Small Stretch Pairwise Spanners and D-spanners

A Thesis

Submitted to the
Tata Institute of Fundamental Research, Mumbai
for the degree of Master of Sciences

in Computer Science

by

Nithin M. Varma

under the guidance of

Dr. T. Kavitha

School of Technology and Computer Science
Tata Institute of Fundamental Research
Mumbai

July 2014

Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work was done under the guidance of Professor T. Kavitha, at the Tata Institute of Fundamental Research, Mumbai.

Nithin M. Varma

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

T. Kavitha

Abstract

An (α, β)-spanner of an undirected unweighted connected graph $G = (V, E)$ is a subgraph H such that:

$$d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta,$$

for all pairs $(u, v) \in V \times V$, where $d_H(u, v)$ and $d_G(u, v)$ are the distances between u and v in H and G respectively. The quantities α and β are non negative real numbers and are called the multiplicative stretch and additive stretch of the spanner respectively. If $\alpha = 1$, the spanner is called *additive*. In this report, we focus our attention to additive spanners. Additive spanners are well studied.

We study a natural generalization of the additive spanner problem where we look to approximate the distances of only a specified set of pairs of nodes. Given a graph $G = (V, E)$ and a set $\mathcal{P} \subseteq V \times V$, an (α, β) $\mathcal{P}$-spanner, or a *pairwise spanner*, of G is a subgraph H such that $d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$ for all $(u, v) \in \mathcal{P}$. We obtain polynomial time constructions for the following pairwise spanners:

- a $(1, 2)$ $\mathcal{P}$-spanner with $\tilde{O}(n|\mathcal{P}|^{1/3})$ edges when $\mathcal{P} \subseteq V \times V$ is arbitrary,

- a $(1, 2)$ $\mathcal{P}$-spanner with $\tilde{O}(n|\mathcal{P}|^{1/4})$ edges when $\mathcal{P} = S \times V$ for some $S \subseteq V$.

In the special case when $\mathcal{P}$ contains exactly those pairs of nodes which are at a distance at least D in G, an (α, β) $\mathcal{P}$-spanner of G is also called as an (α, β) D-spanner. For any integer $k \geq 1$, we present polynomial time algorithms to construct:

- a $(1, 4k)$ D-spanner with $\tilde{O}(n^{3/2}/D^{k/(2k+2)})$ edges.

A part of this work has appeared in the proceedings of ICALP 2013 as a paper titled Small Stretch Pairwise Spanners [KV13].

Acknowledgements

This thesis would not have happened without the support and encouragement of many people. I am extremely grateful to my advisor Dr. Kavitha Telikepalli for getting me interested in the field of Graph Algorithms through her wonderful course lectures, introducing me to the exciting area of Graph Spanners and for all the help, support and encouragement she has given me throughout. I could learn a lot from her and her clarity of thought continue to inspire me.

I express my sincere gratitude to Dr. Arkadev Chattopadhyay, Dr. Prahladh Harsha and Dr. Jaikumar Radhakrishnan for serving on my thesis committee as well as helping me by pointing out the several issues which an earlier version of this report had.

I am particularly thankful to Jaikumar and Prahladh for putting faith in me, for encouraging me continuously and for everything that they have taught me over the years that I spent at TIFR.

A portion of the work described in this report appeared in the proceedings of ICALP 2013. I am indebted to Kavitha, Dr. Naveen Garg and Dr. Kurt Mehlhorn for giving me the opportunity to attend the conference by arranging a visit to Max Planck Institute for Informatics, Saarbrucken, Germany and also for funding my trip under the IMPECS grant. I extend my gratitude to Dr. Geevarghese Philip and Dr. Sayan Bhattacharya at MPII for all their help during my days there.

I owe a lot to the faculty, staff and colleagues at the School of Technology and Computer Science, TIFR for creating a friendly environment conducive for research.

Last but not least, I thank my friends and family for being with me and supporting me through all the highs and lows all these years.

Contents

Chapter 1

Introduction

A spanner of an undirected unweighted graph $G = (V, E)$ with *stretch function* $f : \mathbb{N} \to \mathbb{N}$ is a *subgraph* H of G such that for any two nodes $s, t \in V$:

$$d_H(s,t) \leq f(d_G(s,t))$$

where $d_G(s,t)$ and $d_H(s,t)$ are the distances between the nodes s and t in G and H respectively. The number of edges in H is often referred to as the *size* of the spanner. Spanners were introduced by Peleg and Schaffer [PS89]. The use of spanners arise naturally in situations where we need to store *approximate distances* in a *space efficient* manner. Their applications include space-efficient routing schemes [Cow01, CW04, AP92], synchronizers [PU89], approximate distance oracles [TZ05, BS04, RTZ05, PR10], near-shortest path algorithms [ACIM99, BK10, RZ04, Elk05], etc.

Sparse spanners with *low* stretch functions are desired in many of these. This motivates us to look for the sparsest possible spanner of a graph for a given stretch function. The principal direction of research in the area of graph spanners is to answer questions about the inherent tradeoffs between their size and stretch.

A widely studied class of spanners are those with stretch functions of the form $f(d) = \alpha d + \beta$ where α and β are non-negative real numbers. Such spanners are called as (α, β)-spanners. Formally, an (α, β)-spanner of an undirected unweighted graph G on n nodes is a *subgraph* H of G such that for any two nodes s, t:

$$d_H(s,t) \leq \alpha \cdot d_G(s,t) + \beta.$$

The quantities α and β are called the *multiplicative* stretch and *additive* stretch of the spanner H, respectively. The pair (α, β) is called the stretch of the spanner. If $\beta = 0$, the spanner is called *multiplicative* and if $\alpha = 1$, the spanner is called *additive*. If in addition to α being 1, $\beta = O(1)$, the spanner is said to be *purely additive*

Multiplicative Spanners. Multiplicative spanners are very well studied. It is known that one can compute a $(2k-1, 0)$-spanner with $O\left(n^{1+1/k}\right)$ edges for every graph on n nodes [HZ96, BS06, RZ04, RTZ05]. This bound is believed to be tight on the basis of the still unproven Girth[1] Conjecture of Erdős [Erd64]. The girth conjecture says that there exist graphs with $\Omega(n^{1+1/k})$ edges and girth $2k+1$ for any integer k. Removing any edge from such a graph would increase the distance between its endpoints to at least $2k$. This means that the only $(2k-1, 0)$-spanner of such a graph is the graph itself.

[1] Girth is the length of the smallest cycle in the graph

Additive Spanners. In this report, we restrict our attention to additive spanners. The stretch function of an additive spanner is of the form $f(d) = d + \beta$. Since additive spanners *approximate* distances better than multiplicative spanners, they are more desirable.

The stretch of multiplicative spanners can be bounded by bounding the distance between the endpoints of edges in the original graph missing from the spanner. This straightforward technique does not seem to apply for additive spanners. The first purely additive spanner for unweighted undirected graphs is due to Aingworth et al. [ACIM99]. They gave the construction of a $(1, 2)$-spanner with $\tilde{O}(n^{1.5})$ edges. This was improved to $O(n^{1.5})$ in [DHZ00, EP04]. Later Baswana et al. [BKMP05] came up with a $(1, 6)$-spanner having $O(n^{4/3})$ edges. On the lower bound side, Woodruff [Woo06] has shown that there exists a graph on n nodes for which any $(1, 2k-1)$-spanner has $\Omega\left(\frac{1}{k}n^{1+1/k}\right)$ edges. This lower bound implies that the $(1, 2)$-spanner of [EP04] is optimal. For a long time, the $(1, 2)$-spanner and the $(1, 6)$-spanner were the only purely additive spanners known. Recently Chechik [Che13] came up with a randomized algorithm which constructs a $(1, 4)$-spanner having $\tilde{O}(n^{1.4})$ expected size with very high probability. Chechik [Che13] also came up with a $(1, \tilde{O}(n^{\frac{1-3\delta}{2}}))$-spanner with $\tilde{O}(n^{1+\delta})$ edges for any $\delta \in [\frac{3}{17}, \frac{1}{3}]$. It is important to mention in this context that we do not know even the existence of any spanner which is sparser than the $(1, 6)$-spanner of [BKMP05] and whose stretch is subpolynomial in the number of nodes. We summarise the details of the constant stretch additive spanners in the following table.

Additive Stretch	**Size**	**Reference**
2	$O(n^{1.5})$	[EP04]
6	$O(n^{1.33})$	[BKMP05]
4	$\tilde{O}(n^{1.4})$	[Che13]

Table 1.1: Purely Additive Spanners

Pairwise Spanners. The apparent difficulty involved in even arguing the existence of additive spanners of size $o(n^{4/3})$ and additive stretch that is subpolynomial in the number of nodes, is motivating enough to consider the problem of whether or not it is possible to get sparser subgraphs if the requirement is to not approximate all the distances. In several situations where spanners come useful, one may not be interested in *approximating* the distances between all pairs of vertices. To facilitate the discussion of such cases, we generalize the definition of spanners to *pairwise spanners*.

Given an undirected unweighted graph $G = (V, E)$ and a set $\mathcal{P} \subseteq V \times V$, an (α, β) $\mathcal{P}$-spanner, or a *pairwise spanner*, of G is a subgraph H such that for any two nodes s, t where $(s, t) \in \mathcal{P}$:

$$d_H(s, t) \leq \alpha \cdot d_G(s, t) + \beta.$$

In the above definition, there are many possible settings to $\mathcal{P}$. For instance, $\mathcal{P}$ may be (i) an arbitrary subset of $V \times V$, (ii) $S \times V$ for some $S \subseteq V$, or (iii) $S \times S$ for some $S \subseteq V$. When $\mathcal{P} = S \times V$, the spanner is referred to as a *sourcewise spanner*.

Pairwise spanners with $\alpha = 1$ are called additive pairwise spanners and the ones having $\beta = O(1)$, in addition, are called as purely additive pairwise spanners. Additive pairwise spanners have been studied in the past. In [CE05], the authors looked at the special case

when the distances corresponding to pairs in a set $\mathcal{P} \subseteq V \times V$ have to be exactly preserved. They called such subgraphs $\mathcal{P}$-preservers. One of their main results was a construction of $\mathcal{P}$-preservers with $O(\min(n\sqrt{|\mathcal{P}|}, |\mathcal{P}|\sqrt{n}))$ edges. They left it open to study the approximate variants of their preservers, i.e. the problem of $\mathcal{P}$-spanners. Cygan et al. [CGK13] answered this problem for different settings of $\mathcal{P}$. They showed tradeoffs between the stretch and size of the spanner for some of these settings. Those results are as follows:

- a $(1, 4k)$ $\mathcal{P}$-spanner with $O(n^{1+\frac{1}{2k+1}} \cdot ((4k+5) \cdot |\mathcal{P}|)^{\frac{k}{4k+2}})$ edges for arbitrary $\mathcal{P} \subseteq V \times V$
- a $(1, 2k)$ $S \times V$-spanner with $O(n^{1+\frac{1}{2k+1}} \cdot (k \cdot |S|)^{\frac{k}{2k+1}})$ edges where $S \subseteq V$

By setting $k = \lfloor \log n \rfloor$ in the above, we can see that any graph G on n nodes has an $\tilde{O}(n \cdot |\mathcal{P}|^{1/4})$-sized $(1, 4\log n)$ $\mathcal{P}$-spanner as well as an $\tilde{O}(n \cdot \sqrt{|S|})$-sized $(1, 2\log n)$ $S \times V$-spanner and that these can be constructed efficiently.

Our Results on purely additive $\mathcal{P}$-spanners. We give constructions for an additive pairwise spanner and an additive sourcewise spanner when the stretch is small. We state these results in the following theorems.

Theorem 1.0.1. *There is a polynomial time algorithm which takes a graph $G = (V, E)$ on n nodes and a set $\mathcal{P} \subseteq V \times V$ as its inputs and computes an $O(n \cdot (|\mathcal{P}| \log n)^{1/3})$-sized $(1, 2)$ $\mathcal{P}$-spanner of G.*

Note that the above pairwise spanner, in spite of having a smaller stretch, is sparser than the $(1, 4)$ $\mathcal{P}$-spanner of Cygan et al. [CGK13] for all values of $|\mathcal{P}|$. It is sparser than the $\mathcal{P}$-preserver of [CE05] when $|\mathcal{P}|$ is $\omega(n^{3/4})$. It is also sparser than the $(1, 2)$ all-pairs spanner when $|\mathcal{P}|$ is $o(n^{3/2})$. We have to ignore some logarithmic factors for all these relations to hold.

Theorem 1.0.2. *There is a polynomial time algorithm which takes a graph $G = (V, E)$ on n nodes and a set $S \subseteq V$ as its inputs and computes an $O(n \cdot (|\mathcal{P}| \log n)^{1/4})$-sized $(1, 2)$ $\mathcal{P}$-spanner of G where $\mathcal{P} = S \times V$.*

We remark that our sourcewise spanner is always sparser than the $\mathcal{P}$-preserver of [CE05] when $\mathcal{P} = S \times V$ and the $(1, 2)$ $S \times V$-spanner of [CGK13] for any $S \subseteq V$. These relations hold only when we ignore some logarithmic factors.

Additive all-pairs spanners can be considered as special cases of additive pairwise spanners when the set of pairs $\mathcal{P}$ is $V \times V$. In this report, we also show an algorithm to construct a $(1, 4)$ spanner which is very similar to our algorithms for the above small stretch pairwise spanners. The existing $(1, 4)$-spanner algorithm due to Chechik [Che13] is randomized, and computes a $(1, 4)$-spanner of expected size $O(n^{1.4} \log^{0.2} n)$ with high probability. On the contrary, ours is a deterministic $(1, 4)$-spanner algorithm with the same worst case bound on the size of the subgraph output.

D-spanners. We next consider a variant of the $\mathcal{P}$-spanner problem when the pairs are implicitly specified via an integral distance threshold $D \in [1, n]$. The requirement here is to approximate distances between all pairs separated by a distance at least D in the original graph. Such pairwise spanners are referred to as D-spanners. An (α, β) D-spanner of a graph $G = (V, E)$ is an (α, β) $\mathcal{P}$-spanner where $\mathcal{P} = \{(u, v) \in V \times V : d_G(u, v) \geq D\}$.

The problem of D-spanners was motivated by Elkin and Peleg [EP04] who made the important observation that it is relatively easy to approximate large distances in graphs. In

their paper they show a $(1+\epsilon, 0)$ D-spanner with $O(n^{1+\lambda})$ edges for any $\epsilon > 0$, $\lambda > 0$ where D is a function of ϵ and λ.

Later, Bollobás et al. [BCE05] came up with the construction of a subgraph that exactly preserves the distances between those pairs of nodes which are at a distance at least D in the original graph. They called such subgraphs D-preservers. They showed that any graph on n nodes has a D-preserver with $O(n^2/D)$ edges. They also show graphs for which any D-preserver has $\Omega(n^2/D)$ edges for all n.

Our Results on D-spanners. In this report we address the problem of computing D-spanners whose stretch is additive (called as additive D-spanners). We show the following result.

Theorem 1.0.3. *There is a polynomial time algorithm that takes a graph $G = (V, E)$ on n nodes and a number $D \in [1, n]$ as its inputs and computes an $\tilde{O}(n \cdot \sqrt{n/D^{k/(k+1)}})$-sized $(1, 4k)$ D-spanner of G for any integer $k \geq 1$.*

This result shows a tradeoff between the stretch and sparseness of a D-spanner. In particular, when $k = \lfloor \log n \rfloor$, we obtain a D-spanner of size $\tilde{O}(n \cdot \sqrt{n/D})$ and additive stretch at most $4 \log n$. This is always sparser than the $O(n^2/D)$-sized D-preserver of [BCE05], for any value of D (ignoring logarithmic factors).

Organization. The report is organized as follows. In Chapter 2, we present the overview of our algorithms along with a description of two important techniques that we use throughout. We also describe some of the important existing spanner constructions which have crucially used all or some of these techniques. Chapter 3 contains our algorithms for the pairwise spanner and the sourcewise spanner of additive stretch 2. We also describe our determinstic $(1, 4)$-spanner algorithm in that chapter. In Chapter 4, we present our result on the size-stretch tradeoffs of additive D-spanners. We present our concluding remarks and further directions for investigation in Chapter 5.

Chapter 2

Main Algorithmic Techniques

The problems which we study can all be abstracted in the following way.

Generic $(1,t)$ $\mathcal{P}$*-Spanner Problem*: Given a graph $G = (V, E)$ and a subset $\mathcal{P} \subseteq V \times V$, compute a sparse subgraph H of G such that:

$$d_H(u,v) \leq d_G(u,v) + t \text{ for all } (u,v) \in \mathcal{P} \ .$$

Our algorithms to solve all its variants have a lot of features in common. In this chapter, we present some of the key ideas used in all our algorithms. We first present an overview of our algorithms to solve the various pairwise spanner problems that we consider.

These algorithms have three main phases. We start with an empty graph H on the same vertex set as G.

In the first phase, we partition V into disjoint subsets $C_1, C_2, \ldots, C_\lambda$ and U such that nodes in each C_i have a common neighbor. Since the input graph may be assumed to be *sufficiently dense* (otherwise the graph itself is output as the spanner), there are quite a few *high* degree nodes. Such nodes together with their neighbours form natural candidates for forming these groups (or *clusters* as we call them). We add some of the edges within a cluster to H to ensure that they form diameter 2 subgraphs in H. We will also add to H, all the edges incident on nodes which are not part of any such *cluster*, since roughly speaking, they are nodes with *low* degrees. These steps constitute the Clustering phase of our algorithms.

Once the clustering phase is over, pairs in $\mathcal{P}$ are classified based on the number of edges that are missing from H, in the shortest path between them.

For pairs in $\mathcal{P}$ whose shortest paths have *many* missing edges, we make use of the following observation. It is that the number of clusters incident a shortest path is proportional to the number of missing edges in that path. Based on this, we apply a greedy strategy to select a *few* clusters that intersect all the shortest paths with *many* missing edges. We then add all the edges in shortest paths trees rooted at a few appropriate nodes, one from each cluster in this set, to H. This phase is called the Shortest Paths Tree Addition Phase.

If the number of missing edges in the shortest path ρ between nodes u and v, where $(u,v) \in \mathcal{P}$, is *small*, we consider adding either that path or a *slightly longer* path between the same nodes to H. We do this for each such *cheap* path. These steps (with variations depending on the problem considered) constitute the Path Buying phase of our algorithms.

We return the subgraph H formed after all these phases as the output.

2.1 Clusters and Shortest Path Trees

The steps in the Clustering and Shortest Paths Tree Addition phases are the same in all our algorithms. We present their formal descriptions in this section.

We first describe the steps in the Clustering Phase. A clustering of a graph $G = (V, E)$ is a partition of its vertex set V into subsets $C_1, \ldots, C_\lambda$ called *clusters* and the set $U = V \setminus \cup_i C_i$ of *unclustered* vertices. Associated with each cluster C_i is a node called its *cluster center*, denoted by center(C_i), with the following property:

– In the graph G, center(C_i) is a common neighbor of all $x \in C_i$.

Several clusters can share the same cluster center and center(C_i) $\notin C_i$, for any i. The cluster to which a clustered node v belongs is denoted by $C(v)$.

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

- Initially all the vertices are unclustered and $\mathcal{C}_h = \emptyset$.
- While there exists a $v \in V$ with at least h unclustered neighbours:
  - Let C be the set of unclustered neighbors of v; Set center(C) $= v$.
  - All vertices in C are marked clustered; $\mathcal{C}_h = \mathcal{C}_h \cup \{C\}$.
- Set U to the set of unclustered nodes.
- Initialise G_h to the empty graph.
- Add all the edges with at least one endpoint in U to G_h.
- For each clustered node v, add the edge between v and center($C(v)$) to G_h.

Thus, each cluster in $\mathcal{C}_h$ is a collection of at least h vertices; so there can be at most n/h clusters in $\mathcal{C}_h$.

Associated with $\langle \mathcal{C}_h, U \rangle$ is the subgraph G_h (also referred to as the cluster subgraph), which has (i) all the edges incident on unclustered nodes, and (ii) all the edges between a clustered node and the center of its cluster. The spanner H being constructed is set to the subgraph G_h after the clustering phase.

We now prove some properties of the clustering $\langle \mathcal{C}_h, U \rangle$ and the cluster subgraph G_h.

Lemma 2.1.1. *The number of edges in G_h is $O(nh)$.*

Proof. The termination condition of the while loop in the clustering procedure is the absence of any node with h or more unclustered neighbors. Thus when this procedure terminates, each node has less than h unclustered neighbors. Hence the total number of edges with at least one unclustered endpoint is at most nh.

The set of edges of the form (x, c) where x is a clustered node and c its cluster center, form a forest. Thus the total number of such edges can be at most $n - 1$. This completes the proof that G_h has $O(nh)$ edges. □

Lemma 2.1.2. *The diameter of any cluster with respect to G_h (as well as G) is at most 2.*

Proof. All nodes belonging to the same cluster have a common neighbour in G_h (as well as G). Thus, the diameter of any cluster with respect to G_h (as well as G) is at most 2. □

Before we go on to describe the steps in the Shortest Paths Tree Addition Phase, we make some conventions and definitions regarding paths in the input graph G. We associate with each pair of vertices $(u, v) \in V \times V$, an arbitrary shortest u-v path in G. This path will be called as *the* shortest u-v path. Let $\mathcal{R} = \{\rho_1, \rho_2, \ldots, \rho_{\binom{n}{2}}\}$ be the set of all $\binom{n}{2}$ pairwise shortest paths in G.

Definition 2.1.3. *For any path ρ in G, $\mathsf{cost}(\rho)$ denotes the number of edges of ρ that are absent in G_h.*

Definition 2.1.4. *A path ρ is called expensive if $\mathsf{cost}(\rho) \geq (n \log n)/h^2$. A path is cheap if it is not expensive.*

A cluster C is said to intersect a shortest path ρ if C and ρ have at least one node in common. In the Shortest Paths Tree Addition phase, we select $O(h)$ clusters, so that each expensive path in $\mathcal{R}$ has some selected cluster intersecting it. We then add the edges in the shortest paths trees rooted at the centers of these selected clusters to the spanner H being constructed.

We first prove that the number of clusters intersecting a shortest path is proportional to its cost.

Lemma 2.1.5. *Let ρ be a shortest path in G with $\mathsf{cost}(\rho) \geq t$. Then there are at least $t/3$ clusters of $\mathcal{C}_h$ intersecting ρ.*

Proof. Since all the edges incident on unclustered nodes are present in G_h, the number of clustered nodes in any path is at least as large as its cost. A cluster can have at most three nodes in common with a shortest path, since the diameter of a cluster is at most 2 as proved in Lemma 2.1.2. Therefore, the total number of clusters intersecting with a shortest path of cost t or more is at least $t/3$. □

We are now ready to describe the procedure to form a set of clusters such that each expensive shortest path is intersected by some cluster from this set.

- Initially all expensive paths are uncovered and let $S_h = \emptyset$.
- While there exists an uncovered expensive path:
  - Let C be the cluster that intersects the largest number of uncovered expensive paths (ties broken arbitrarily).
  - Mark all expensive paths intersecting C as covered.
  - $S_h = S_h \cup \{C\}$

The above procedure terminates, since in each iteration, at least one uncovered path gets covered. Lemma 2.1.6 gives an upper bound on the number of clusters in S_h.

Lemma 2.1.6. *The number of clusters added to S_h by the above procedure is $O(h)$.*

Proof. We prove the statement by bounding the number of iterations of the while loop in the procedure.

Let k_i denote the number of uncovered expensive paths at the beginning of the i^{th} iteration of the while loop. Each such expensive path has cost at least t, where $t = (n \cdot \log n)/(h^2)$. Therefore, each such path has at least $t/3$ clusters intersecting it, by Lemma 2.1.5. The total number of clusters in $\mathcal{C}_h$ is at most n/h. Hence at the beginning of i^{th} iteration, there is a cluster C' which intersects at least $(k_i \cdot t/3)/(n/h)$ uncovered expensive paths. Clearly, the cluster which our procedure chooses in the i^{th} iteration also covers at least as many uncovered expensive paths as C'. Substituting the value of t, the number of uncovered expensive paths that get covered in the i^{th} iteration of the procedure is at least $(k_i \cdot \log n)/3h$. This means that in each iteration, the number of uncovered expensive paths decreases by a factor of $(1 - (\log n/3h))$.

Hence the number of uncovered expensive paths at the end of r^{th} iteration is at most $k_1(1 - (\log n/3h))^r$. Since $k_1 \leq \binom{n}{2}$, we can see that after $6h$ iterations, the number of uncovered expensive paths drops to less than 1. As the termination condition of the while loop is the non-existence of any uncovered expensive path, the number of iterations of the while loop is at most $6h$.

Thus, the number of clusters selected by the procedure for the addition of shortest paths trees is $O(h)$. □

For a cluster C, let T_C denote the shortest paths tree in G rooted at center(C). In the following Lemma, we prove that the union of edges in the shortest path trees rooted at the centers of clusters in S_h is a subgraph that approximates all the expensive shortest paths within an additive term of 2.

Lemma 2.1.7. *Let $(u, v) \in V \times V$ be such that the u-v shortest path ρ in G is expensive. Then the subgraph $\bigcup_{C \in S_h} T_C$ has a path of length at most $d_G(u, v) + 2$ between u and v.*

Proof. Since ρ is expensive, the procedure ensures that some cluster C' intersecting ρ is added to S_h by the end of it. Let r denote center(C'). Since the subgraph $\bigcup_{C \in S_h} T_C$ contains all the edges in T'_C, the shortest paths from r to both u and v are present in H. In other words $d_H(r, u) = d_G(r, u)$ and $d_H(r, v) = d_G(r, v)$.

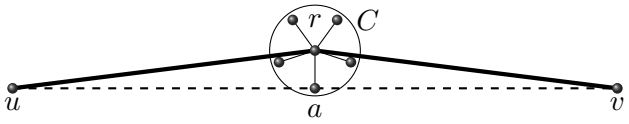

Figure 2.1: *Shortest paths in H (thick) from* center(C) *to u and v*

Let a be a node common to C' and ρ. As a and r are neighbours in G, by triangle inequality, $d_G(r, w) \leq d_G(a, w) + 1$ for $w \in \{u, v\}$. From these, we can infer that $d_H(u, v) \leq d_G(u, v) + 2$. □

Thus, we have shown that adding the edges in $\bigcup_{C \in S_h} T_C$ to the spanner being constructed, approximates the u-v distance of all pairs (u, v) with expensive shortest paths, within an additive stretch of 2. The number of edges in $\bigcup_{C \in S_h} T_C$ is $O(nh)$, since each shortest paths tree contains $n - 1$ edges and the number of trees in the union is $O(h)$.

2.2 Techniques Used in Past Works

Most of the ideas and techniques used in proving our results have already been used in other purely additive all-pairs and pairwise spanner constructions found in the literature. In this section, we give an outline of some of the key results which have crucially used the techniques of Clustering, Path Buying and Shortest Paths Tree Addition.

The clustering phase is common to all the following spanner algorithms. The $(1, 2)$ spanner algorithm [DHZ00, EP04] uses only clustering and shortest paths tree addition. The $(1, 6)$ spanner algorithm of Baswana et al. [BKMP05] uses the techniques of clustering and path buying. The path buying technique was first used in that algorithm. The $(1, 4)$ spanner algorithm of Chechik [Che13] combines all the three techniques. Her algorithm is randomized. Our algorithms for small stretch pairwise spanners in Chapter 3 are very similar to Chechik's algorithm except that our algorithms are deterministic.

The algorithm for $(1, 4k)$ D-spanner that we describe in Chapter 4 also uses all the above three techniques. But the path buying step in that algorithm is not as simple as those in any of the algorithms mentioned above. The path buying step in the D-spanner algorithm resembles the iterated path buying technique used by Cygan et al. [CGK13] to obtain their construction of $(1, 4k)$ $\mathcal{P}$-spanners. We outline that construction as well in this section.

Note that in all of these constructions, the input graph $G = (V, E)$ is assumed to be on n nodes.

An $O(n^{3/2})$-sized $(1, 2)$ all-pairs spanner. A $(1, 2)$ spanner can be constructed in the following way. First perform Clustering with $h = \lceil n^{1/2} \rceil$. Then add the union of shortest paths trees rooted at *all* cluster centers to the cluster subgraph. The resulting graph is output as the spanner. Consider a pair of vertices (u, v). If the u-v shortest path consists only of unclustered nodes, then that path is entirely present in the cluster subgraph. If not, there is some cluster intersecting that path. This guarantees the existence of a u-v path of additive stretch 2 in H using the edges in the shortest paths tree rooted at the center of that cluster. Thus H is a $(1, 2)$ spanner. The total number of clusters is at most $\lceil n^{1/2} \rceil$ and therefore the number of edges in the union of shortest paths trees added is $O(n^{3/2})$. Hence the number of edges in the spanner is $O(n^{3/2})$.

An $O(n^{4/3})$-sized $(1, 6)$ all-pairs spanner. The $(1, 6)$ spanner construction that we describe here is due to Baswana et al. [BKMP05]. The first step is clustering with $h = \lceil n^{1/3} \rceil$. The spanner H is initialised to the cluster subgraph. Then they go over shortest paths in $\mathcal{R}$ and consider each of them for addition to H. With each $\rho \in \mathcal{R}$, they associate two quantities - a cost and a value. The cost of ρ, denoted by $\mathsf{cost}(\rho)$ has the same meaning as in Definition 2.1.3. The value of ρ with respect to a subgraph H, denoted by $\mathsf{value}_H(\rho)$, is the number of pairs of clusters (C_1, C_2) such that the distance between C_1 and C_2 in H decreases by adding ρ to H. A path $\rho \in \mathcal{R}$ is added to H only if it satisfies:

$$\mathsf{cost}(\rho) \leq 3 \cdot \mathsf{value}_H(\rho).$$

This phase is called as the Path Buying phase. The subgraph at the end of the two phases is output as the spanner.

It is not hard to see that the output is a $(1, 6)$ spanner. Let ρ be a u-v shortest path of cost t. There are at least $t/3$ clusters incident on ρ by Lemma 2.1.5. Assume that ρ failed the path buying criterion and was not added to H during the path buying phase. In other

words, $t/3 > \mathsf{value}_H(\rho)$ at the time ρ was considered for addition to H. Let C_u and C_v be the clusters intersecting ρ closest to u and v along ρ, respectively. They then show the existence of a cluster C incident on ρ whose distance in H from C_u and C_v does not decrease by the addition of ρ to H (otherwise $\mathsf{value}_H(\rho)$ will be at least $t/3$). In other words, the C-C_u and C-C_v distances in H are at most the respective distances along ρ. This is illustrated in Figure 2.2.

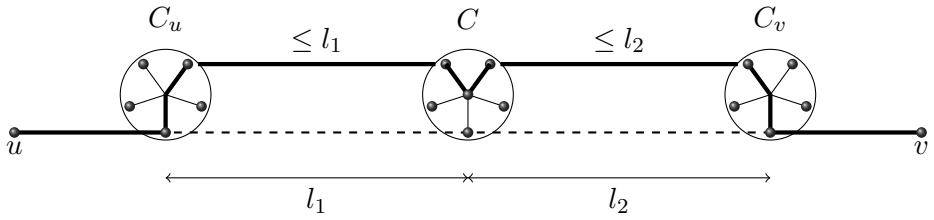

Figure 2.2: *Path in H (thick) from u to v of additive stretch at most 6*

It is clear then, that there is a path in H between u and v of additive stretch 6. The number of edges in the cluster subgraph is bounded by $O(n^{4/3})$ by Lemma 2.1.1. The number of edges added during the Path Buying phase can be bounded by summing up the values at the time of buying, of all the paths that were added. They argue that the distance between two clusters can decrease only a constant number of times during Path Buying and use this to show that the above sum of values of paths bought is proportional to the total number of cluster pairs. As the number of clusters is at most $\lceil n^{2/3} \rceil$, this turns out to be $O(n^{4/3})$. Thus the total number of edges added in the output is also $O(n^{4/3})$.

An $O(n^{1.4} \log^{0.2} n)$-sized $(1,4)$ all-pairs spanner. The first $(1,4)$ spanner construction is due to Chechik [Che13]. The algorithm is randomized and it outputs a $(1,4)$ spanner of expected size $O(n^{1.4} \log^{0.2} n)$ with high probability. We sketch their method here. Let $h = \lceil n^{0.4} \log^{0.2} n \rceil$. Call a vertex *heavy* if its degree is at least h and *light* otherwise.

They initialise H to the empty graph. They select a set of nodes by independently sampling at random every node with probability $1/h$, and designate the chosen nodes as the cluster centers. A heavy node joins the cluster of an arbitrary cluster center neighboring it. The edges connecting such heavy nodes and their cluster center are added to H. All edges incident on light nodes, and all edges incident on heavy nodes with no cluster center in their neighborhoods, are also added to H. It can be seen that these steps are in spirit, equivalent to the clustering phase performed in our algorithms.

In the next phase, they select a set of nodes by independently sampling at random every node with probability h/n and they add the edges in the union of shortest paths trees rooted at these nodes to H.

In the last phase, for each pair of clusters (C_1, C_2), they look at the shortest paths in $\mathcal{R}$ associated with vertex pairs in $C_1 \times C_2$, which in addition, have at most h^3/n heavy nodes on them. They add to H, one such path ρ which is as short as any other among them. The subgraph H at the end of these three phases is output as the spanner.

We show the stretch guarantee in what follows. Consider two clustered heavy nodes u and v. Assume that the u-v shortest path ρ has at most h^3/n heavy nodes on it. Let $C(u)$ and $C(v)$ denote the clusters to which u and v belong. It is not hard to see that the last phase of the above algorithm guarantees a path in H from $C(u)$ to $C(v)$ which is at least as

short as ρ. This implies a path from u to v in H of length at most $|\rho| + 4$. In case the path has light or unclustered heavy endpoints, we can carry out a similar argument by focussing on the clustered heavy nodes on the path closest to the end points. This is because of the fact that the edges incident on light nodes as well as unclustered heavy nodes are present in H.

With high probability, all shortest paths with at least h^3/n heavy nodes are approximated within an additive stretch of 2 after the shortest paths tree addition. This is because, such paths have at least h^4/n nodes in their neighborhood and hence with high probability, there is at least one node among them at which a shortest paths tree is rooted. Thus the subgraph H output is a $(1, 4)$ spanner with high probability.

They show that the expected number of edges added to H in the first phase is $O(nh)$. The expected number of shortest paths trees added during the second phase is h and hence the expected number of edges added to H in that phase is $O(nh)$. The expected number of clusters is n/h. A path having at most t heavy nodes has cost at most t. Thus in the third phase, we add at most one shortest path of cost at most h^3/n for each pair of clusters. From this, it can be seen that the expected number of edges added during this phase is also $O(nh)$. Thus the expected size of the spanner is $O(nh)$, i.e. $O(n^{1.4} \log^{0.2} n)$.

An $O(n^{1+\frac{1}{2k+1}}((4k+2) \cdot |\mathcal{P}|)^{\frac{k}{4k+2}})$-sized $(1, 4k)$ $\mathcal{P}$-spanner. Instead of outlining the construction of the $(1, 4k)$ $\mathcal{P}$-spanner, we will sketch the construction of a $O(n \cdot (|\mathcal{P}| \cdot \log n)^{1/4}))$-sized $(1, 4 \log n)$ $\mathcal{P}$-spanner, which is obtained by setting $k = \lfloor \log n \rfloor$ in the general result, since it has all the main ideas we want to present.

There are only two phases to the algorithm. The first phase is the usual clustering phase with $h = \lceil (|\mathcal{P}| \cdot \log n)^{1/4} \rceil$. The spanner H is initialised to the cluster subgraph G_h.

In the path buying phase, we go over pairs in $\mathcal{P}$. For a specific pair $(u, v) \in \mathcal{P}$, let ρ_{uv} always denote the path from u to v that we *desire* to add to H.

The path ρ_{uv} is initialised to the u-v shortest path. We add ρ_{uv} to H if it satisfies:

$$\mathsf{cost}(\rho_{uv}) \leq 12 \cdot \sqrt{\mathsf{value}_H(\rho_{uv})},$$

where the functions $\mathsf{cost}(.)$ and $\mathsf{value}_H(.)$ have the same meanings as the ones used in the description of $(1, 6)$ spanner. If the criterion is violated, we compute an alternate u-v path ρ' such that (i) ρ' is at most 4 longer than ρ_{uv}, and (ii) $\mathsf{cost}(\rho') \leq \mathsf{cost}(\rho_{uv})/2$. We then set ρ_{uv} to ρ' and try to add it to H using the same path buying criterion again. We repeat these steps for a particular $(u, v) \in \mathcal{P}$ until we can add some u-v path ρ_{uv} to H.

We will need to recompute ρ_{uv} at most $\log n$ times for each $(u, v) \in \mathcal{P}$ since the cost of ρ_{uv}, which can initially be at most n, reduces by a constant factor each time it is recomputed. The length of ρ_{uv} increases by at most 4 each time it is recomputed. Thus, by the end of the path buying phase, it is guaranteed that there is a u-v path in H of additive stretch at most $4 \log n$ for every $(u, v) \in \mathcal{P}$. This proves that H is a $(1, 4 \log n)$ $\mathcal{P}$-spanner of G.

We will now give a rough reasoning as to why such an alternate u-v path exists when ρ_{uv} fails the path buying criterion. Consider a path ρ_{uv} which violates the path buying criterion. Let its cost be t. Let L denote the longest prefix of ρ_{uv} with exactly $\lfloor t/4 \rfloor$ missing edges and let R denote its longest suffix with the same number of missing edges. Since there are at least $t/4$ clustered nodes on L and R each, there are at least $t/12$ clusters intersecting each of them. It can be seen that there is a cluster C_1 intersecting L and a cluster C_2 intersecting R such that the distance between C_1 and C_2 does not decrease by adding ρ_{uv}. If it were not true, the value of ρ_{uv} would have been at least $t^2/144$ which is against our premise that ρ_{uv} is a path which violated the path buying criterion.

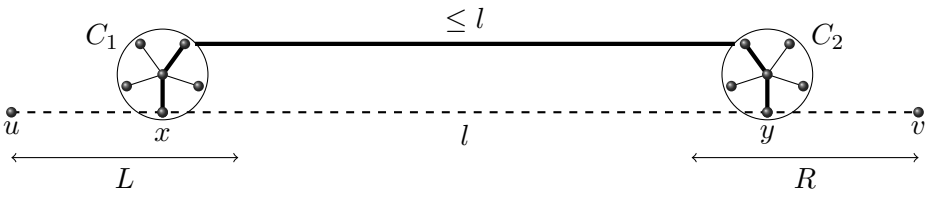

Figure 2.3: *Path in H from C_1 to C_2 of length at most l*

As is illustrated by Figure 2.3, this means that there is a path in H from C_1 to C_2 whose length is at most the distance between the same along ρ_{uv}.

Let x be the node in the intersection of C_1 and ρ_{uv} that is closest to u along ρ_{uv} and y be the node in the intersection of C_2 and ρ_{uv} that is closest to v along ρ_{uv} (as indicated in the figure). The new u-v path ρ' which we compute is hence a concatenation of (i) the subpath of L from u to x, (ii) the shortest path in H from x to y of length at most $l + 4$, and (iii) the subpath of R from y to v. Clearly, ρ' is at most 4 longer than ρ_{uv}. The cost of ρ' is at most half the cost of ρ_{uv} since the only missing edges in ρ' come from the subpaths corresponding to L and R. One more thing that needs to be taken care of is the fact that any cluster shares at most three nodes with ρ'. It is easy to see that this can also be achieved without disturbing the other properties of ρ'.

Thus we get a $(1, 4\log n)$ $\mathcal{P}$-spanner of G. The number of edges added during clustering is $O(nh)$ as usual. Using arguments similar to the ones used to bound the number of edges added in the path buying phase of the $(1, 6)$ spanner, we can show that the number of edges added to H during the Path Buying phase here too is $O(nh)$. We omit the details of that. Thus the final subgraph H returned is a $(1, 4\log n)$ $\mathcal{P}$-spanner of G having $O(n\cdot(|\mathcal{P}|\cdot\log n)^{1/4})$ edges.

Chapter 3

Small Stretch Pairwise Spanners

In this Chapter, we present our algorithms for constructing pairwise spanners when the pairs to be approximated are specified explicitly. The inputs to our algorithms are a graph $G = (V, E)$ and a set $\mathcal{P} \subseteq V \times V$. We consider the cases when the set of pairs $\mathcal{P}$ is (i) an arbitrary subset of $V \times V$, (ii) $S \times V$ for some $S \subseteq V$, and (iii) $V \times V$. For the first two cases, we obtain a $(1, 2)$ $\mathcal{P}$-spanner and for the last case, we obtain a $(1, 4)$ $\mathcal{P}$-spanner.

In all these algorithms, we first perform the steps in Clustering and Shortest Paths Tree Addition phases with suitable parameters. The path buying step here amounts to simply adding the shortest paths corresponding to some specific pairs in $\mathcal{P}$.

3.1 A (1,2) $\mathcal{P}$-spanner

In this section, we consider the problem of computing a (1,2) $\mathcal{P}$-spanner of $G = (V, E)$, where $\mathcal{P} \subseteq V \times V$ specified as part of the input is arbitrary. The following is the algorithm to achieve that. Its input is an undirected unweighted graph $G = (V, E)$ on n vertices, and a set $\mathcal{P} \subseteq V \times V$ of pairs whose distances are to be approximated. Note that in the Path Buying step, when we say that a path ρ is added to H, we mean H becoming $H \cup \{\rho\}$.

1. Initialize H to the empty graph and set h to $\lceil (|\mathcal{P}| \log n)^{1/3} \rceil$.
2. *Clustering and SPT Addition.* Perform the steps in the Clustering and Shortest Paths Tree Addition phases (as described in Section 2.1) with h as the parameter. This computes G_h, $\mathcal{C}_h$ and S_h.
   - Add all the edges in G_h to H.
   - Add all the edges in shortest paths trees rooted at the centers of the clusters in S_h to H.
3. *Path Buying.* For each $(u, v) \in \mathcal{P}$ associated with a cheap path, do:
   - Add ρ to H, where ρ is the u-v shortest path.
4. Return H

The fact that H returned by the algorithm is a (1,2) $\mathcal{P}$-spanner is proved in the following lemma. This lemma also bounds the number of edges in H.

Lemma 3.1.1. *The subgraph H returned by the above algorithm is a $(1,2)$ $\mathcal{P}$-spanner of G and it has $O(n \cdot (|\mathcal{P}| \log n)^{1/3})$ edges.*

Proof. First we will prove that H is a (1,2) $\mathcal{P}$-spanner of G. Consider $(u,v) \in \mathcal{P}$ and let ρ be the shortest u-v path. If ρ is expensive, $d_H(u,v) \leq d_G(u,v) + 2$ by Lemma 2.1.7, since H contains the union of all shortest paths trees rooted at centers of clusters in S_h. If not, ρ is added to H in the Path Buying step and hence $d_H(u,v) = d_G(u,v)$. Thus for all pairs $(u,v) \in \mathcal{P}$, we have $d_H(u,v) \leq d_G(u,v) + 2$. So H is a (1,2) $\mathcal{P}$-spanner of G.

The number of edges in G_h is $O(nh)$ by Lemma 2.1.1. By Lemma 2.1.6, the number of clusters in S_h is $O(h)$. Hence, the total number of edges in the union of shortest paths trees rooted at the centers of clusters in S_h is $O(nh)$. Thus the total number of edges added to H in Step 2 of the algorithm is $O(nh)$. We bound the number of edges added in the Path Buying step in the following way. At most $|\mathcal{P}|$ paths, whose costs are each bounded by $(n \log n)/h^2$, are added to H in that step. Therefore the number of edges added is bounded by $|\mathcal{P}| \cdot ((n \log n)/h^2)$, which is $O(nh)$. Hence the total number of edges in H is $O(nh)$, which is the same as $O(n \cdot (|\mathcal{P}| \log n)^{1/3})$. □

This proves Theorem 1.0.1 from Chapter 1.

3.2 A (1,2) $S \times V$-spanner

In this section, we consider the problem of computing a (1,2) $S \times V$-spanner of $G = (V, E)$, where $S \subseteq V$ specified as part of the input is arbitrary. The following is the algorithm to achieve that. Here in the Path Buying step, for each pair (s, C) where $s \in S$ and C is a cluster, we add at most one cheap path among those from s to C.

1. Initialize H to the empty graph and set h to $\lceil (|S| \log n)^{1/4} \rceil$.

2. *Clustering and SPT Addition.* Perform the steps in the Clustering and Shortest Paths Tree Addition phases (as described in Section 2.1) with h as the parameter. This computes G_h, $\mathcal{C}_h$ and S_h.

   - Add all the edges in G_h to H.
   - Add all the edges in shortest paths trees rooted at the centers of the clusters in S_h to H.

3. *Path Buying.* For each $(s, C) \in S \times \mathcal{C}_h$, do:

   - For some $v \in C$, add the s-v shortest path ρ to H, such that:
     (a) ρ is a cheap path, and
     (b) ρ is as short as any other cheap path associated with pairs in $\{s\} \times C$,

     if such a $v \in C$ exists.

4. Return H

The fact that H returned by the algorithm in this case is a (1,2) $S \times V$-spanner is proved in the following lemma. This lemma also bounds the size of H returned.

Lemma 3.2.1. *The subgraph H returned by the above algorithm is a $(1,2)$ $S \times V$-spanner of G and it has $O(n \cdot (n|S| \log n)^{1/4})$ edges.*

Proof. First we will prove that H is a (1,2) $S \times V$-spanner of G. Let $(s, v) \in S \times V$ be such that v is clustered. Let ρ be its associated shortest path. If ρ is expensive, $d_H(u, v) \leq d_G(u, v) + 2$ by Lemma 2.1.7, since H contains the union of all shortest paths trees rooted at centers of clusters in S_h. Otherwise, two cases arise depending on whether ρ was added to H in the Path Buying Step or not. If ρ got added, $d_H(s, v) = d_G(s, v)$. If not, it means that there is a $v' \in C(v)$ such that the s-v' shortest path ρ' is cheap, $d_G(s, v') \leq d_G(s, v)$ and ρ' got added to H in the Path Buying Step. This would imply that $d_H(s, v) \leq d_G(s, v') + 2 \leq d_G(s, v) + 2$ as illustrated in Figure 3.1.

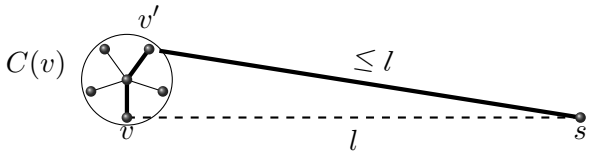

Figure 3.1: *Path in H (thick) from s to v via v' of length at most $l + 2$*

Thus whenever $s \in S$ and v is clustered, $d_H(s, v) \leq d_G(s, v) + 2$. If v is unclustered, consider the first clustered node on the path from v to s. Call it u. The section of the s-v shortest path between u and v is present in H, since all its edges have at least one endpoint unclustered. From what we have already proved, $d_H(s, u) \leq d_G(s, u) + 2$. Therefore it follows that,

$$d_H(s, v) \leq d_H(s, u) + d_H(u, v) \leq d_G(s, u) + 2 + d_G(u, v) \leq d_G(s, v) + 2.$$

So, $d_H(s, v) \leq d_G(s, v) + 2$ for all $(s, v) \in S \times V$ and hence H is a (1,2) $S \times V$-spanner of G.

Using arguments similar to the ones used in the proof of Lemma 3.1.1, we can see that the number of edges added to H in Step 2 is $O(nh)$. We bound the number of edges added in the Path Buying step in the following way. For each pair $(s, C) \in S \times \mathcal{C}_h$, at most one cheap path is added to H. As the number of clusters is at most n/h, the number of such distinct cheap paths is bound by $|S| \cdot n/h$. Therefore the number of edges added in that step is bounded by $(|S| \cdot n/h) \cdot ((n \log n)/h^2)$, which is $O(nh)$. Hence the total number of edges in H is $O(nh)$, which is the same as $O(n \cdot (n|S| \log n)^{1/4})$. □

We can thus conclude Theorem 1.0.2 from the proof of above lemma.

3.3 A (1,4)-spanner

Now we consider the problem of computing a (1,4) all pairs spanner of a graph $G = (V, E)$. The following is the algorithm to achieve that.

1. Initialize H to the empty graph and set h to $\lceil n^{0.4} \log^{0.2} n \rceil$.

2. *Clustering and SPT Addition.* Perform the steps in the Clustering and Shortest Paths Tree Addition phases (as described in Section 2.1) with h as the parameter. This computes G_h, $\mathcal{C}_h$ and S_h.

   - Add all the edges in G_h to H.
   - Add all the edges in shortest paths trees rooted at the centers of the clusters in S_h to H.

3. *Path Buying.* For each $(C_1, C_2) \in \mathcal{C}_h \times \mathcal{C}_h$, do:

   - For some $(u, v) \in C_1 \times C_2$, add the u-v shortest path ρ to H, such that:
     (a) ρ is a cheap path, and
     (b) ρ is as short as any other cheap path associated with pairs in $C_1 \times C_2$,
     if such a $(u, v) \in C_1 \times C_2$ exists.

4. Return H

The fact that H returned by the algorithm in this case is a (1,4)-spanner is proved in the following lemma. This lemma also bounds the size of H returned.

Lemma 3.3.1. *The subgraph H returned by the above algorithm is a $(1, 4)$-spanner of G and its size is $O(n^{1.4} \log^{0.2} n)$.*

Proof. First we will prove that H is a (1,4)-spanner of G. Let (u, v) be a pair of nodes in which both u and v are clustered and let ρ be the u-v shortest path. If ρ is expensive, addition of shortest paths trees in Step 2 ensures that $d_H(u, v) \leq d_G(u, v) + 2$. Otherwise, two cases arise depending on whether ρ was added to H or not. If ρ got added $d_H(u, v) = d_G(u, v)$. If not, it means that there is a $u' \in C(u)$ and $v' \in C(v)$ such that the u'-v' shortest path ρ' is cheap, $d_G(u', v') \leq d_G(u, v)$ and ρ' was added to H. This would imply that $d_H(u, v) \leq d_G(u, v) + 4$ as illustrated by Figure 3.2.

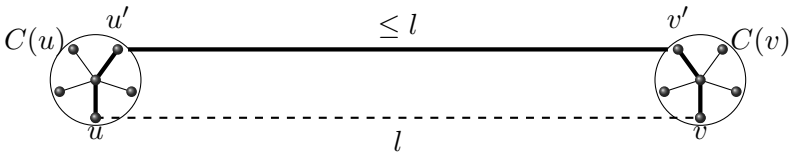

Figure 3.2: *Path in H (thick) from u to v (via u' and v') of length at most $l + 4$*

Thus whenever u and v are clustered, $d_H(u, v) \leq d_G(u, v) + 4$. If either v or u is unclustered, consider the clustered nodes on the u-v shortest path closest to u and v. Call them y and w. The sections of the u-v shortest path between u and y as well as w and v are contained in H, since all their edges have at least one endpoint unclustered. From what we have already

proved, $d_H(y,w) \leq d_G(y,w) + 4$. Therefore it follows that,

$$\begin{aligned} d_H(u,v) &\leq d_H(u,y) + d_H(y,w) + d_H(w,v) \\ &\leq d_G(u,y) + d_G(y,w) + 4 + d_G(w,v) \\ &\leq d_G(u,v) + 4. \end{aligned}$$

So, $d_H(u,v) \leq d_G(u,v) + 4$ for all $(u,v) \in V \times V$ and hence H is a (1,4)-spanner of G.

Using arguments similar to the ones used in the proof of Lemma 3.1.1, we can see that the number of edges added to H in Step 2 is $O(nh)$. We bound the number of edges added during the Path Buying step as follows. For each pair (C_1, C_2) where C_1 and C_2 are clusters, at most one cheap path was added to H. As the number of clusters is at most n/h, the number of such distinct pairs is bound by n^2/h^2. Therefore, the number of edges added to H in that step is bounded by $(n^2/h^2) \cdot ((n \log n)/h^2)$, which is $O(nh)$. Therefore the total number of edges in H is $O(nh)$, which is the same as $O(n^{1.4} \log^{0.2} n)$. □

We can thus conclude the following theorem.

Theorem 3.3.2. *There is a polynomial time algorithm which takes a graph $G = (V, E)$ on n nodes as input and computes an $O(n^{1.4} \log^{0.2} n)$-sized $(1, 4)$ spanner of G.*

With this, we conclude our discussion on pairwise spanners when the pairs to be approximated are specified explicitly.

Chapter 4

Additive D-spanners

In this chapter, we consider the problem of computing a (1,t) $\mathcal{P}$-spanner of a given undirected unweighted graph $G = (V, E)$ on n vertices where $\mathcal{P} = \{(u, v) \in V \times V : d_G(u, v) \geq D\}$ and $D \in [1, n]$ is an integral distance threshold specified as part of the input. We call such a pairwise spanner as a (1,t) D-spanner.

In Section 4.1, we describe the construction of a $(1, 4)$ D-spanner and in Section 4.2, we generalize this to $(1, 4k)$ D-spanners for all integers $k \geq 1$.

4.1 A (1,4) D-spanner

In this section we describe our algorithm to compute a (1,4) D-spanner of a given graph. The inputs to the algorithm are a graph $G = (V, E)$ and an integer distance threshold $D \in [1, n]$.

The path buying phase in this algorithm is similar to that used in the $(1, 6)$ spanner algorithm of [BKMP05] which was described in Section 2.2. In addition to the function $\mathsf{cost}(.)$ (Definition 2.1.3), we also use a new function $\mathsf{value}_H(.)$. We define it below.

Let ρ be a path (not necessarily shortest) in G. For a vertex v on ρ and a cluster C intersecting ρ, let $d_\rho(v, C)$ denote the distance along ρ between v and C, i.e., the length of the smallest subpath of ρ between v and a node in C.

Definition 4.1.1. *For any path ρ in G and subgraph H of G, $\mathsf{value}_H(\rho)$ denotes the number of pairs (v, C) such that $d_\rho(v, C) < d_H(v, C)$, where vertex v and cluster C are incident on ρ.*

That is, for a path ρ, $\mathsf{value}_H(\rho)$ counts the (vertex, cluster) pairs incident on ρ whose distance along ρ is strictly smaller than their distance in the subgraph H. Our algorithm to compute a (1,4) D-spanner of G is presented below.

1. Initialise H to the empty graph and set h to $\lceil \sqrt{n} \cdot (\log n/D)^{1/4} \rceil$.

2. *Clustering and SPT Addition.* Perform the steps in the Clustering and Shortest Paths Tree Addition phases (as described in Section 2.1) with h as the parameter. This computes G_h, $\mathcal{C}_h$ and S_h.

   - Add all the edges in G_h to H.
   - Add all the edges in shortest paths trees rooted at the centers of the clusters in S_h to H.

3. *Path Buying.* For each $(u, v) \in V \times V$ such that $d_G(u, v) \geq D$, do:

   - Let ρ be the shortest u-v path.
   - Add ρ to H, if it satisfies:

$$\mathsf{cost}(\rho) \leq \mathsf{value}_H(\rho) \cdot \sqrt{\frac{\log n}{D}}.$$

4. Return H

The fact that H returned by the algorithm in this case is a (1,4) D-spanner is proved in the following lemma. The proof of this lemma has most of the major ideas involved in our general result on D-spanners. The proof sketch is as follows. As in the case of our $\mathcal{P}$-spanners of the previous section, we argue that the expensive paths are all *taken care of* by the SPT Addition Phase. All cheap paths of length at least D are considered for addition to H in Path Buying Phase. If a cheap path does not get added during this phase, we argue that there is a path between its endpoints whose length is at most 4 more and which is entirely present in H.

Lemma 4.1.2. *The subgraph H returned by the above algorithm is a (1,4) D-spanner of G.*

Proof. Consider a pair $(u, v) \in V \times V$ such that $d_G(u, v) \geq D$. Let ρ be the u-v shortest path. Note that $h = \lceil \sqrt{n} \cdot (\log n/D)^{1/4} \rceil$ here.

If $\mathsf{cost}(\rho) \geq \sqrt{D \log n}$, ρ has cost at least $(n \log n)/h^2$ and therefore by Lemma 2.1.7, $d_H(u, v) \leq d_G(u, v) + 2$.

If $\mathsf{cost}(\rho) < \sqrt{D \log n}$, there are two cases to consider. One of them is that ρ got added to H in the path buying phase. Then $d_H(u, v) = d_G(u, v)$.

If ρ did not get added to H in the path buying phase, we can infer that ρ had violated the path buying criterion when it was considered for addition to H. Combining this with the fact that $\mathsf{cost}(\rho) < \sqrt{D \log n}$, we get:

$$\sqrt{D \log n} > \mathsf{cost}(\rho) > \mathsf{value}_H(\rho) \cdot \sqrt{\frac{\log n}{D}}. \tag{4.1}$$

From this, we get that $\mathsf{value}_H(\rho) < D$.

Let x and y be the first and last clustered nodes on ρ respectively. Let C_1 and C_2 be the respective clusters to which x and y belong. Without loss of generality, we may assume that C_1 and C_2 are distinct. Otherwise, a u-v path of length at most $d_G(u, v) + 1$ is present in H and we would be done. Let L, M and R denote the u-x, x-y and y-v subpaths of ρ,

respectively. The subpaths L and R are entirely present in G_h (and therefore in H, at the time ρ was considered) because all their edges have at least one endpoint unclustered.

We claim that there exists a node w on ρ such that:

$$d_H(w, C_1) \leq d_\rho(w, C_1) \text{ and } d_H(w, C_2) \leq d_\rho(w, C_2).$$

If not, for each node z on ρ, we have either $d_H(z, C_1) > d_\rho(z, C_1)$ or $d_H(z, C_2) > d_\rho(z, C_2)$. Since there are at least D nodes on ρ, this would imply that $\mathsf{value}_H(\rho) \geq D$. This contradicts the premise that $\mathsf{value}_H(\rho) < D$.

Now we will show that the existence of such a node w guarantees a path in H between u and v of length at most $d_G(u, v) + 4$.

Assume that w is a node on L. Let $d_\rho(w, y)$ be l. Therefore, from the definition of w, $d_H(w, C_2) \leq d_\rho(w, C_2) \leq l$. Then, as illustrated by Figure 4.1, we can see that there exists a path from u to v in H which is a concatenation of (i) the subpath of L from u to w, (ii) a shortest path in H from w to y, and (iii) the subpath of R from y to v. It is easy to see that this path has length at most $d_G(u, v) + 2$.

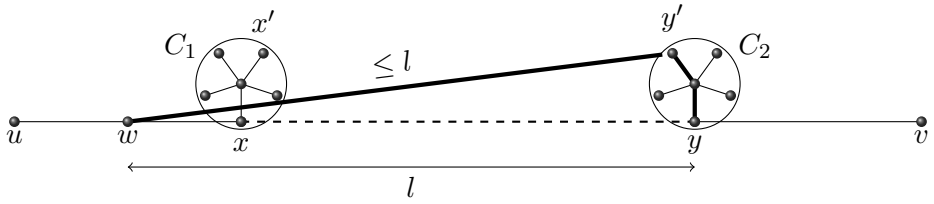

Figure 4.1: *Path in H (thick) from w to y (via y') of length at most $l + 2$*

The case when w is on R is symmetric and there also we get $d_H(u, v) \leq d_G(u, v) + 2$.

Finally, assume that w is on the subpath M. Let $d_\rho(w, x)$ be l_1 and $d_\rho(w, y)$ be l_2. Using arguments similar to above, we can infer that $d_H(w, C_1) \leq l_1$ and $d_H(w, C_2) \leq l_2$. Then, as illustrated by Figure 4.2, we can see that there exists a path from u to v in H which is a concatenation of (i) L, (ii) a shortest path in H from x to w, (iii) a shortest path in H from w to y, and (iii) R. It is easy to see that this path has length at most $d_G(u, v) + 4$.

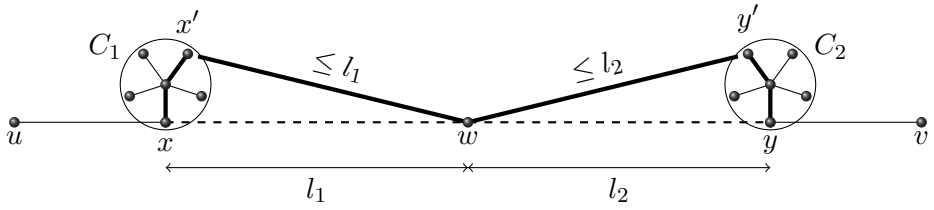

Figure 4.2: *Path in H (thick) from x to y via x',w and y' of length at most $l_1 + l_2 + 4$*

Thus we have proved that for all $(u, v) \in V \times V$ such that $d_G(u, v) \geq D$, $d_H(u, v) \leq d_G(u, v) + 4$. □

The following lemma bounds the number of edges in the subgraph H returned. We say that a pair $(v, C) \in V \times \mathcal{C}_h$ *supports* a path ρ in a graph H, if that vertex-cluster pair contributes 1 to $\mathsf{value}_H(\rho)$, or in other words, if $d_\rho(v, C) < d_H(v, C)$.

Lemma 4.1.3. *The subgraph H returned by our algorithm has $O(n^{3/2} \cdot (\log n/D)^{1/4})$ edges.*

Proof. The number of edges added to H in Step 2 is $O(nh)$, using arguments similar to those in Lemma 3.1.1. The number of edges added to H in the Path Buying phase is at most the sum of costs of paths added during that phase. Since any path that gets added to H satisfies the Path Buying condition, we have the following hold true:

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

where the summations are both over the set of paths $\rho \in \mathcal{R}$ that got added to H during the Path Buying step and H_ρ denotes the subgraph H at the time ρ was considered. From the definition of the function $\mathsf{value}_H(.)$, we can see that $\mathsf{value}_{H_\rho}(\rho)$ is the number of pairs in $V \times \mathcal{C}_h$ that *support* ρ in H_ρ. Therefore, we get:

$$\begin{aligned}\sum_{\rho} \mathsf{value}_{H_\rho}(\rho) &= \sum_{\rho} \text{number of pairs } (v, C) \in V \times \mathcal{C}_h \text{ that support } \rho \text{ for addition to } H_\rho \\ &= \sum_{(v,C)} \text{number of paths } \rho \in \mathcal{R} \text{ that } (v, C) \text{ supported for addition to } H,\end{aligned}$$

where the first and second summations are over all the paths in $\mathcal{R}$ that got added to H and the third one is over all pairs $(v, C) \in V \times C_h$.

We now claim that a pair (v, C) could have supported at most three path in $\mathcal{R}$ for addition to H. After the first path $\rho \in \mathcal{R}$ which (v, C) supported gets added to H, the distance between them in the new graph is $d_\rho(v, C)$ which is at most $d_G(v, C) + 2$ since, ρ is a shortest path in G. This distance can decrease at most 2 more times. Hence, the total number of shortest paths that a cluster-node pair supports is at most 3. The total number of cluster-node pairs is at most n^2/h. Thus the sum of values of shortest paths added to H is at most $3 \cdot n^2/h$. Therefore, the sum of costs of the paths added to H during the Path Buying step is at most

$$\sqrt{\frac{\log n}{D}} \cdot \sum_{\rho} \mathsf{value}_{H_\rho}(\rho) = \sqrt{\frac{\log n}{D}} \cdot \frac{3n^2}{h}.$$

Substituting the value of h, we can see that this is $O(nh)$.

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

4.2 A (1,4k) D-spanner

We are now ready to describe our algorithm to construct a (1,4k) D-spanner of any graph G for any integer $k \geq 1$. The inputs to this algorithm are a graph $G = (V, E)$, an integral distance threshold $D \in [1, n]$ and an integer $k \geq 1$. Let $\mathcal{P} = \{(u, v) \in V \times V : d_G(u, v) \geq D\}$.

The clustering phase and shortest paths tree addition phase of this algorithm are similar to those of the other algorithms we have seen so far. In the path buying phase of this algorithm, for each pair of nodes $(u, v) \in \mathcal{P}$ such that the u-v shortest path ρ is cheap, we try adding ρ to the current subgraph based on a path buying criterion. If it is not *affordable* in our current subgraph, we use a sub-routine NextPath to get another u-v path that is *slightly longer* and *less costlier*. We then see if this new path satisfies our path-buying criterion. This continues until we find an affordable u-v path.

The path buying criterion here makes use of the function $\mathsf{value}_H(.)$ (see Definition 4.1.1) as well as a function $\mathsf{cost}_H(.)$ which is defined as follows.

Definition 4.2.1. *The cost of a path ρ with respect to a subgraph H of G, denoted as $\mathsf{cost}_H(\rho)$, is defined as the number of edges of ρ which are absent from H.*

The algorithm description follows.

1. Initialise H to the empty graph and set h to $\left\lceil \sqrt{(4k+3)\cdot n} \cdot \left(\frac{\log n}{D}\right)^{k/(2k+2)} \right\rceil$.
2. *Clustering and SPT Addition.* Perform the steps in the Clustering and Shortest Paths Tree Addition phases (as described in Section 2.1) with h as the parameter. This computes G_h, $\mathcal{C}_h$ and S_h.
   - Add all the edges in G_h to H.
   - Add all the edges in shortest paths trees rooted at the centers of the clusters in S_h to H.
3. *Path Buying.* For each (u, v) such that $d_G(u, v) \geq D$, do:
   - Let ρ be the shortest u-v path.
   - If $\mathsf{cost}(\rho) < \left(\frac{D^{k/(k+1)} \log^{1/(k+1)} n}{4k+3}\right)$, do:
     (a) While $\mathsf{cost}_H(\rho) > 6 \cdot \mathsf{value}_H(\rho) \cdot \left(\frac{\log n}{D}\right)^{k/(k+1)}$, do:
         – $\rho = \textsc{NextPath}(\rho, H)$
     (b) Add ρ to H.
4. Return the subgraph H

This completes our algorithm. We first prove that there is a polynomial time subroutine NextPath, which when invoked on a u-v path ρ that has violated the path buying criterion, returns another u-v path whose length is at most 4 more than ρ and whose cost is smaller than that of ρ by a factor of $\tilde{O}(D^{1/(k+1)})$. We do this in Lemma 4.2.2. The proof sketch is as follows. NextPath is invoked on a u-v path ρ and a subgraph H only if it fails the path buying criterion with respect to H. We first prove that there exists two clusters incident on ρ, *close* to its endpoints, and a node on ρ such that the distance of this node from either cluster in H is at most the respective distance between the same along ρ. Using this observation, we then prove that it is possible to efficiently construct an alternate u-v path which satisfies our requirements.

Lemma 4.2.2. *There is a sub-routine* NextPath *which when invoked on a u-v path ρ and a graph H in the Path Buying phase, computes in polynomial time, another u-v path ρ' such that:*

1. $\mathsf{cost}_H(\rho') \leq \dfrac{\mathsf{cost}_H(\rho)}{D^{1/(k+1)} \cdot \log^{k/(k+1)} n}$
2. $|\rho'| \leq |\rho| + 4$
3. *Any cluster C shares at most 3 nodes with ρ'*

Proof. Let u and v be the endpoints of ρ. We assume that ρ satisfies the property that any cluster shares at most 3 nodes with ρ. This is true the first time NextPath is called on a path ρ between u and v, since ρ at that time is the u-v shortest path. We will prove that this property holds for the new path ρ' as well.

Let c denote $\mathsf{cost}_H(\rho)$ and t denote $D^{1/(k+1)} \log^{k/(k+1)} n$. View the path as *going from* u to v. Let L denote the longest prefix of ρ containing $\lfloor c/2t \rfloor$ missing edges in H and let R denote its longest suffix with the same number of missing edges. Let $\mathcal{F}_u$ and $\mathcal{F}_v$ be the sets of clusters intersecting L and R respectively. We may assume without loss of generality that $|\mathcal{F}_u| \leq |\mathcal{F}_v|$.

The sub-routine NextPath was invoked on ρ and H because ρ violated the Path Buying criterion with respect H in the Path Buying phase of the algorithm.

We claim that there exists a node w on ρ, clusters $C_1 \in \mathcal{F}_u$ and $C_2 \in \mathcal{F}_v$ such that:

$$d_H(w, C_1) \leq d_\rho(w, C_1) \text{ and } d_H(w, C_2) \leq d_\rho(w, C_2).$$

If this were not true, then for every node z on ρ there are at least $|\mathcal{F}_u|$ clusters C such that $d_H(z, C) > d_\rho(z, C)$. This means that each node on ρ contributes at least $|\mathcal{F}_u|$ to $\mathsf{value}_H(\rho)$. Thus,

$$\mathsf{value}_H(\rho) \geq D \cdot |\mathcal{F}_u|, \tag{4.2}$$

as there are at least D nodes on ρ. Since L and R each contain $\lfloor c/2t \rfloor$ missing edges, they have at least $\lfloor c/2t \rfloor + 1$ clustered nodes each. Thus $|\mathcal{F}_u|$ and $|\mathcal{F}_v|$ are both at least $c/(6t)$ since, we have assumed that any cluster shares at most 3 nodes with ρ. Substituting this in equation 4.2, we get that:

$$\mathsf{value}_H(\rho) \geq \frac{c \cdot D}{6 \cdot D^{1/(k+1)} \log^{k/(k+1)} n} \implies \mathsf{cost}_H(\rho) \leq 6 \cdot \mathsf{value}_H(\rho) \cdot \left(\frac{\log n}{D}\right)^{k/(k+1)}.$$

This is a contradiction, as we know that ρ did not satisfy the Path Buying criterion with respect to the graph H.

Now we will show that the existence of such a triplet C_1, C_2 and w is enough to ensure the existence of a path ρ' as in the statement of the lemma. Let x be the node in C_1 closest to u along ρ and y be the node in C_2 closest to v along ρ. We have three cases depending on the position of w on ρ.

- If w lies in between u and x, ρ' is the concatenation of the (i) subpath of ρ from u to w, (ii) shortest path in H from w to y, and (iii) subpath of ρ from y to v.
- If w lies in between y and v, ρ' is the concatenation of the (i) subpath of ρ from u to x, (ii) shortest path in H from x to w, and (iii) subpath of ρ from w to v.
- If w is in between x and y, ρ' is the concatenation of the (i) subpath of ρ from u to x, (ii) shortest path in H from x to y via w, and (iii) subpath of ρ from y to v.

Using the same arguments as those used in the proof of Lemma 4.1.2, we can show that $|\rho'| \leq |\rho| + 2$ (refer Figure 4.1) in the first two cases and that $|\rho'| \leq |\rho| + 4$ (refer Figure 4.2) in the third case. In all the three cases, the portions of ρ' with missing edges are in subpaths of L or R. Thus, the total number of edges missing in ρ' is at most c/t. Therefore, we have proved the existence of a ρ' which satisfies:

$$\mathsf{cost}_H(\rho') \leq \frac{\mathsf{cost}_H(\rho)}{D^{1/(k+1)} \cdot \log^{k/(k+1)} n} \text{ and } |\rho'| \leq |\rho| + 4.$$

Now, assume that there exists some cluster C that shares more than 3 nodes with ρ'. We let a and b denote the nodes in the intersection of C and ρ', closest to u and v along ρ', respectively. If we replace the subpath of ρ' between a and b (which is of length at least 3) with the path between them in H (of length at most 2 by Lemma 2.1.2), we get an alternate u-v path whose cost and length are at most those of ρ'. It also has the additional property that C shares at most 3 nodes with it. We can replace ρ' by this path. We can do this for any cluster which shares more than 3 nodes with ρ'. So, we can conclude that ρ' is a path satisfying all the three properties in the statement of the lemma.

The subroutine NextPath can first find C_1, C_2 and w as above by iterating over relevant triplets, each consisting of a pair of clusters from $\mathcal{F}_u \times \mathcal{F}_v$ and a node on ρ, and then compute ρ' using it as described. □

We are now ready to prove that the algorithm returns a $(1, 4k)$ D-spanner of the input graph.

Lemma 4.2.3. *The subgraph H returned by the above algorithm is a $(1, 4k)$ D-spanner of G.*

Proof. Consider a pair $(u, v) \in V \times V$ such that $d_G(u, v) \geq D$. Let ρ be the shortest path associated with it.

Note that $h = \left\lceil \sqrt{(4k+3) \cdot n} \cdot (\log n/D)^{k/(2k+2)} \right\rceil$.

Therefore if $\mathsf{cost}(\rho) \geq \frac{D^{k/(k+1)} \log^{1/(k+1)} n}{4k+3}$, ρ is an expensive path and therefore by Lemma 2.1.7, $d_H(u, v) \leq d_G(u, v) + 2$.

If $\mathsf{cost}(\rho) < \frac{D^{k/(k+1)} \log^{1/(k+1)} n}{4k+3}$, ρ is considered in the Path Buying phase. The number of times that the procedure NextPath is called in the iteration corresponding to ρ is at most k. This can be shown as follows. From the previous lemma, we know that with each invocation of NextPath, the cost of the u-v path under consideration reduces by a factor of $D^{1/(k+1)} \cdot \log^{k/(k+1)} n$. Thus, after k invocations, the cost of the u-v path becomes:

$$\frac{D^{k/(k+1)} \log^{1/(k+1)} n}{(4k+3) \cdot (D^{1/(k+1)} \cdot \log^{k/(k+1)} n)^k} < 1.$$

This means that the path is entirely present in H after k invocations of NextPath. Since in each invocation, the length of the path increases by at most 4, we can conclude that $d_H(u, v) \leq d_G(u, v) + 4k$ at the end of the iteration of the main loop corresponding to the pair (u, v). □

We now bound the number of edges in the subgraph returned by the algorithm. Recall that a pair $(v, C) \in V \times \mathcal{C}_h$ *supports* a path ρ in H if this pair of vertex and cluster contributes 1 to $\mathsf{value}_H(\rho)$.

Lemma 4.2.4. *The subgraph H returned by the above algorithm has $\tilde{O}(n^{3/2}/D^{k/(2k+2)})$ edges.*

Proof. The number of edges added to H in Step 2 is $O(nh)$. The number of edges added to H in the Path Buying phase is bounded by $\sum_{\rho} \mathsf{cost}_{H_\rho}(\rho)$, where the sum is over all ρ that got added to H and H_ρ denotes the state of H when ρ got added. All the paths ρ that got added to H satisfied the path-buying condition

$$\mathsf{cost}_{H_\rho}(\rho) \leq 6 \cdot \mathsf{value}_{H_\rho}(\rho) \cdot (\log n/D)^{k/(k+1)}.$$

Thus we have that,

$$\sum_{\rho} \mathsf{cost}_{H_\rho}(\rho) \leq 6 \cdot \left(\frac{\log n}{D}\right)^{k/(k+1)} \cdot \sum_{\rho} \mathsf{value}_{H_\rho}(\rho), \tag{4.3}$$

where the sum is over the paths ρ added to H.

Since $\mathsf{value}_{H_\rho}(\rho)$ is the number of pairs $(v, C) \in V \times \mathcal{C}_h$ that supported the addition of ρ to H_ρ, we have:

$$\sum_{\rho} \mathsf{value}_{H_\rho}(\rho) = \sum_{(v,C) \in V \times \mathcal{C}_h} (\text{number of paths added to } H \text{ that } (v, C) \text{ supports}) \tag{4.4}$$

where the first summation is over all the paths ρ that got added to H.

Since the paths added to H have an additive stretch of at most $4k$, the distance between a node v and a cluster C, the first time (v, C) supports a path is at most $d_G(v, C) + 4k + 2$. So the distance between a cluster and a node can be decreased at most $4k + 3$ times during the path buying phase. Thus a cluster-node pair supports at most $4k + 3$ paths added to H. The total number of clusters is at most n/h and hence the total number of cluster-node pairs is at most n^2/h. So the sum of values of paths added to H can be at most $(4k + 3) \cdot n^2/h$. Hence the total number of edges added to H can be bounded as follows:

$$\sum_{\rho} \mathsf{cost}_{H_\rho}(\rho) \leq 6 \cdot \left(\tfrac{\log n}{D}\right)^{k/(k+1)} \cdot \sum_{\rho} \mathsf{value}_{H_\rho}(\rho) \tag{4.5}$$

$$= 6 \cdot \left(\tfrac{\log n}{D}\right)^{k/(k+1)} \cdot (4k + 3) \cdot \tfrac{n^2}{h} \tag{4.6}$$

We can easily see by substitution that this is $O(nh)$. Therefore the number of edges in H is $O(nh)$, which is also $\tilde{O}(n^{3/2}/D^{k/(2k+2)})$. □

Thus, we have proved Theorem 1.0.3. An interesting Corollary of the theorem follows by substituting $k = \lfloor \log n \rfloor$.

Corollary 4.2.5. *There is a polynomial time algorithm that takes a graph $G = (V, E)$ on n nodes and an integer $D \in [1, n]$ as its inputs and computes an $\tilde{O}(n \cdot \sqrt{n/D})$-sized $(1, 4 \log n)$ D-spanner of G.*

Chapter 5

Conclusion

The main theme of the thesis is an investigation on algorithms for computing sparse pairwise spanners of undirected unweighted graphs.

As part of our study, we obtain constructions for sparse small stretch pairwise spanners of a graph $G = (V, E)$. We consider the cases when the set of pairs $\mathcal{P}$ is an arbitrary subset of $V \times V$ and when the set $\mathcal{P}$ is of the form $S \times V$ for an arbitrary subset $S \subseteq V$. In both these cases, we obtain a $(1, 2)$ $\mathcal{P}$-spanner. Taking cue from these results, we also show the first deterministic algorithm to construct a $(1, 4)$ all pairs spanner with $O(n^{1.4} \log^{0.2} n)$ edges.

Another direction we have explored is that of computing sparse D-spanners. The main result is a tradeoff between the size and stretch for D-spanners.

We now list a few open problems as well as directions for further study.

Problem 1: Sparser Pairwise Spanners. A natural and important open problem in the context of Chapter 3 would be to see if we can get sparser pairwise spanners with the same stretch as ours in the case when the pairs are arbitrary. More specifically, we can raise the following question.

- Is it possible to compute $(1, 2)$ $\mathcal{P}$-spanners with $O(n|\mathcal{P}|^{1/4})$ edges when $\mathcal{P} \subseteq V \times V$ is arbitrary ?

If this can be answered in the affirmative, many of the pairwise and all pairs spanners we know will follow as corollaries of this. For arbitrary $\mathcal{P} \subseteq V \times V$, the only known $\tilde{O}(n|\mathcal{P}|^{1/4})$-sized $\mathcal{P}$-spanner has an additive stretch of $4 \log n$. We have described this result due to Cygan et al. [CGK13] in Section 2.2.

Note that we already show an $\tilde{O}(n|\mathcal{P}|^{1/4})$-sized $(1, 2)$ $\mathcal{P}$-spanner when $\mathcal{P} = S \times V$ for $S \subseteq V$ in Section 3.2. But the bound we have obtained on the number of edges of a $(1, 2)$ $\mathcal{P}$-spanner is $\tilde{O}(n|\mathcal{P}|^{1/3})$ for arbitrary $\mathcal{P}$.

Problem 2: Sparse $S_1 \times S_2$-spanners. A problem that we propose is whether it is possible to compute sparse $\mathcal{P}$-spanners of a graph $G = (V, E)$ when $\mathcal{P}$ is of the form $S_1 \times S_2$, where S_1 and S_2 are two disjoint subsets of V.

The problem can be motivated in the following way. S_1 denotes a set of sources and S_2, a set of destinations. The objective is to find if there are *nearly shortest* paths from all sources to all destinations such that the union of all such paths is sparse.

It might be possible to obtain an $\tilde{O}(n|\mathcal{P}|^{1/4})$-sized $(1, 2)$ $\mathcal{P}$-spanner when $\mathcal{P} = S_1 \times S_2$ for $S_1, S_2 \subseteq V$, since the pairs in this case have more structure than in the case when $\mathcal{P}$ is arbitrary.

Problem 3: Better Algorithms for Pairwise Spanners. We remark that our results are more combinatorial rather than algorithmic, in the sense that we have not made efforts to optimize the running time of our algorithms. It would be an interesting pursuit to figure out ways to improve the performance of the algorithms for pairwise spanners. Woodruff [Woo10] has made significant progress in this direction in the context of purely additive all-pairs spanners. They show a nearly quadratic time algorithm for constructing a $(1, 6)$ all pairs spanner of size $\tilde{O}(n^{4/3})$. Our current goal is to adapt the ideas used therein to the case of pairwise spanners.

Problem 4: $o(n^{4/3})$-sized purely additive spanners. We conclude the thesis with the most important open problem in the area of additive spanners which is whether it is possible to construct an all pairs additive spanner with stretch that is sub-polynomial in the number of nodes and size $o(n^{4/3})$. The currently known lower bounds [Woo06] do not rule out the possibility of existence of such spanners. We do not believe that it is possible to get $o(n^{4/3})$-sized purely additive spanners using the present techniques.

The study of additive spanners constitute a fundamental problem in the area of Graph Algorithms, mainly because of their wide applicability in both theory as well as practice. There is a lot of work that needs to be done in this area. We believe that making significant progress in any of the problems mentioned above, will be an important contribution to the area.

Bibliography

[ACIM99] 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.

[AP92] B. Awerbuch and D. Peleg. Routing with polynomial communication-space tradeoff. SIAM Journal on Discrete Math., 5(2):151–162, 1992.

[BCE05] B. Bollobás, D. Coppersmith, and M. Elkin. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Mathematics*, 19(4):1029–1055, 2005.

[BK10] S. Baswana and T. Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. SIAM Journal on Computing, 39(7):2865–2896, 2010.

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (alpha, beta)-spanners and purely additive spanners. In *SODA*, pages 672–681, 2005.

[BS04] S. Baswana and S. Sen. Approximate distance oracles for unweighted graphs in $o(n^2 \log n)$ time. In Proc. 15th ACM-SIAM Symposium on Discrete Algorithms, SODA '04, pages 271–280, 2004.

[BS06] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2006.

[CE05] Don Coppersmith and Michael Elkin. Sparse source-wise and pair-wise distance preservers. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 660–669, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[CGK13] M. Cygan, F. Grandoni, and T. Kavitha. On pairwise spanners. In *30th International Symposium on Theoretical Computer Science(STACS)*, 2013.

[Che13] S. Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 498–512. Society for Industrial and Applied Mathematics, 2013.

[Cow01] L. J. Cowen. Compact routing with minimum stretch. Journal of Algorithms, 28:170–183, 2001.

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

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

[Elk05] M. Elkin. Computing almost shortest paths. ACM Transactions on Algorithms, 1(2):283–323, 2005.

[EP04] M. Elkin and D. Peleg. $(1+\epsilon,\beta)$-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004.

[Erd64] P. Erdős. Extremal problems in graph theory. *Theory of graphs and its applications*, pages 29–36, 1964.

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

[KV13] Telikepalli Kavitha and Nithin M. Varma. Small stretch pairwise spanners. In *ICALP (1)*, pages 601–612, 2013.

[PR10] M. Patrascu and L. Roditty. Distance oracles beyond the thorup-zwick bound. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 815–823. IEEE, 2010.

[PS89] D. Peleg and A.A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989.

[PU89] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. SIAM Journal on Computing, 18:740–747, 1989.

[RTZ05] L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. *Automata, languages and programming*, pages 102–102, 2005.

[RZ04] L. Roditty and U. Zwick. On dynamic shortest paths problems. In Proc. 12th Annual European Symposium on Algorithms (ESA), pages 580–591, 2004.

[TZ05] M. Thorup and U. Zwick. Approximate distance oracles. Journal of the ACM, 52(1):1–24, 2005.

[Woo06] D.P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 389–398. IEEE, 2006.

[Woo10] David P. Woodruff. Additive spanners in nearly quadratic time. In *ICALP (1)*, pages 463–474, 2010.