_p(x,r)+2$.

Proof of Claim. Assume without loss of generality that $|A| \le |B|$. If the above claim is not true, then for every vertex w on p there are at least $|A|$ clusters C such that $\delta_{H_\rho}(w,C) > \delta_p(w,C)$. This means that each vertex on p contributes at least $|A|$ to $\mathsf{value}_{H_\rho}(\rho)$. Thus the quantity $\mathsf{value}_{H_\rho}(p) \ge D \cdot |A|$ as there are at least D vertices on p.

Since L and R each contain $\lfloor \gamma/(2t) \rfloor$ missing edges, each of them contains at least $\lfloor \gamma/(2t) \rfloor + 1 \ge \gamma/(2t)$ clustered vertices. Hence by our invariant, $|A| \ge \gamma/(6t)$. So

$$\mathsf{value}_{H_\rho}(p) \ge D \cdot \gamma/(6t) \implies \mathsf{cost}_{H_\rho}(p) \le 6 \cdot \mathsf{value}_{H_\rho}(\rho) \cdot \frac{t}{D} = 6 \cdot \mathsf{value}_{H_\rho}(\rho) \cdot \left(\frac{\ln n}{D^k}\right)^{1/(k+1)}$$

This is a contradiction because it means p violates the condition in the while-loop of step (2b) in our algorithm, i.e., p is affordable. However recall that for the subroutine NextPath(p) to be called, it has to be the case that p was unaffordable. □

To finish the description of the subroutine NextPath(p), we need to show that the path p' maintains the invariant that no cluster has more than 3 vertices in common with it. This invariant need not hold for the path p' constructed above, however we can easily modify p' so that this invariant holds. Suppose some cluster C has more than 3 vertices in common with p'. Let $a \in C$ and $b \in C$ be the vertices on p' that are closest to u and v, respectively. By replacing the a-b subpath of p' with the a-b path of length at most 2 in the post-clustering subgraph, we shorten p' without increasing its cost. By repeating this for every cluster that has more than 3 vertices in common with the current p', we obtain a new u-v path p' that satisfies the above invariant.

Lemma 10. *We have $|p'| \le |p| + 4$ and $\mathsf{cost}_{H_\rho}(p') \le \mathsf{cost}_{H_\rho}(p)/t$, where $t = (D \ln n)^{1/(k+1)}$.*

Proof. It immediately follows from the construction of p' that $|p'| \le |p| + 4$ (from (i)-(iii) listed below the claim). Regarding $\mathsf{cost}_{H_\rho}(p')$, the only portions of p' with edges that are not present in H_ρ belong to subpaths L and R. Each of L and R, by definition, contains $\lfloor \gamma/2t \rfloor$ edges that are missing in H_ρ. Thus the total number of edges in p' not present in H_ρ is at most γ/t, which is $\mathsf{cost}_{H_\rho}(p)/t$, where $t = (D \ln n)^{1/(k+1)}$. □

The following lemma shows that for any pair (u,v) whose distance in G is at least D, the u-v distance in the final subgraph H is at most $\delta_G(u,v) + 4k$.

Lemma 11. *If $(u,v) \in V \times V$ such that $\delta_G(u,v) \ge D$, then $\delta_H(u,v) \le \delta_G(u,v) + 4k$.*

Proof. Consider a pair of vertices (u,v) such that $\delta_G(u,v) \ge D$. If ρ (the shortest u-v path) has cost at least $(1/k) \cdot (D \ln n)^{k/(k+1)}$, then this path is expensive and so $\delta_H(u,v) \le \delta_G(u,v) + 2$. So suppose ρ is cheap. The path ρ gets considered in the path-buying step.

Here we initialize p to ρ and each time p is reassigned to NextPath(p), the cost of p in the current graph H_ρ reduces by a factor of at least t (by Lemma 10). Hence after k successive invocations of NextPath, we have

$$\mathsf{cost}_{H_\rho}(p) \;<\; \frac{(D\ln n)^{k/(k+1)}}{k\cdot t^k} \;=\; \frac{(D\ln n)^{k/(k+1)}}{k(D\ln n)^{k/(k+1)}} \;=\; \frac{1}{k} \;\leq\; 1.$$

Since $\mathsf{cost}_{H_\rho}(p)$ takes only integral values, this means $\mathsf{cost}_{H_\rho}(p) = 0$. In other words, the path p has *no* missing edges in the subgraph H_ρ, i.e., p is present in the current subgraph H_ρ.

Summarizing, either we find an affordable p during the first k iterations of the while-loop or if the condition in the while-loop was satisfied for the first k iterations, then it has to get violated in the $(k+1)$-th iteration because $\mathsf{cost}_{H_\rho}(p) = 0$ for the u-v path p obtained during the k-th iteration of the while-loop. Since in each invocation of NextPath(p), the length of the path increases by at most 4 and the starting path has length $|\rho| = \delta_G(u,v)$, it follows that the final subgraph H has a u-v path of length at most $\delta_G(u,v) + 4k$. □

We now bound the size of H. We do this in Lemma 12, whose proof is similar to the proof of Lemma 9 from Section 4.1. Thus Theorem 3 stated in Section 1 follows.

Lemma 12. *The final subgraph H has size $O(\sqrt{k}n^{3/2}\cdot(\log n/D^k)^{1/(2k+2)})$.*

Proof. The number of edges added in the first phase of our algorithm (steps (1.1) and (1.2) of the $\mathcal{P}$-spanner algorithm) is $O(nh)$, where $h = \lceil\sqrt{nk}(\ln n/D^k)^{1/(2k+2)}\rceil$. We now need to bound the number of edges added in the path-buying step. This is at most $\sum_p \mathsf{cost}_{H_p}(p)$, where the sum is over all the paths that were bought in step (2c) in our algorithm and H_p was the current subgraph when p was considered in the path-buying step. Since any path that gets added to H satisfies our path-buying condition, adding up over all paths p bought in our algorithm, we get

$$\sum_p \mathsf{cost}_{H_p}(p) \;\leq\; 6\cdot\sum_p \mathsf{value}_{H_p}(p)\cdot(\ln n/D^k)^{1/(k+1)} \tag{5}$$

We bound $\sum_p \mathsf{value}_{H_p}(p)$ now. As before, we say a (vertex, cluster) pair (v,C) supports path p if (v,C) contributes positively to $\mathsf{value}_{H_p}(p)$, i.e., v and C are incident on p and $\delta_p(v,C) < \delta_{H_p}(v,C)$. So we have for the paths p bought in our algorithm,

$$\sum_p \mathsf{value}_{H_p}(p) = \sum_p \text{number of pairs } (v,C) \text{ that supported } p \tag{6}$$

$$= \sum_{(v,C)} \text{number of paths bought that } (v,C) \text{ supported.} \tag{7}$$

A pair (v,C) could have supported at most $4k+3$ paths that are bought in our algorithm. This is because every path p considered in step (2b) of our algorithm is within an additive stretch of $4k$ from a shortest path; also vertices in C are within a distance of 2 from each other. Thus it follows that the first time a path p supported by (v,C) gets bought in step (2c), we have $\delta_p(v,C) \leq \delta_G(v,C)+4k+2$. Thereafter, (v,C) can support at most $4k+2$ paths that are bought in our algorithm since the v-C distance in the current graph improves whenever a path supported by (v,C) gets bought in our algorithm. So the right side of (7) is at most $\sum_{(v,C)}(4k+3)$, which is $(4k+3)|V||\mathcal{C}_h| \leq (4k+3)n^2/h$. Using (5) now, we have

$$\sum_p \mathsf{cost}_{H_p}(p) \;\leq\; 6\cdot\frac{(4k+3)n^2}{h}\cdot\left(\frac{\ln n}{D^k}\right)^{1/(k+1)} \;\leq\; \frac{6(4k+3)n^2}{\sqrt{nk}}\cdot\left(\frac{\ln n}{D^k}\right)^{1/(2k+2)}$$

Thus the size of the final subgraph H is $O(\sqrt{k}n^{3/2}\cdot(\log n/D^k)^{1/(2k+2)})$. □

Conclusions. We considered the following variant of the additive spanner problem in an undirected unweighted graph $G = (V, E)$: given a set $\mathcal{P} \subseteq V \times V$, compute a subgraph H of G such that $\delta_H(u, v) \leq \delta_G(u, v) + \beta$ for all pairs $(u, v) \in \mathcal{P}$. Such a subgraph H is called a $\mathcal{P}$-spanner with additive stretch β.

We showed a simple polynomial time algorithm for constructing a $\mathcal{P}$-spanner of size $O(n \cdot (|\mathcal{P}| \log n)^{1/3})$ with additive stretch 2. When $\mathcal{P} = S \times V$, where $S \subseteq V$, we adapted this to a $\mathcal{P}$-spanner with additive stretch 2 and size $O(n \cdot (|\mathcal{P}| \log n)^{1/4})$. We then adapted this to an all-pairs spanner with additive stretch 4 and size $O(n \cdot (n^2 \log n)^{1/5}) = O(n^{1.4} \log^{0.2} n)$.

We also considered the case when the set $\mathcal{P}$ is implicitly specified via a distance bound D and $\mathcal{P} = \{(u, v) : \delta_G(u, v) \geq D\}$. We showed that an approximate D-preserver of size $\tilde{O}(n^{3/2}/D^{k/(2k+2)})$ and additive stretch $4k$ can be computed in polynomial time, for any integer $k \geq 1$. This implies for $D \geq 2$, an approximate D-preserver of size $\tilde{O}(n \cdot \sqrt{n/D})$ and additive stretch $4\lfloor \log D \rfloor$.

References

1. D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
2. B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1): pages 263-277, 1998.
3. B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Math.*, 5(2): pages 151-162, 1992.
4. S. Baswana and T. Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing*, 39(7): pages 2865-2896, 2010.
5. S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. Additive Spanners and (α, β)-Spanners. *ACM Transactions on Algorithms*, 7(1), 2010.
6. 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. *Random Structures and Algorithms*, 30(4): pages 532-563 , 2007.
7. S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in $O(n^2 \log n)$ time. *ACM Transactions on Algorithms*, 2(4): pages 557-577, 2006.
8. B. Bollobás, D. Coppersmith, and M. Elkin. Sparse Distance Preservers and Additive Spanners. *SIAM Journal on Discrete Math.*, 19(4): pages 1029-1055, 2005.
9. S. Chechik. New Additive Spanners. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498-512, 2013.
10. V. Chvátal. A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 4(3): 233-235, 1979.
11. E. Cohen. Fast algorithms for constructing t-spanners and paths of stretch t. In *Proc. 34th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 648-658, 1993.
12. 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.
13. L. J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 28: pages 170-183, 2001.
14. L. J. Cowen and C. G. Wagner. Compact roundtrip routing in directed networks. *Journal of Algorithms*, 50(1): pages 79-95, 2004.
15. M. Cygan, F. Grandoni, and T. Kavitha. On Pairwise Spanners. In *Proc. 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 209-220, 2013.
16. D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5): pages 1740-1759, 2004.
17. M. Elkin. Computing almost shortest paths. In *ACM Transactions on Algorithms*, 1(2): pages 283-323, 2005.
18. M. Elkin and D. Peleg. $(1+\epsilon, \beta)$-spanner construction for general graphs. *SIAM Journal on Computing*, 33(3): pages 608-631, 2004.
19. C. Gavoille, D. Peleg, S. Perennes, and R. Raz. Distance labeling in graphs. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 210-219, 2001.
20. S. Halperin and U. Zwick. Unpublished result, 1996.
21. T. Kavitha. New Pairwise Spanners. In *Proc. 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 513-526, 2015.

22. T. Kavitha and N. M. Varma. Small Stretch Pairwise Spanners. In *Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 601-612, 2013.
23. M. B. T. Knudsen. Additive Spanners: A Simple Construction. In *Proc. 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 277-281, 2014.
24. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13: pages 383-390, 1975.
25. M. Parter. Bypassing Erdős' girth conjecture: Hybrid spanners and sourcewise spanners. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 608–619, 2014.
26. D. Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3): pages 167-176, 2000.
27. D. Peleg and A. A. Schaffer. Graph Spanners. *Journal of Graph Theory*, 13: pages 99-116, 1989.
28. D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing*, 18: pages 740-747, 1989.
29. S. Pettie. Low Distortion Spanners. *ACM Transactions on Algorithms*, 6(1), 2009.
30. I. Reiman. Über ein Problem von K. Zarankiewicz. *Acta. Math. Acad. Sci. Hungar.*, 9: pages 269-273, 1958.
31. L. Roditty and U. Zwick. On dynamic shortest paths problems. In *Proc. 12th Annual European Symposium on Algorithms (ESA)*, pages 580-591, 2004.
32. L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proc. 32nd Int. Colloq. on Automata, Languages, and Programming (ICALP)*, pages 261-272, 2005.
33. M. Thorup and U. Zwick. Approximate Distance Oracles. *Journal of the ACM*, 52(1): pages 1-24, 2005.
34. 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.
35. D. P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 389-398, 2006.
36. D. P. Woodruff. Additive Spanners in Nearly Quadratic Time. In *Proc. 37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 463-474, 2010.