Almost Shortest Paths with Near-Additive Error in Weighted Graphs

Michael Elkin ✉
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Yuval Gitlitz ✉
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Ofer Neiman ✉
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

Let $G = (V, E, w)$ be a weighted undirected graph with n vertices and m edges, and fix a set of s sources $S \subseteq V$. We study the problem of computing *almost shortest paths* (ASP) for all pairs in $S \times V$ in both classical centralized and parallel (PRAM) models of computation. Consider the regime of multiplicative approximation of $1 + \varepsilon$, for an arbitrarily small constant $\varepsilon > 0$ (henceforth $(1 + \varepsilon)$-ASP for $S \times V$). In this regime existing centralized algorithms require $\Omega(\min\{|E|s, n^\omega\})$ time, where $\omega < 2.372$ is the matrix multiplication exponent. Existing PRAM algorithms with polylogarithmic depth (aka time) require work $\Omega(\min\{|E|s, n^\omega\})$.

In a bold attempt to achieve centralized time close to the lower bound of $m + ns$, Cohen [10] devised an algorithm which, in addition to the multiplicative stretch of $1 + \varepsilon$, allows also *additive error* of $\beta \cdot W_{\max}$, where $W_{\max}$ is the maximum edge weight in G (assuming that the minimum edge weight is 1), and $\beta = (\log n)^{O(\frac{\log 1/\rho}{\rho})}$ is polylogarithmic in n. It also depends on the (possibly) arbitrarily small parameter $\rho > 0$ that determines the running time $O((m + ns)n^\rho)$ of the algorithm.

The tradeoff of [10] was improved in [15], whose algorithm has similar approximation guarantee and running time, but its β is $(1/\rho)^{O(\frac{\log 1/\rho}{\rho})}$. However, the latter algorithm produces distance estimates rather than actual approximate shortest paths. Also, the additive terms in [10, 15] depend linearly on a possibly quite large *global* maximum edge weight $W_{\max}$.

In the current paper we significantly improve this state of affairs. Our centralized algorithm has running time $O((m + ns)n^\rho)$, and its PRAM counterpart has polylogarithmic depth and work $O((m + ns)n^\rho)$, for an arbitrarily small constant $\rho > 0$. For a pair $(s, v) \in S \times V$, it provides a path of length $\hat{d}(s, v)$ that satisfies $\hat{d}(s, v) \leq (1 + \varepsilon)d_G(s, v) + \beta \cdot W(s, v)$, where $W(s, v)$ is the weight of the heaviest edge on some shortest $s - v$ path. Hence our additive term depends linearly on a *local* maximum edge weight, as opposed to the *global* maximum edge weight in [10, 15]. Finally, our $\beta = (1/\rho)^{O(1/\rho)}$, i.e., it is significantly smaller than in [10, 15].

We also extend a centralized algorithm of Dor et al. [14]. For a parameter $\kappa = 1, 2, \ldots$, this algorithm provides for *unweighted* graphs a purely additive approximation of $2(\kappa - 1)$ for *all pairs shortest paths* (APASP) in time $\tilde{O}(n^{2+1/\kappa})$. Within the same running time, our algorithm for *weighted* graphs provides a purely additive error of $2(\kappa-1)W(u, v)$, for every vertex pair $(u, v) \in \binom{V}{2}$, with $W(u, v)$ defined as above.

On the way to these results we devise a suite of novel constructions of spanners, emulators and hopsets.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases spanners, hopset, shortest paths, PRAM, distance oracles

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.23

Related Version *Full Version*: https://arxiv.org/abs/1907.11422

Funding *Michael Elkin*: ISF grant 2344/19
Yuval Gitlitz: Partially supported by the Lynn and William Frankel Center for Computer Sciences and ISF grant 970/21.
Ofer Neiman: ISF grant 970/21

1 Introduction

We study the problem of computing *almost shortest paths* from a set $S \subseteq V$ of designated vertices, called *sources*, to all other vertices in an n-vertex m-edge weighted undirected graph $G = (V, E, w)$, with non-negative edge weights. We aim at approximation guarantee of the form $(1+\varepsilon, \beta \cdot W)$, meaning that for every pair $(s, v) \in S \times V$, our algorithm will return a path of length $\hat{d}(s, v)$ (called *distance estimate*) that satisfies

$$d_G(s, v) \leq \hat{d}(s, v) \leq (1+\varepsilon) d_G(s, v) + \beta \cdot W(s, v) \ , \tag{1}$$

where $W(s, v)$ is the weight of the heaviest edge on some shortest $s - v$ path in G (If there are multiple shortest paths, pick the one with minimal $W(s, v)$).

Here $\varepsilon > 0$ is an arbitrarily small positive constant, and β is typically a large constant that depends on ε, and on the running time of the algorithm.[1] We call this problem $(1+\varepsilon, \beta \cdot W)$-*ASP for* $S \times V$.

The problem of computing approximate shortest paths is one of the most central, fundamental and well-researched problems in Graph Algorithms. We study this problem in both the classical centralized and in the parallel (PRAM CRCW) models of computation. Next, we overview the main previous results for this problem in these two settings, and describe our new results. We start with the centralized model, and then turn our attention to the PRAM model.

1.1 Centralized Setting

The classical algorithm of Dijkstra for the exact *single-source shortest path* (SSSP) problem provides running time of $O(m + n \log n)$ [24]. Thorup [42] devised an algorithm with running time $O(m + n \log \log n)$ for the case when all edge weights are integer. Using this algorithm separately from each of the $s = |S|$ sources results in running time of at least $(m \cdot s)$.

On the opposite end of the spectrum, one can compute $(1+\varepsilon)$-APASP (All Pair Approximate Shortest Paths) using matrix multiplication in $\tilde{O}(n^\omega)$ time [26, 4, 45], where $\omega < 2.372$ is the matrix multiplication exponent [13, 44, 27]. There are also a number of combinatorial (i.e., not exploiting fast matrix multiplication) APASP algorithms. In particular, Cohen and Zwick [11] showed that 3-APASP can be computed in $\tilde{O}(n^2)$ time. (They also provided a few additional algorithms with approximation ratio between 2 and 3 and running time greater than n^2.) Baswana and Kavitha [6] improved their approximation guarantee to $(2, W)$ (with the same running time of $\tilde{O}(n^2)$), and with W defined as in (1).

Finally, Cohen [10] devised an algorithm that for an arbitrarily small parameter $\rho > 0$, solves $(1+\varepsilon, \beta \cdot W_{\max})$-ASP for $S \times V$ in $\tilde{O}((m + s \cdot n) \cdot n^\rho)$ time[2], with $\beta = \beta_{Coh} = (\log n)^{O(\frac{\log(1/\rho)}{\rho})}$. Here $W_{\max}$ is the maximum edge weight in the entire graph (assuming the minimum edge weight is 1).

This result was improved by Elkin [15]. The algorithm of Elkin [15] also solves $(1+\varepsilon, \beta \cdot W_{\max})$-ASP for $S \times V$ in $\tilde{O}((m + sn)n^\rho)$ time, with $\beta = \beta_{Elk} = (1/\rho)^{O(\frac{\log(1/\rho)}{\rho})}$. This algorithm reports distance estimates, rather than actual paths.[3]

---

[1] In the introduction we will mostly suppress the dependence on ε. It can however be found in the technical part of the paper.

[2] The running time of the algorithm of [10], as well as of the algorithm of [15] and of our algorithm, is actually slightly better than $O((m + sn)n^\rho)$. Specifically, it is roughly $O(mn^\rho + sn^{1+1/2^{1/\rho}})$. We use the simpler expression of $O((m + sn)n^\rho)$ to simplify presentation. More precise and general bounds can be found in the technical part of the paper.

[3] There is a variant of the algorithm of [15] which reports actual paths, but requires time $\tilde{O}((m + sn)n^\rho \cdot W_{\max})$. This running time is typically prohibitively large as it depends linearly on $W_{\max}$.

The running time of [10, 15] is close (up to n^ρ, for an arbitrarily small constant $\rho > 0$) to the lower bound of $\Omega(m + n \cdot s)$. This is unlike other aforementioned algorithms, whose running time is much larger, i.e., $\Omega(\min\{m \cdot s, n^2\})$. However, the algorithms of [10, 15] suffer from a number of drawbacks. First, their additive term is linear in the maximum edge weight $W_{\max}$. Second, the coefficient β in them is quite large, even for a relatively large values of the parameter ρ. Third, as was mentioned above, the algorithm of [15] returns distance estimates, as opposed to actual paths that implement these estimates, and in the algorithm of [10] the coefficient β of the additive term is super-constant (specifically, polylogarithmic in n).

In the current paper we address these issues, and devise an algorithm for $(1+\varepsilon, \beta \cdot W)$-ASP for $S \times V$ with running time $O((m + sn)n^\rho)$, for an arbitrarily small parameter $\rho > 0$, with $\beta = (1/\rho)^{O(1/\rho)}$. This algorithm does report paths, rather than just distance estimates. Note also that the additive term grows linearly with the *local* maximum edge weight, i.e., with the weight of heaviest edge on each particular source-destination shortest path, as opposed to the *global* maximum edge weight $W_{\max}$. Finally, its coefficient β is significantly smaller than $\beta_{Elk} = (1/\rho)^{O(\frac{\log(1/\rho)}{\rho})}$, though it is admittedly still quite large[4]. (The coefficient β_{Coh} of [10] depends polylogarithmically on n, while the coefficient β in [15] and here are independent of n.)

We also extend an algorithm of Dor et al. [14] to weighted graphs. Specifically, the algorithm of [14] works for *unweighted* undirected graphs. For any parameter $\kappa = 1, 2, \ldots$, it provides an additive $2(\kappa - 1)$-APASP in $\tilde{O}(n^{2+1/\kappa})$ time. Our extension applies to *weighted* undirected graphs. It computes additive $2(\kappa - 1)W$-approximation for all pairs shortest paths within the same time $\tilde{O}(n^{2+1/\kappa})$, i.e., for any vertex pair $u, v \in V$, it produces a path of length at most $d_G(u, v) + 2(\kappa - 1) \cdot W(u, v)$, where $W(u, v)$ is as in (1).

Note that the linear dependence of additive error on W is unavoidable, as an algorithm with stretch $(1 + \varepsilon, o(W))$ can be translated into an algorithm with the same running time and with a purely multiplicative stretch of $1 + \varepsilon$.

1.2 Parallel Setting

In the PRAM model, multiple processors are connected to a single memory block, and the operations are performed in parallel by these processors. We will mostly be concerned with the Concurrent Read Concurrent Write (CRCW) PRAM model, that allows multiple processors to access any memory cell at any given round. The running time is measured by the number of rounds, and the work by the number of processors multiplied by the number of rounds.

Early algorithms for these problems [43, 31, 38, 39] require $\Omega(\sqrt{n})$ parallel time. Algorithms of [26, 4, 45], that were discussed in Section 1.1, can also be applied in PRAM. They provide $(1 + \varepsilon)$-APASP in polylogarithmic time and $\tilde{O}(n^\omega)$ work. The algorithm of Cohen [10], for a parameter $\rho > 0$, solves $(1 + \varepsilon)$-ASP for $S \times V$ in polylogarithmic time $(\log n)^{O(\frac{\log(1/\rho)}{\rho})}$ and work $\tilde{O}((m + n^{1+\rho})s + m \cdot n^\rho)$.

This tradeoff was then improved in [18, 19], where the running time is $(\log n)^{O(1/\rho)}$, and the work is the same as in [10]. Further spectacular progress was recently achieved by [33, 5], who devised $(1 + \varepsilon)$-SSSP algorithms with time $(\log n)^{O(1)}$ and work $\tilde{O}(m)$. Nevertheless, for $(1 + \varepsilon)$-ASP problem from the set $S \subseteq V$ of sources, one needs to run these algorithms in parallel from all the s sources. As a result, their work complexity becomes $\tilde{\Theta}(ms)$.

[4] We are able to further decrease β to $2^{O(1/\rho)}$, at the expense of increasing the multiplicative stretch from $1 + \varepsilon$ to $3 + \varepsilon$.

To summarize, all existing solutions for the problem with polylogarithmic time have work complexity $\Omega(\min\{m \cdot s, n^\omega\})$. We devise the first algorithm with polylogarithmic time $(\log n)^{O(1/\rho)}$ and work complexity $\tilde{O}((m+ns)n^\rho)$, for an arbitrarily small constant $\rho > 0$. In other words, our work complexity is within n^ρ, for an arbitrarily small constant $\rho > 0$, close to the lower bound of $\Omega(m + ns)$. On the other hand, unlike algorithms of [45, 10, 18, 19, 33, 5], whose approximation guarantee is a purely multiplicative $1+\varepsilon$, for an arbitrarily small $\varepsilon > 0$, our approximation guarantee is $(1+\varepsilon, \beta \cdot W)$, with $\beta = (1/\rho)^{O(1/\rho)}$, and W as in (1).

Moreover, our result can, in fact, be viewed as a *PRAM distance oracle*. Specifically, following the preprocessing that requires time $(\log n)^{O(1/\rho)}$ and work $\tilde{O}(mn^\rho)$, our algorithm stores a compact data structure of size $\tilde{O}(n^{1+\rho})$. Given a query vertex s, this data structure provides distance estimates $\hat{d}(s,v)$ for all $v \in V$, which satisfies (1) in *constant time* and using work $\tilde{O}(n^{1+\rho})$, where $\beta = (1/\rho)^{O(1/\rho)}$. Note that the distance oracle has size arbitrarily close to linear in n, its preprocessing time is polylogarithmic and preprocessing work is arbitrarily close to linear in m, and the query time is constant and the query work is arbitrarily close to linear in n. (Note that as the query provides distance estimates for n vertex pairs $\{(s,v) \mid v \in V\}$, its query work complexity must be $\Omega(n)$.)

1.3 Hopsets, Spanners and Emulators

From the technical viewpoint, these results are achieved via a combination of our new constructions of emulators, spanners and hopsets. For parameters $\alpha, \beta \geq 1$, we say that a graph $H = (V, E', w)$ is an (α, β)-*hopset* for a (weighted) graph $G = (V, E, w)$, if by adding E' to the graph, every pair $x, y \in V$ has an α-approximate shortest path consisting of at most β hops; Formally,

$$d_G(x,y) \leq d^{(\beta)}_{G \cup H}(x,y) \leq \alpha \cdot d_G(x,y) \ ,$$

where $d^{(\beta)}_{G\cup H}$ is the shortest path in $G \cup H$ containing at most β edges. The parameter α is called the *stretch*, and β is the *hopbound*.

We say that H is an (α, β)-*emulator* if for every $x, y \in V$,

$$d_G(x,y) \leq d_H(x,y) \leq \alpha \cdot d_G(x,y) + \beta \ ,$$

and H is a *spanner* if it is an emulator and a subgraph of G.

Hopsets and near-additive spanners are fundamental combinatorial constructs, and play a major role in efficient approximation of shortest paths in various computational models. These objects have been extensively investigated in recent years [21, 16, 41, 23, 36, 37, 7, 17, 32, 10, 9, 35, 28, 34, 29, 25, 18, 2, 19, 30]. The main interest is to understand the triple tradeoff between the size of the hopset (respectively, spanner), to the stretch α, and to the hopbound (resp., additive stretch) β. For algorithmic applications, it is also crucial to bound the construction time of the hopset/spanner/emulator.

We show near-additive spanners for *weighted* graphs, where the additive stretch for the pair x, y may depend also on the largest edge weight on the corresponding shortest path from x to y, $W(x,y)$. For a parameter $0 < \rho < 1$, we devise an algorithm that constructs a $(1+\varepsilon, \beta \cdot W)$-spanner of size $O(n^{1+1/2^{1/\rho}} + n/\rho)$ with $\beta \leq \left(\frac{1/\rho}{\varepsilon}\right)^{O(1/\rho)}$. We also show how to analyze the construction so that it yields smaller additive stretch, while increasing the multiplicative one. Specifically, we get a $(c, \beta \cdot W)$-spanner of same size as above, for every constant $c > 3$, and with $\beta = \left(\frac{1}{\varepsilon}\right)^{O(1/\rho)}$. Our emulators have a somewhat improved β. All of our results admit near-linear time algorithms, i.e., their running time is $O(|E|n^\rho)$, for an arbitrarily small constant $\rho > 0$.

1.4 Technical Overview

We adapt the constructions of [41, 36, 17, 19, 30] of hopsets, spanners and emulators so that they are suitable for weighted graphs, and provide an improved additive stretch (or hopbound) β. The basic idea in all these constructions is to generate a random hierarchy of vertex sets $V = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_k = \emptyset$, where for each $0 \leq i < k-1$, each element in A_i is included in A_{i+1} with probability $\approx n^{-2^{i-k}}$ (one should think of $k = 1/\rho$). We also refer to vertices at A_i as vertices at level i. For each $v \in V$, the pivot $p_i(v)$ is the closest vertex in A_i to v. Then the set of edges H is created by connecting, for every $0 \leq i \leq k-1$, every vertex $v \in A_i \setminus A_{i+1}$ to its *bunch*: all other vertices in A_i that are closer to it than $p_{i+1}(v)$. One difference in our construction is that we also connect each vertex to all its (at most k) pivots. The main technical innovation in this work is the new analysis of this construction, yielding various hopsets, emulators and spanners that apply for *weighted* graphs, and admit improved parameters. (Previous constructions of spanners and emulators [21, 41, 36, 17, 30] applied only for unweighted graphs.)

The previous analysis of the stretch for some pair $x, y \in V$ goes roughly as follows. Divide the $x-y$ path into intervals, and try to connect these intervals using low stretch paths in H (and for hopsets, also some of the graph edges, but with few hops). Each interval can either have a low stretch path; or fail, in which case that interval admits a nearby pivot (of some level i). Then, consider two failed intervals (the leftmost and rightmost ones), and try to find an $x-y$ path via the pivots of these intervals.

We note that this partitioning of the $x-y$ path to equal size intervals (used in previous works), cannot work directly for weighted graphs, since it may not be possible to divide the path to intervals of equal (or even near-equal) size. We provide a subtle adaptation of this technique so that it can handle weights. In particular, we distinguish between short and long distances: The sufficiently long distances may suffer a partition to un-equal intervals, as the induced error is dominated by the multiplicative stretch of $1+\varepsilon$. For the short distances, we stop this partitioning when it becomes too "expensive", and resort to an argument similar to the one in [40], which has large multiplicative stretch (of roughly $2^{1/\rho}$). However, at the point where we stop, that stretch can be accounted for by the additive stretch $\beta \cdot W$.

An additional ingredient in our new analysis of H as an emulator/spanner for weighted graphs, given a pair $x, y \in V$, is to iteratively find a vertex z on the $x-y$ shortest path (sufficiently far from x) that admits in H a path with low multiplicative stretch from x. When there is no such z, we show that we can reach y with small additive stretch. This technique can be used to obtain the improved dependence of β on the parameter $k = 1/\rho$.

Recall that emulators are insufficient for reporting paths. In particular, the approach of [10, 15] was based on emulators, rather than on spanners, and it is not clear if these algorithms can be adapted to build spanners (for weighted graphs). On the other hand, with our approach we can convert our constructions of emulators into constructions of spanners. Specifically, to build a spanner, we must use graph edges. So in order to connect vertices $v \in A_i \setminus A_{i+1}$ to the vertices in their bunch $B(v)$, we need to add paths of possibly many edges in the graph (rather than a single edge, as for emulators/hopsets). To do this we connect every vertex v to all vertices in its *half-bunch* (see Section 4 for its definition), as opposed to connecting it to all vertices in its full bunch. (The latter is the case in the construction of emulators.) This turns out to be sufficient to ensure that the union of all these paths does not contain too many edges. For this analysis we employ ideas of counting pairwise path intersections, developed in the context of distance preservers [12] and near-additive spanners and distance oracles for unweighted graphs [36, 22]. We simplify this approach, and extend it to weighted graphs.

1.5 Organization

In Section 2 we describe the construction of our emulators and hopsets. In Section 3 we analyze this construction, showing it provides emulators for weighted graphs. In Section 4 we describe the construction of our spanners for weighted graphs (proofs are deferred to Appendix D) and use them for our centralized algorithms for the ASP problem. In Section 5 we provide efficient implementations of our constructions, and use them in Section 6 for solving ASP in PRAM, and for PRAM distance oracles.

Our hopsets and emulators with improved β appear in Appendices A and B.

Our algorithm for pure additive APASP for weighted graphs is available in the full version of the paper.

Bibliographical note

Related results about (α,β)-spanners for *unweighted* graphs and (α,β)-hopsets with $\alpha \geq 3+\varepsilon$ were achieved independently of us and simultaneously by [8]. In another submission [20] , an $(1+\varepsilon)$-ASP algorithm for $S \times V$ with $|S| = n^r$, for some $0 < r \leq 1$. was devised. The running time of this algorithm in the centralized setting is $\tilde{O}(n^{\omega(r)})$, where $\omega(r)$ is the rectangular matrix multiplication exponent. ($n^{\omega(r)}$ is the time required to multiply an $n^r \times n$ matrix by an $n \times n$ one.) In PRAM setting that algorithm runs in polylogarithmic time and $\tilde{O}(n^{\omega(r)})$ work. For graphs $G = (V, E, w)$ and sets of sources $S \subseteq V$ of size s such that $m + ns = o(n^2)$, the algorithms that we devise in the current submission are more efficient than in [20].

Following our work, [3] devised algorithms for purely additive spanners (e.g., spanners with multiplicative stretch exactly 1) for weighted graphs. In their results, the additive term depends on the global maximal edge weight $W_{\max}$, and the size of their spanners is always $\Omega(n^{4/3})$ (the latter is unavoidable for purely additive spanners, due to a lower bound of [1]).

2 Construction

We use a similar construction to that of [41, 19, 30]. One difference is that every vertex connects to pivots in all levels. (Recall that the main difference is in the analysis.) Let $G = (V, E, w)$ be a weighted graph with n vertices, and fix an integer parameter $k \geq 1$. Let $\nu = 1/(2^k - 1)$. Let $A_0 \dots A_k$ be sets of vertices such that $A_0 = V$, $A_k = \emptyset$, and for $0 \leq i \leq k-2$, A_{i+1} is created by sampling independently every element from A_i with probability $q_i = n^{-2^i\nu} \cdot 2^{-2^i-1}$. For every $0 \leq i \leq k-1$, the expected size of A_i is:

$$N_i = E[|A_i|] = n \prod_{j=0}^{i-1} q_j = n^{1-(2^i-1)\nu} \cdot 2^{-2^i-i+1}$$

For every $i \in [k-1]$, define the pivot $p_i(v)$ to be the closest vertex in A_i to v, breaking ties by lexicographic order. For every $u \in A_i \setminus A_{i+1}$ define the bunch (see Figure 1)

$$B(u) = \{v \in A_i | d_G(u,v) < d_G(u, A_{i+1})\} \cup \{p_j(u) | i < j < k\} \ . \tag{2}$$

That is, the bunch $B(u)$ contains all the vertices which are in A_i and closer to u than $p_{i+1}(u)$, and at most k pivots. We then define $H = \{(u,v) \ : \ u \in V, v \in B(u)\}$, where the weight of the edge (u,v) is set as the weight of the shortest path between u, v in G.

▶ **Lemma 1.** *The size of H is*

$$H = O(kn + n^{1+\nu})$$

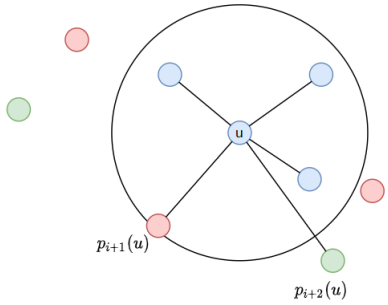

Figure 1 The bunch of u. Here vertices in $A_i \setminus A_{i+1}$ are colored in blue, in $A_{i+1} \setminus A_{i+2}$ are colored in red and in A_{i+2} are colored in green.

The proof of Lemma 1 is similar to previous works, we include it for completeness in appendix C.

3 Near-Additive Emulators for Weighted Graph

In this section we will show that H as defined in Section 2 is a $(1+\varepsilon, \beta \cdot W)$-emulator for weighted graphs. Note that the construction there does not depend on ε, and indeed, H will be an emulator for all values of $0 < \varepsilon < 1$ simultaneously (with $\beta = \beta(\varepsilon)$ depending on ε).

Let $G = (V, E)$ be a weighted graph with non-negative weights $w : E \to \mathbb{R}_+$, recall that for $x, y \in V$ we have $W(x, y) = \max\{w(e) \ : \ e \in P_{xy}\}$ (where P_{xy} is a shortest path from x to y in G). Let $k \geq 1$ and $\Delta > 3$ be given parameters (think of $\Delta = 3 + O(k/\varepsilon)$). We begin by proving two lemmas, handling long and short distances, respectively. The first lemma asserts that pairs which are sufficiently far apart admit either a low stretch path, or a nearby pivot of a higher level.

▶ **Lemma 2.** *Fix $\Delta > 3$. Let $0 \leq i < k$ and let $x, y \in V$ such that $d_G(x, y) \geq (3\Delta)^i W(x, y)$. Then at least one of the following holds:*

1. $d_H(x, y) \leq (1 + \frac{4i}{\Delta - 3}) d_G(x, y)$
2. $d_H(x, p_{i+1}(x)) \leq \frac{\Delta}{\Delta - 3} d_G(x, y)$

(For $i = k - 1$, the first item must hold since p_{i+1} doesn't exist).

Proof. Denote $W = W(x, y)$. The proof is by induction on i. For the base case $i = 0$, if $y \in B(x)$ then $d_H(x, y) = d_G(x, y)$ and the first item holds. Otherwise, if $x \in A_1$ then $d_G(x, p_1(x)) = 0$, so the second item holds. The last case is that $x \in A_0 \setminus A_1$ and $y \notin B(x)$, then by (2) we have $d_H(x, p_1(x)) = d_G(x, p_1(x)) \leq d_G(x, y) \leq \frac{\Delta}{\Delta - 3} d_G(x, y)$, thus the second item holds. Assume the lemma holds for i and prove for $i + 1$. Let $x, y \in V$ be a pair of vertices such that $d_G(x, y) \geq (3\Delta)^{i+1} W(x, y)$

Divide the shortest path between x and y into J segments $\{L_j = [u_j, u_{j+1}]\}_{j \in [J]}$ of length at least $(3\Delta)^i W$ and at most $d_G(x, y)/\Delta$. It can be done as follows: define $u_1 = x, j = 2$ and walk on the shortest path from x to y. Define u_j as the first vertex which $d_G(u_{j-1}, u_j) \geq (3\Delta)^i W$ or define $u_j = y$ if $d_G(u_{j-1}, y) < (3\Delta)^i W$. Increase j by 1 and repeat. Note that each segment has length at most $(3\Delta)^i W + W$. Finally, we join the last two segments. The length of the last segment is at most $(3\Delta)^i W + W + (3\Delta)^i W \leq 3^{i+1} \Delta^i W \leq d_G(x, y)/\Delta$. The length of any segment except the last is also at most $(3\Delta)^i W + W \leq d_G(x, y)/\Delta$.

Apply the induction hypothesis for every segment L_j with parameter i. If for all the segments the first item holds, then first item holds for x, y and $i+1$, since

$$\begin{aligned} d_H(x,y) &\leq \sum_{j\in J} d_H(u_j,u_{j+1}) \leq \sum_{j\in J}(1+\frac{4i}{\Delta-3})d_G(u_j,u_{j+1}) \\ &\leq (1+\frac{4i}{\Delta-3})d_G(x,y). \end{aligned}$$

Otherwise, for at least one segment the second item holds. Let L_l (resp., L_{r-1}) be the leftmost (resp., rightmost) segment for which the second item holds. We have that the second item holds for the pair u_l, u_{l+1}, and by symmetry of the first item, the second item also holds for the pair u_r, u_{r-1} with parameter i. Hence

$$\begin{aligned} d_H(u_r,p_{i+1}(u_r)) &\leq \frac{\Delta}{\Delta-3}d_G(u_{r-1},u_r) \leq \frac{d_G(x,y)}{\Delta-3}, \\ d_H(u_l,p_{i+1}(u_l)) &\leq \frac{\Delta}{\Delta-3}d_G(u_l,u_{l+1}) \leq \frac{d_G(x,y)}{\Delta-3}. \end{aligned} \tag{3}$$

Consider first the case that $p_{i+1}(u_r) \in B(p_{i+1}(u_l))$. In this case H contains the edge $\{p_{i+1}(u_l), p_{i+1}(u_r)\}$, so we have

$$d_H(p_{i+1}(u_l),p_{i+1}(u_r)) \leq d_G(p_{i+1}(u_l),u_l) + d_G(u_l,u_r) + d_G(u_r,p_{i+1}(u_r)). \tag{4}$$

By the triangle inequality,

$$\begin{aligned} &d_H(u_l,u_r) \qquad (5)\\ &\quad\leq d_H(u_l,p_{i+1}(u_l)) + d_H(p_{i+1}(u_l),p_{i+1}(u_r)) + d_H(p_{i+1}(u_r),u_r) \\ &\quad\overset{(4)}{\leq} 2d_H(u_l,p_{i+1}(u_l)) + d_G(u_l,u_r) + 2d_H(p_{i+1}(u_r),u_r) \\ &\quad\overset{(3)}{\leq} \frac{4d_G(x,y)}{\Delta-3} + d_G(u_l,u_r). \qquad (6) \end{aligned}$$

Thus, the distance between x and y in H,

$$\begin{aligned} d_H(x,y) &\leq \sum_{j\in[J]} d_H(u_j,u_{j+1}) \\ &\leq \sum_{j=1}^{l-1}\left(1+\frac{4i}{\Delta-3}\right)d_G(u_j,u_{j+1}) + d_H(u_l,u_r) + \sum_{j=r}^{J}\left(1+\frac{4i}{\Delta-3}\right)d_G(u_j,u_{j+1}) \\ &\leq \left(1+\frac{4i}{\Delta-3}\right)d_G(x,u_l) + d_H(u_l,u_r) + \left(1+\frac{4i}{\Delta-3}\right)d_G(u_r,y) \\ &\overset{(6)}{\leq} \left(1+\frac{4i}{\Delta-3}\right)d_G(x,u_l) + \frac{4d_G(x,y)}{\Delta-3} + d_G(u_l,u_r) + \left(1+\frac{4i}{\Delta-3}\right)d_G(u_r,y) \\ &\leq \left(1+\frac{4(i+1)}{\Delta-3}\right)d_G(x,y), \end{aligned}$$

therefore the first item holds.

The other case is that $p_{i+1}(u_r) \notin B(p_{i+1}(u_l))$. Then

$$d_H(p_{i+1}(u_l),p_{i+2}(p_{i+1}(u_l))) \leq d_G(p_{i+1}(u_l),p_{i+1}(u_r)) \ . \tag{7}$$

We can bound the distance $d_H(x, p_{i+2}(x))$ by

$$\begin{aligned}
&d_H(x, p_{i+2}(x)) \\
&\quad \le d_G(x, u_l) + d_G(u_l, p_{i+1}(u_l)) + d_G(p_{i+1}(u_l), p_{i+2}(p_{i+1}(u_l))) \\
&\quad \le d_G(x, u_l) + d_G(u_l, p_{i+1}(u_l)) + d_G(p_{i+1}(u_l), p_{i+1}(u_r)) \\
&\quad \overset{(4)}{\le} d_G(x, u_l) + d_G(u_l, p_{i+1}(u_l)) + d_G(u_l, p_{i+1}(u_l)) + d_G(u_l, u_r) + d_G(p_{i+1}(u_r), u_r) \\
&\quad \overset{(3)}{\le} d_G(x, y) + \frac{3d_G(x, y)}{\Delta - 3} = \frac{\Delta}{\Delta - 3} d_G(x, y).
\end{aligned}$$

Hence the second item holds. ◀

The previous lemma is useful for vertices which are very far from each other, since for $i = k - 1$ the first item must hold. For vertices which are close to each other, we will need the following lemma.

▶ **Lemma 3.** *Let $0 \le i < k$ and fix $x, y \in V$. Let $m = \max\{d_G(x, p_i(x)), d_G(y, p_i(y)), d_G(x, y)\}$. Then at least one of the following holds:*

1. $d_H(x, y) \le 5m$
2. $d_H(x, p_{i+1}(x)) \le 4m$ *and* $d_H(y, p_{i+1}(y)) \le 4m$

(For $i = k - 1$, the first item must hold since p_{i+1} doesn't exist).

Proof. If $p_i(y) \in B(p_i(x))$ the first item holds, since

$$\begin{aligned}
d_H(x, y) &\le d_H(x, p_i(x)) + d_H(p_i(x), p_i(y)) + d_H(p_i(y), y) \\
&\le d_G(x, p_i(x)) + d_G(p_i(x), x) + d_G(x, y) + d_G(y, p_i(y)) + d_G(p_i(y), y) \\
&\le 5m \ .
\end{aligned}$$

If $p_i(y) \notin B(p_i(x))$, then $d_G(p_i(x), p_{i+1}(p_i(x))) \le d_G(p_i(x), p_i(y))$, in this case the second item holds, as

$$\begin{aligned}
d_H(x, p_{i+1}(x)) &\le d_H(x, p_i(x)) + d_H(p_i(x), p_{i+1}(p_i(x))) \\
&\le d_G(x, p_i(x)) + d_G(p_i(x), x) + d_G(x, y) + d_G(y, p_i(y)) \le 4m \ .
\end{aligned}$$

The bound on $d_H(y, p_{i+1}(y))$ is symmetric. ◀

We are now ready to prove the following theorem.

▶ **Theorem 4.** *For any weighted graph $G = (V, E)$ on n vertices, and any integer $k > 1$, there exists H of size at most $O(kn + n^{1+1/(2^k-1)})$, which is a $(1 + \varepsilon, \beta \cdot W)$-emulator for any $0 < \varepsilon < 1$, where $\beta = O(\frac{k}{\varepsilon})^{k-1}$.*

Proof. Fix $\Delta = 3 + \frac{4(k-1)}{\varepsilon}$, $\beta = 10(3\Delta)^{k-1}$. Let $x, y \in V$, and $W = W(x, y)$. If $d_G(x, y) \ge (3\Delta)^{k-1}W$, we can apply Lemma 2 for x, y and $i = k - 1$. Since $p_k(x)$ does not exist, the first item must hold. Thus,

$$d_H(x, y) \le \left(1 + \frac{4(k-1)}{\Delta - 3}\right) d_G(x, y) = (1 + \varepsilon) d_G(x, y) \ .$$

Otherwise, take the integer $0 \le i < k - 1$ satisfying $(3\Delta)^i W \le d_G(x, y) < (3\Delta)^{i+1}W$ (note that there must be such an i, since $d_G(x, y) \ge W$). Apply Lemma 2 for x, y and i. If the first item holds, we will get $1 + \varepsilon$ stretch as before.

Otherwise, the second item holds, and we know that $d_G(x, p_{i+1}(x)) \leq \frac{\Delta}{\Delta-3} d_G(x,y) \leq 2d_G(x,y)$ (using that $k \geq 2$). By symmetry of x, y in the first item of Lemma 2, we have $d_G(y, p_{i+1}(y)) \leq 2d_G(x,y)$ as well. Set $j = i+1$ and apply Lemma 3 with x, y, j, noting that $m \leq 2d_G(x,y)$. If the first item holds, we found a path in H from x to y of length at most $5m \leq 10 d_G(x,y) \leq 10(3\Delta)^{k-1} W = \beta \cdot W$.

If the second item holds, we increase j by one and apply Lemma 3 again. We continue this procedure until the first item holds. The fact that the second item holds implies that the bound m (the maximal distance of x, y to the level j pivots) increases every iteration by a factor of at most 4. Since the first item must hold for $j = k-1$, the path we found is of length at most $10 \cdot 4^{k-i-2} d_G(x,y) \leq 10 \cdot 4^{k-i-2} \cdot (3\Delta)^{i+1} W$, which is maximized for $i = k-2$. Hence the additive stretch is at most $10 \cdot (3\Delta)^{k-1} W = O((9 + \frac{12(k-1)}{\varepsilon})^{k-1} W)$, as required. ◀

▶ Remark 5. We note that the analysis did not use the fact that the path from x to y is a shortest path. In particular, for every path P from x to y of length d, we can obtain a path in H of length at most $(1+\varepsilon) \cdot d + \beta \cdot W(P)$, where $W(P)$ is the largest edge weight in P.

4 Near-Additive Spanners for Weighted Graphs

In this section we devise our spanners for weighted graphs. We first describe the new construction, that differs from that of Section 2 in several aspects, which are required in order to keep the size of the spanner under control (and independent of $W_{\max}$). In particular, since we add paths, rather than edges, between each vertex to other vertices in its bunch, we need to ensure the size of H is small enough. To do that, we use half-bunches rather than bunches to define H (see (8) below), and show that there are few intersections between the aforementioned paths (see Lemma 17). One last ingredient is altering the sampling probabilities, so that the argument on intersection goes through. This approach refines and improves ideas from [36, 22].

Construction

Let $G = (V, E)$ be a weighted graph with n vertices, and fix an integer parameter $k \geq 3$. Let $\nu = \frac{1}{(4/3)^k - 1}$. Let $A_0 \ldots A_k$ be sets of vertices such that $A_0 = V$, $A_k = \emptyset$, and for $0 \leq i \leq k-2$, A_{i+1} is created by sampling every element from A_i with probability $q_i = n^{-4^i \nu / 3^{i+1}}$. For every $0 \leq i \leq k-1$, the expected size of A_i is:

$$N_i = E[|A_i|] = n \prod_{j=0}^{i-1} q_j = n^{1 - \frac{\nu}{3} \sum_{j=0}^{i-1} (4/3)^j} = n^{1 - ((4/3)^i - 1)\nu} .$$

For every $i \in [k-1]$, define the pivot $p_i(v)$ to be the closest vertex in A_i to v, breaking ties by lexicographic order. For every $u \in A_i \setminus A_{i+1}$ define the *half bunch*

$$B_{1/2}(u) = \{ v \in A_i \ : \ d_G(u,v) < d_G(u, A_{i+1})/2 \} . \tag{8}$$

Let $H = \{ P_{uv} \ : \ u \in V, v \in B_{1/2}(u) \cup \{ p_j(u) | i < j < k \} \}$, where P_{uv} is the shortest path between u, v in G (if there is more than one, break ties consistently, by vertex id, say).

▶ **Theorem 6.** *For any weighted graph $G = (V, E)$ on n vertices, and any integer $k > 2$, there exists H of size at most $O(kn + n^{1+1/((4/3)^k - 1)})$, which is a $(1+\varepsilon, \beta \cdot W)$-spanner for any $0 < \varepsilon < 1$, with $\beta = O(k/\varepsilon)^{k-1}$.*

▶ **Theorem 7.** *For any weighted graph $G = (V, E)$ on n vertices, and any integer $k > 2$, there exists H of size at most $O(kn + n^{1+1/((4/3)^k-1)})$, which is a $(3+\varepsilon, \beta \cdot W)$-spanner for any $\varepsilon > 0$ with $\beta = O(1 + 1/\varepsilon)^{k-1}$.*

Full proofs for both theorems are given in appendix D.

5 Efficient Implementation

Since we use very similar constructions to the ones in [19], we can use their efficient implementations (connecting to all pivots, which is the difference between constructions, can be done efficiently in their framework as well). We consider here the standard model of computation, and the PRAM (CRCW) model. Given a parameter $1/k < \rho < 1/2$, we will want poly-logarithmic parallel time and $\tilde{O}(|E| \cdot n^{\rho})$ work / centralized time. This is achieved by adding additional $\lceil 1/\rho \rceil$ sets A_i, that are sampled with uniform probability $n^{-\rho}$, which in turn increases the exponent of β by an additive $1/\rho + 1$. (In the case of multiplicative stretch $1+\varepsilon$, it also increases the base of the exponent in β.)

We summarize the efficient implementation results for hopsets and emulators in the following theorem.

▶ **Theorem 8.** *For any weighted graph $G = (V, E)$ on n vertices, parameters $k > 2$ and $1/k < \rho < 1/2$, there is a randomized algorithm running in time $\tilde{O}(|E| \cdot n^{\rho})$, that w.h.p. computes H of size at most $O(kn + n^{1+1/(2^k-1)})$, such that for any $0 < \varepsilon < 1$ this H is:*

1. *A $(1+\varepsilon, \beta \cdot W)$-emulator with $\beta = O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$.*
2. *A $(3+\varepsilon, \beta \cdot W)$-emulator with $\beta = O(1/\varepsilon)^{k+1/\rho}$.*
3. *A $(3+\varepsilon, \beta)$-hopset with $\beta = O(1/\varepsilon)^{k+1/\rho}$.*

Given ε in advance, the algorithm can also be implemented in the PRAM (CRCW) model, in parallel time $\left(\frac{\log n}{\varepsilon}\right)^{O(k+1/\rho)}$ and work $\tilde{O}(|E| \cdot n^{\rho})$, while increasing the size of H by a factor of $O(\log^* n)$.

For spanners, recall that in Section 4 we have a somewhat different construction, and in the analysis we enforce a stricter requirement on the sampling probabilities q_i. To handle this, we start sampling with the uniform probability $n^{-\rho}$ only when $N_i \le n^{1-3\rho}$ (and not when $N_i \le n^{1-\rho}$ like before). Now the bound of Claim 18 still holds, as $N_i/q_i^3 \le n$ even for these latter sets. The "price" we pay for waiting until $N_i \le n^{1-3\rho}$ is that the work will now be $|E| \cdot n^{3\rho}$. Rescaling ρ by 3, we get the following.

▶ **Theorem 9.** *For any weighted graph $G = (V, E)$ on n vertices, parameters $k > 6$ and $1/k < \rho < 1/6$, there is a randomized algorithm running in time $\tilde{O}(|E| \cdot n^{\rho})$, that w.h.p. computes H of size at most $O(kn + n^{1+1/(2^k-1)})$, such that for any $0 < \varepsilon < 1$ this H is:*

1. *A $(1+\varepsilon, \beta \cdot W)$-spanner with $\beta = O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+3/\rho}$.*
2. *A $(3+\varepsilon, \beta \cdot W)$-spanner with $\beta = O(1/\varepsilon)^{k+3/\rho}$.*

Given ε in advance, the algorithm can also be implemented in the PRAM (CRCW) model, in parallel time $\left(\frac{\log n}{\varepsilon}\right)^{O(k+1/\rho)}$ and work $\tilde{O}(|E| \cdot n^{\rho})$, while increasing the size of H by a factor of $O(\log^* n)$.

6 Almost Shortest Paths in Weighted Graphs

Given a weighted graph $G = (V, E, w)$ with n vertices and a set $S \subseteq V$ of s sources, fix parameters $k > 6$, $0 < \varepsilon < 1$ and $0 < \rho < 1/6$. Here we show that our hopsets, emulators and spanners can be used for a $(1+\varepsilon, \beta \cdot W)$-approximate shortest paths for pairs in $S \times V$, in various settings.

For the standard centralized setting, we first compute a $(1+\varepsilon, \beta \cdot W)$-spanner H of size $O(kn + n^{1+1/(2^k-1)})$ with $\beta = O(\frac{k+3/\rho}{\varepsilon})^{k+3/\rho}$, in time $\tilde{O}(|E| \cdot n^{\rho})$ as in Theorem 9. Next, for every $u \in S$ run Dijkstra's shortest path algorithm in H, which takes time $O(s \cdot (|E(H)| + n \log n)) = \tilde{O}(s \cdot n^{1+1/(2^k-1)})$.

The total running time for computing $(1+\varepsilon, \beta \cdot W)$-approximate shortest path for all $S \times V$, is $\tilde{O}(|E| \cdot n^{\rho} + s \cdot n^{1+1/(2^k-1)})$. One may choose $\rho = 1/k$ and obtain the following.

▶ **Theorem 10.** *For any weighted graph $G = (V, E)$ on n vertices, a set $S \subseteq V$ of s sources, and parameters $k > 6$ and $0 < \varepsilon < 1$, there is a randomized algorithm running in time $\tilde{O}(|E| \cdot n^{1/k} + s \cdot n^{1+1/(2^k-1)})$, that computes $(1+\varepsilon, \beta \cdot W)$-approximate shortest paths for all pairs in $S \times V$, where $\beta = \left(\frac{k}{\varepsilon}\right)^{O(k)}$.*

We remark that there is a more general tradeoff by choosing ρ as a free parameter. In addition, if one desires improved additive stretch, simply use the $(3+\varepsilon, \beta \cdot W)$-spanner with $\beta = \varepsilon^{-O(k)}$.

6.1 PRAM Shortest Paths and Distance Oracles

Given a weighted graph $G = (V, E, w)$ with n vertices, fix parameters $k \geq 1$, $0 < \varepsilon < 1$ and $0 < \rho < 1/6$. The first step in both settings (computing approximate shortest paths and distance oracles) is the same. We construct a $(1+\varepsilon, \beta \cdot W)$-emulator G' of size $O(kn + n^{1+1/(2^k-1)}) \cdot \log^* n$ with $\beta = O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$, in parallel time $\left(\frac{\log n}{\varepsilon}\right)^{O(k+1/\rho)}$ and work $\tilde{O}(|E| \cdot n^{\rho})$ as in Theorem 8. Next, compute a $(1+\varepsilon, \beta)$-hopset H size $O(n^{1+1/(2^k-1)}) \cdot \log^* n$ for G' with the same $\beta = O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$ within the same parallel time and work [19].[5] We store both G' and H.

Approximate Shortest Paths

For the sake of simplicity we will choose $\rho = 1/k$. Given a set S of s sources, for each source $u \in S$ run β rounds of the Bellman-Ford algorithm in the graph $G' \cup H$ starting at u. In each round of Bellman-Ford, every vertex sends its neighbors the current distance estimate to u that it has, and they update their distance estimate if needed. Since $G' \cup H$ is a sparse graph with $\tilde{O}(n^{1+1/(2^k-1)})$ edges, with $\tilde{O}(n^{1+1/(2^k-1)})$ processors one can implement each iteration in PRAM (CRCW) in $O(2^k)$ time (see [19] for more details). As we have only β rounds, the total parallel time for all the Bellman-Ford rounds from all vertices in S is $(k/\varepsilon)^{O(k)}$, and the total work is $\tilde{O}(s \cdot n^{1+1/(2^k-1)})$.

As the error of the emulator is $(1+\varepsilon, \beta \cdot W)$, and the hopset has only multiplicative $1+\varepsilon$ stretch, the total error is only $(1+O(\varepsilon), O(\beta \cdot W))$. We thus have the following result.

[5] We remark that even though the emulator and hopset have exactly the same construction, we run the hopset algorithm on G' and not on G, thus we get a different set of edges.

▶ **Theorem 11.** *For any weighted graph $G = (V, E)$ on n vertices, a set $S \subseteq V$ of s sources, and parameters $k > 2$ and $0 < \varepsilon < 1$, there is a PRAM randomized algorithm running in $\left(\frac{\log n}{\varepsilon}\right)^{O(k)}$ parallel time and using $\tilde{O}(|E| \cdot n^{1/k} + s \cdot n^{1+1/(2^k-1)})$ work, that computes $(1+\varepsilon, \beta \cdot W)$-approximate shortest paths for all pairs in $S \times V$, where $\beta = \left(\frac{k}{\varepsilon}\right)^{O(k)}$.*

As above, we can get a more general tradeoff with the parameter ρ, and improve the additive stretch by using the hopsets and emulators from Sections A,B, albeit the multiplicative stretch will increase.

Distance Oracles

Recall that we store the emulator G' and a hopset H for G'. Whenever a query $u \in V$ arrives, we run β rounds of Bellman-Ford algorithm in the graph $G' \cup H$. As noted above, each round of Bellman-Ford can be implemented in PRAM (CRCW) in $O(2^k)$ time using $\tilde{O}(n^{1+1/(2^k-1)})$ processors. So the total parallel time for the query is $O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$. Rescaling ε, we get a $\left(1+\varepsilon, O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho} \cdot W\right)$-approximation. We conclude that the properties of the distance oracle we devise are:

- Has size $O(kn + n^{1+1/(2^k-1)}) \cdot \log^* n$.
- Given query $u \in V$, can report $(1+\varepsilon, \beta \cdot W)$-approximation to *all* distances in $\{u\} \times V$, with $\beta = O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$.
- Has query time $O\left(\frac{k+1/\rho}{\varepsilon}\right)^{k+1/\rho}$ and $\tilde{O}(n^{1+1/(2^k-1)})$ work.
- The preprocessing time is $\left(\frac{\log n}{\varepsilon}\right)^{O(k+1/\rho)}$ and work $\tilde{O}(|E| \cdot n^{\rho})$.

References

1. Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *J. ACM*, 64(4):28:1–28:20, 2017. doi:10.1145/3088511.
2. Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 568–576, 2017. doi:10.1137/1.9781611974782.36.
3. Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen Kobourov, and Richard Spence. Weighted additive spanners, 2020. arXiv:2002.07152.
4. Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997. doi:10.1006/jcss.1997.1388.
5. Alexandr Andoni, Clifford Stein, and Peilin Zhong. Parallel approximate undirected shortest paths via low hop emulators. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 322–335. ACM, 2020. doi:10.1145/3357713.3384321.
6. Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *FOCS*, pages 591–602, 2006. doi:10.1109/FOCS.2006.29.
7. Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (alpha, beta)-spanners. *ACM Transactions on Algorithms*, 7(1):5, 2010. doi:10.1145/1868237.1868242.

8 Uri Ben-Levy and Merav Parter. New (α, β) spanners and hopsets. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1695–1714. SIAM, 2020. doi:10.1137/1.9781611975994.104.

9 Aaron Bernstein. Fully dynamic (2 + epsilon) approximate all-pairs shortest paths with fast query and close to linear update time. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 693–702, 2009. doi:10.1109/FOCS.2009.16.

10 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000. doi:10.1145/331605.331610.

11 Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *J. Algorithms*, 38(2):335–353, 2001. doi:10.1006/jagm.2000.1117.

12 D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 660–669, 2005.

13 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.

14 D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29:1740–1759, 2000.

15 M. Elkin. Computing almost shortest paths. In *Proc. 20th ACM Symp. on Principles of Distributed Computing*, pages 53–62, 2001.

16 M. Elkin. An unconditional lower bound on the time-approximation tradeoff of the minimum spanning tree problem. In *Proc. of the 36th ACM Symp. on Theory of Comput. (STOC 2004)*, pages 331–340, 2004.

17 Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. *ACM Trans. Algorithms*, 15(1):4:1–4:29, 2019. doi:10.1145/3274651.

18 Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. *SIAM J. Comput.*, 48(4):1436–1480, 2019. doi:10.1137/18M1166791.

19 Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in RNC. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019.*, pages 333–341, 2019. doi:10.1145/3323165.3323177.

20 Michael Elkin and Ofer Neiman. Centralized, parallel, and distributed multi-source shortest paths via hopsets and rectangular matrix multiplication. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 27:1–27:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.27.

21 Michael Elkin and David Peleg. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631, 2004. doi:10.1137/S0097539701393384.

22 Michael Elkin and Seth Pettie. A linear-size logarithmic stretch path-reporting distance oracle for general graphs. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 805–821, 2015. doi:10.1137/1.9781611973730.55.

23 Michael Elkin and Jian Zhang. Efficient algorithms for constructing (1+epsilon, beta)-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006. doi:10.1007/s00446-005-0147-2.

24 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

25 Stephan Friedrichs and Christoph Lenzen. Parallel metric tree embedding based on an algebraic view on moore-bellman-ford. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '16, pages 455–466, New York, NY, USA, 2016. ACM. doi:10.1145/2935764.2935777.

26 Zvi Galil and Oded Margalit. All pairs shortest distances for graphs with small integer length edges. *Inf. Comput.*, 134(2):103–139, 1997. doi:10.1006/inco.1997.2620.

27 François Le Gall. Powers of tensors and fast matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303. ACM, 2014. doi:10.1145/2608628.2608664.

28 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 146–155, 2014. doi:10.1109/FOCS.2014.24.

29 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 489–498, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897638.

30 Shang-En Huang and Seth Pettie. Thorup-zwick emulators are universally optimal hopsets. *Inf. Process. Lett.*, 142:9–13, 2019. doi:10.1016/j.ipl.2018.10.001.

31 Philip N. Klein and Sairam Subramanian. A linear-processor polylog-time algorithm for shortest paths in planar graphs. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 259–270, 1993. doi:10.1109/SFCS.1993.366861.

32 Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *J. Algorithms*, 25(2):205–220, 1997. doi:10.1006/jagm.1997.0888.

33 Jason Li. Faster parallel algorithm for approximate shortest path. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 308–321. ACM, 2020. doi:10.1145/3357713.3384268.

34 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '15, pages 192–201, New York, NY, USA, 2015. ACM. doi:10.1145/2755573.2755574.

35 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 565–573, 2014. doi:10.1145/2591796.2591850.

36 Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms*, 6(1), 2009. doi:10.1145/1644015.1644022.

37 Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010. doi:10.1007/s00446-009-0091-7.

38 Hanmao Shi and Thomas H. Spencer. Time-work tradeoffs of the single-source shortest paths problem. *J. Algorithms*, 30(1):19–32, 1999. doi:10.1006/jagm.1998.0968.

39 Thomas H. Spencer. Time-work tradeoffs for parallel algorithms. *J. ACM*, 44(5):742–778, 1997. doi:10.1145/265910.265923.

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

41 M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *Proc. of Symp. on Discr. Algorithms*, pages 802–809, 2006.

42 Mikkel Thorup. Undirected single source shortest path in linear time. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 12–21. IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646088.

43 Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991. doi:10.1137/0220006.

44 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 887–898. ACM, 2012. doi:10.1145/2213977.2214056.

45 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

A A $(3+\varepsilon,\beta)$-Hopset

Here we show that the set H of Section 2 serves as a $(3+\varepsilon,\beta)$-hopset, for all $0<\varepsilon<12$ simultaneously, with $\beta = 2^{O(k\cdot\log(1/\varepsilon))}$.

Denote by $d_G^{(t)}(u,v)$ the length of the shortest path between u,v in G that contains at most t edges. The following lemma bounds the number of hops and the stretch of the constructed hopset:

▶ **Lemma 12.** *Fix any $0<\delta\leq 1/4$ and any $x,y\in V$. Then for every $0\leq i\leq k-1$, at least one of the following holds:*

1. $d_{G\cup H}^{(2\cdot(1/\delta)^i-1)}(x,y)\leq(3+\frac{12\delta}{1-3\delta})d_G(x,y)$
2. $d_{G\cup H}^{(1)}(x,p_{i+1}(x))\leq\frac{3}{1-3\delta}d_G(x,y)$

Proof. The proof is by induction on i. For the base case $i=0$, if $y\in B(x)$, then the edge (x,y) was added to the hopset and the first item holds. If not, it means that $d_G(x,y)\geq d_G(x,p_1(x))$. Because every vertex is connected by a direct edge to all its pivots, the second item holds (since the coefficient of the right hand side is between 3 and 12 for $0<\delta\leq 1/4$).

Assume the claim holds for i, and we will prove it holds for $i+1$. Partition the shortest path between x and y into $J\leq 1/\delta$ segments $\{L_j=[u_j,v_j]\}_{j\in[J]}$ each of length at most $\delta\cdot d_G(x,y)$, and at most $(1/\delta-1)$ edges $\{(v_j,u_{j+1})\}_{j\in[J]}$ between consecutive segments. We can use the following: setting $u_1=x$, and for each $j\in[J]$, set v_j as the vertex in the shortest path between u_j and y which is farthest from u_j, but still $d_G(u_j,v_j)\leq\delta\cdot d_G(x,y)$. If $v_j\neq y$, set u_{j+1} as the vertex which follows v_j in the shortest path between x and y. Otherwise set $u_{j+1}=y$. This partition satisfies our requirement $J\leq 1/\delta$ because for every $j\in[J-1]$, $d_G(u_j,u_{j+1})>\delta\cdot d_G(x,y)$ (otherwise, we could have chosen v_j as u_{j+1}).

Next, apply the induction hypothesis for all the pairs (u_j,v_j) with parameter i. If for all the pairs (u_j,v_j) the first item holds, we can show that the first item holds for (x,y) with parameter $i+1$. Consider the path from x to y which uses the guaranteed path in $G\cup H$ of the first item for all the pairs (u_j,v_j), and the edges (v_j,u_{j+1}). The number of hops in this path is bounded by $(1/\delta)\cdot(2(1/\delta)^i-1)+(1/\delta-1)\leq 2\cdot(1/\delta)^{i+1}-1$. The length of the path is bounded by (using the induction hypothesis on each pair):

$$\begin{aligned} d_{G\cup H}^{(2(1/\delta)^{i+1}-1)}(x,y) &\leq \sum_{j\in[J]}(d_{G\cup H}^{(2(1/\delta)^i-1)}(u_j,v_j)+d_G^{(1)}(v_j,u_{j+1})) \\ &\leq (3+\frac{12\delta}{1-3\delta})d_G(x,y)\ . \end{aligned}$$

Otherwise, there exist at least one segment for which the first item doesn't hold. Let $l\in[J]$ be the smallest index so that only the second item holds for the pair (u_l,v_l). By the induction hypothesis

$$d_{G\cup H}^{(1)}(u_l,p_{i+1}(u_l))\leq\frac{3}{1-3\delta}d_G(u_l,v_l)\leq\frac{3\delta}{1-3\delta}d_G(x,y).$$

Since we added the edges $(x, p_{i+1}(x))$, $(y, p_{i+1}(y))$ to the hopset H, by the triangle inequality,

$$\begin{aligned} d^{(1)}_{G\cup H}(x, p_{i+1}(x)) &\le d_G(x, u_l) + d_G(u_l, p_{i+1}(u_l)) \\ &\le d_G(x, u_l) + \frac{3\delta}{1-3\delta} d_G(x, y), \end{aligned} \tag{9}$$

$$\begin{aligned} d^{(1)}_{G\cup H}(y, p_{i+1}(y)) &\le d_G(y, u_l) + d_G(u_l, p_{i+1}(u_l)) \\ &\le d_G(y, u_l) + \frac{3\delta}{1-3\delta} d_G(x, y). \end{aligned} \tag{10}$$

If the edge $(p_{i+1}(x), p_{i+1}(y))$ exists in H, its length can be bounded by

$$\begin{aligned} &d^{(1)}_{G\cup H}(p_{i+1}(x), p_{i+1}(y)) \\ &\quad\le d^{(1)}_{G\cup H}(p_{i+1}(x), x) + d_G(x, y) + d^{(1)}_{G\cup H}(y, p_{i+1}(y)). \end{aligned} \tag{11}$$

Thus, the distance between x and y using 3 hops is

$$\begin{aligned} &d^{(3)}_{G\cup H}(x, y) \\ &\le d^{(1)}_{G\cup H}(x, p_{i+1}(x)) + d^{(1)}_{G\cup H}(p_{i+1}(x), p_{i+1}(y)) + d^{(1)}_{G\cup H}(p_{i+1}(y), y) \\ &\overset{(11)}{\le} 2d^{(1)}_{G\cup H}(x, p_{i+1}(x)) + d_G(x, y) + 2d^{(1)}_{G\cup H}(p_{i+1}(y), y) \\ &\overset{(9)+(10)}{\le} 2\Big(d_G(x, u_l) + \frac{3\delta}{1-3\delta} d_G(x, y)\Big) + d_G(x, y) + 2\Big(d_G(y, u_l) \\ &\qquad\qquad \frac{3\delta}{1-3\delta} d_G(x, y)\Big) \\ &\le (3 + \frac{12\delta}{1-3\delta}) d_G(x, y), \end{aligned}$$

therefore the first item holds.

If $(p_{i+1}(x), p_{i+1}(y)) \notin H$, then we know that $d^{(1)}_{G\cup H}(p_{i+1}(x), p_{i+2}(p_{i+1}(x))) \le d_G(p_{i+1}(x), p_{i+1}(y))$. We can bound the distance $d^{(1)}_{G\cup H}(x, p_{i+2}(x))$ using the triangle inequality:

$$\begin{aligned} &d^{(1)}_{G\cup H}(x, p_{i+2}(x)) \\ &\le d_G(x, p_{i+1}(x)) + d_G(p_{i+1}(x), p_{i+2}(p_{i+1}(x))) \\ &\overset{(11)}{\le} 2d_G(p_{i+1}(x), x) + d_G(x, y) + d_G(y, p_{i+1}(y)) \\ &\le 2(d_G(x, u_l) + \frac{3\delta}{1-3\delta} d_G(x, y)) + d_G(x, y) + d_G(y, u_l) + \frac{3\delta}{1-3\delta} d_G(x, y) \\ &\le 3d_G(x, y) + \frac{9\delta}{1-3\delta} d_G(x, y) = \frac{3}{1-3\delta} d_G(x, y). \end{aligned}$$

Thus the second item holds. ◀

We conclude by summarizing the main result of this section.

▶ **Theorem 13.** *For any weighted graph $G = (V, E)$ on n vertices, and any $k \ge 1$, there exists H of size at most $O(kn + n^{1+1/(2^k-1)})$, which is a $(3+\varepsilon, \beta)$-hopset for any $0 < \varepsilon \le 12$, with $\beta = 2(3 + 12/\varepsilon)^{k-1}$.*

Proof. Let $x, y \in V$. Apply lemma 12 for x, y with $\delta = \frac{\varepsilon}{12+3\varepsilon}$ and $i = k-1$. Since $A_k = \emptyset$, the first item must hold:

$$d^{(2(3+12/\varepsilon)^{k-1})}_{G\cup H}(x, y) \le (3+\varepsilon) d_G(x, y). \quad ◀$$

▶ Remark 14. Note that at its lowest, the hopbound is $O(4^k)$, achieved with stretch 15.

B A $(3+\varepsilon, \beta \cdot W)$-Emulator

In this section we show that the same H constructed in Section 2 can also serve as a $(3+\varepsilon, \beta \cdot W)$ emulator for weighted graphs, for all values of $0 < \varepsilon < 1$ simultaneously.

Let $G = (V, E)$ be a weighted graph and let $k \geq 1$ and $\Delta > 3$ be given parameters (think of $\Delta = 3 + O(1/\varepsilon)$). Fix a pair $x, y \in V$. Define $D_{-1} = 0$ and for any integer $i \geq 0$, let $D_i = W(x,y) \cdot \sum_{j=0}^{i} \Delta^j$. We can easily verify that

$$D_{i+1} = \Delta \cdot D_i + W(x,y). \tag{12}$$

▶ **Lemma 15.** *Let $0 \leq i \leq k$ and let $x, y \in V$ such that $d_G(x,y) \leq D_i$ and $d_H(x, p_i(x)) \leq \frac{2\Delta}{\Delta-3} D_{i-1}$. Define $m = \max\{\Delta D_{i-1}, d_G(x,y)\}$. Then at least one of the following holds:*

1. $d_H(x,y) \leq (3 + \frac{8}{\Delta-3})m$.
2. $d_H(x, p_{i+1}(x)) \leq \frac{2\Delta}{\Delta-3} D_i$.

Proof. The proof is by induction on i. For the base case $i = 0$, if $y \in B(x)$, the first item holds. Otherwise $d_H(x, p_1(x)) \leq d_G(x,y) \leq W(x,y) = D_0$, thus the second item holds.

Assume the claim holds for i and prove for $i+1$. By the triangle inequality:

$$d_H(y, p_{i+1}(y)) \leq d_G(y,x) + d_G(x, p_{i+1}(x)). \tag{13}$$

If $p_{i+1}(y) \in B(p_{i+1}(x))$, we have

$$d_H(p_{i+1}(x), p_{i+1}(y)) \leq d_G(p_{i+1}(x), x) + d_G(x,y) + d_G(y, p_{i+1}(y)). \tag{14}$$

Thus, the distance between x and y is

$$\begin{aligned}
&d_H(x,y) \\
&\quad \leq d_H(x, p_{i+1}(x)) + d_H(p_{i+1}(x), p_{i+1}(y)) + d_H(p_{i+1}(y), y) \\
&\quad \overset{(14)}{\leq} 2d_H(x, p_{i+1}(x)) + d_G(x,y) + 2d_H(p_{i+1}(y), y) \\
&\quad \overset{(13)}{\leq} 2d_H(x, p_{i+1}(x)) + d_G(x,y) + 2(d_G(y,x) + d_G(x, p_{i+1}(x))) \\
&\quad \leq 3d_G(x,y) + \frac{4 \cdot 2\Delta}{\Delta - 3} D_i \\
&\quad \leq \left(3 + \frac{8}{\Delta - 3}\right) m,
\end{aligned}$$

therefore the first item holds.

If $p_{i+1}(y) \notin B(p_{i+1}(x))$, then we know that

$$d_H(p_{i+1}(x), p_{i+2}(p_{i+1}(x))) \leq d_G(p_{i+1}(x), p_{i+1}(y)) \ . \tag{15}$$

We can bound the distance $d_H(x, p_{i+2}(x))$ as follows.

$$\begin{aligned}
&d_H(x, p_{i+2}(x)) \\
&\quad \leq d_G(x, p_{i+1}(x)) + d_G(p_{i+1}(x), p_{i+2}(p_{i+1}(x))) \\
&\quad \overset{(15)}{\leq} 2d_G(p_{i+1}(x), x) + d_G(x, y) + d_G(y, p_{i+1}(y)) \\
&\quad \overset{(13)}{\leq} 2d_H(x, p_{i+1}(x)) + d_G(x, y) + d_G(y, x) + d_G(x, p_{i+1}(x)) \\
&\quad \leq 2d_G(x, y) + \frac{3 \cdot 2\Delta}{\Delta - 3} D_i \\
&\quad \overset{(12)}{\leq} 2D_{i+1} + \frac{6}{\Delta - 3} D_{i+1} \\
&\quad = \frac{2\Delta}{\Delta - 3} D_{i+1}.
\end{aligned}$$

Hence the second item holds. ◀

We are now ready to state the result of this section.

▶ **Theorem 16.** *For any weighted graph $G = (V, E)$ on n vertices, and any $k \geq 1$, there exists H of size at most $O(kn + n^{1+1/(2^k-1)})$, which is a $(3 + \varepsilon, \beta \cdot W)$-emulator for any $\varepsilon > 0$ with $\beta = O(1 + 1/\varepsilon)^{k-1}$.*

Proof. Let $x, y \in V$. Recall that P_{xy} is the shortest path between x and y in G, and fix $\Delta = 3 + 8/\varepsilon$. Initialize $i = 0$. Let z be the farthest vertex from x in P_{xy} satisfying $d_G(x, z) \leq D_i$. Note the requirement $d_H(x, p_0(x)) \leq \frac{2\Delta}{\Delta-3} D_{-1} = 0$ holds since $p_0(x) = x$. Apply lemma 15 on x, z and i. If the second item holds, we increase i by one, update z to be the last vertex in $P(x, y)$ satisfying $d_G(x, z) \leq D_i$, and apply the lemma again for x, z and i (since the second item held for $i - 1$, we have that $d_H(x, p_i(x)) \leq \frac{2\Delta}{\Delta-3} D_{i-1}$ indeed holds).

Consider now the index i such that the first item holds (we must find such an index, since at $i = k - 1$ there is no pivot in level k). If it is the case that $d_G(x, y) \geq D_i$ then since $D_i - \Delta D_{i-1} = W(x, y)$, it must be that $d_G(x, z) \geq \Delta D_{i-1}$, as otherwise we could have taken a further away z (recall that every edge on this path has weight at most $W(x, y)$). Therefore $m = d_G(x, z)$ and we found a path in H from x to z with stretch at most $3 + \frac{8}{\Delta-3}$. Next we update $x = z$, $i = 0$ and repeat the same procedure all over again.

The last remaining case is that we found an index i such that the first item holds but $d_G(x, y) < D_i$. Note that in such a case it must be that $z = y$. The path in H we have from x to y is of length at most $(3 + \frac{8}{\Delta-3}) \cdot D_i = (3 + \varepsilon) \cdot D_i$. As $i \leq k - 1$ and $D_{k-1} \leq 2\Delta^{k-1} \cdot W(x, y)$, we have that

$$d_H(x, y) \leq 2(3 + \varepsilon) \cdot (3 + 8/\varepsilon)^{k-1} \cdot W(x, y) \ ,$$

which is our additive stretch β. ◀

C Full proof of Lemma 1

If we order the vertices in A_i by their distance to u, it is easy to see that the number of vertices which are in A_i and closer than $p_{i+1}(u)$ is bounded by a random variable sampled from a geometric distribution with parameter q_i. Hence $E[|B(u)|] \leq k + 1/q_i = k + n^{2^i \nu} 2^{2^i + 1}$. For $u \in A_{k-1}$, since $p_k(u)$ doesn't exist, $B(u)$ contains all the vertices in A_{k-1}. The number of vertices in A_{k-1} is a random variable sampled from binomial Distribution with parameters $(n, \prod_{j=0}^{k-2} q_j) = (n, n^{-(2^{k-1}-1)\nu} \cdot 2^{-2^{k-1}-k+2})$. Hence, the expected number of edges added by bunches of vertices in A_{k-1} is

$$
\begin{aligned}
E\left[\binom{|A_{k-1}|}{2}\right] &\le E[|A_{k-1}|^2] = E[|A_{k-1}|]^2 + Var(|A_{k-1}|) \\
&= n^2 \prod_{j=0}^{k-2} q_j^2 + n(1 - \prod_{j=0}^{k-2} q_j) \prod_{j=0}^{k-2} q_j \\
&\le n^{2-2(2^{k-1}-1)\nu} \cdot 2^{2(-2^{k-1}-k+2)} + n \cdot 2^{-2^{k-1}-k+2} \\
&\le (n^{1+\nu} + n) 2^{3-k}.
\end{aligned}
$$

Hence, the total expected number of edges in H is:

$$
\begin{aligned}
&\sum_{i=0}^{k-2} (N_i \cdot n^{2^i \nu} \cdot 2^{2^i+1}) + E[|A_{k-1}|^2] + kn \\
&= \sum_{i=0}^{k-2} (n^{1+\nu} \cdot 2^{-i+2}) + E[|A_{k-1}|^2] + kn \\
&= O(kn + n^{1+\nu})
\end{aligned}
$$

D Proofs of Theorems 6,7

D.1 Size analysis

Recall the construction of H described in Section 4. Define the bunch $B(u)$ as in (2). The following lemma will be useful to bound the size of the spanner H.

▶ **Lemma 17.** *Fix $0 \le i \le k-1$. Let $u, v, x, y \in A_i$ be such that $v \in B_{1/2}(u)$ and $y \in B_{1/2}(x)$, and $P_{uv} \cap P_{xy} \neq \emptyset$, then all four points are in $B(u)$, or all four are in $B(x)$.*

Proof. Assume w.l.o.g. that P_{uv} is not shorter than P_{xy}. Let $z \in V$ be a point in the intersection of the two shortest paths, then

$$
\begin{aligned}
d_G(u,x) &\le d_G(u,z) + d_G(z,x) \le d_G(u,v) + d_G(y,x) \\
&\le 2d_G(u,v) < d_G(u, A_{i+1}) \ ,
\end{aligned}
$$

so $x \in B(u)$. The calculation showing $y \in B(u)$ is essentially the same. ◀

Define the shortest path between two vertices consistently, s.t. each subpath is also a shortest path. Therefore two shortest paths can have at most one common subpath.

Fix $0 \le i \le k-2$, and consider the graph G_i containing all the shortest paths P_{uv} with $u \in A_i$ and $v \in B_{1/2}(u)$. We claim that the number of edges in G_i is at most $O(n + C_i)$, where C_i is the number of pairwise intersections between these shortest paths. This is because vertices participating in at most 1 path have degree at most 2, and each intersection increases the degree of one vertex by at most 2 (recall that shortest paths can meet at most once).

▷ Claim 18. $\mathbb{E}[|C_i|] \le O(n^{1+\nu})$.

Proof. By Lemma 17 each intersecting pair of paths P_{uv} and P_{xy} we have that all four points belong to the same bunch. Thus, each $u \in A_i$ can introduce at most $|B(u)|^3$ pairwise intersecting paths. Recall that $|B(u)|$ is a random variable distributed geometrically with parameter q_i, so

$$\mathbb{E}[|B(u)|^3] = \sum_{j=1}^{\infty} j^3 \cdot q_i \cdot (1-q_i)^{j-1}$$
$$\leq q_i \cdot \sum_{j=1}^{\infty} (1-q_i)^{j-1} \cdot j(j+1)(j+2) \leq \frac{6}{q_i^3} .$$

Thus the expected number of intersections at level i is at most

$$\mathbb{E}\left[\sum_{u \in A_i} |B(u)|^3\right] \leq O(N_i/q_i^3) = O(n^{1-((4/3)^i-1)\nu} \cdot (n^{4^i\nu/3^{i+1}})^3)$$
$$= O(n^{1+\nu}) .$$

It remains to bound path intersections in the last level $k-1$. Recall that

$$N_{k-1} = n^{1-((4/3)^{k-1}-1)\nu} = n^{1-((4/3)^{k-1}-1)/((4/3)^k-1)} \leq n^{(1+\nu)/4} .$$

Since the random choices for each point are independent, we have by Chernoff bound that $\Pr[N_{k-1} > 2n^{(1+\nu)/4}] \leq e^{-\Omega(n^{(1+\nu)/4})}$, so with very high probability the last set A_{k-1} contains $O(n^{(1+\nu)/4})$ points. It means that we have $O(\sqrt{n^{1+\nu}})$ paths connecting these points, and even if they all intersect, they can yield at most $O(n^{1+\nu})$ intersections. ◁

It remains to bound the number of edges to pivots. For every $v \in V$, $p_i(v)$ is the closest vertex to v in A_i breaking ties by id. Therefore all the vertices in $P_{v,p_i(v)}$ share the same pivot at level i. Thus we add at most one edge for each vertex at every level, and $O(nk)$ edges overall.

We conclude that the size of H is at most $O(k \cdot n^{1+\nu})$ (we can slightly change the probabilities by introducing a factor of $2^{-4^i/3^{i+1}-1}$ to obtain size $O(kn + n^{1+\nu})$, as we did before.

D.2 Proof of Theorem 6

We use the corresponding analysis of the emulator from Section 3. The use of half-bunches instead of bunches creates the following version of Lemma 2.

▶ **Lemma 19.** *Fix $\Delta > 5$. Let $0 \leq i < k$ and let $x, y \in V$ such that $d_G(x,y) \geq (3\Delta)^i W(x,y)$. Then at least one of the following holds:*

1. $d_H(x,y) \leq (1 + \frac{8i}{\Delta-5}) d_G(x,y)$
2. $d_H(x, p_{i+1}(x)) \leq \frac{2\Delta}{\Delta-5} d_G(x,y)$

The main difference in the proof is in (7), which is replaced by

$$d_H(p_{i+1}(u_l), p_{i+2}(p_{i+1}(u_l))) \leq 2 d_G(p_{i+1}(u_l), p_{i+1}(u_r)) .$$

The new bounds in the Lemma guarantee the calculations still go through. For Lemma 3 which takes care of small distances, we have the following change, with a very similar proof.

▶ **Lemma 20.** *Let $0 \leq i < k$ and fix $x, y \in V$. Let $m = \max\{d_G(x, p_i(x)), d_G(y, p_i(y)), d_G(x,y)\}$. Then at least one of the following holds:*

1. $d_H(x,y) \leq 5m$
2. $d_H(x, p_{i+1}(x)) \leq 7m$

In the proof we set $\Delta = 5 + \frac{8(k-1)}{\varepsilon}$. The rest of the calculations follow analogously, one change is that when iteratively applying Lemma 20, the bound m increases by a factor of 7 (rather than 4, as in Lemma 3), but as $7 \leq 3\Delta$ is still true, it does not change anything.

D.3 Proof of Theorem 7

The stretch analysis is very similar to that of the emulator from Section B, the main difference is in the use of half-bunches rather than the full ones, but this will increase the distance to pivots by a factor of 2, and affect the additive stretch only. We follow the analysis and notation presented in Section B, but with $\Delta > 5$. We replace Lemma 15 with the following.

▶ **Lemma 21.** *Let $0 \leq i \leq k$ and let $x, y \in V$ such that $d_G(x,y) \leq D_i$ and $d_H(x, p_i(x)) \leq \frac{3\Delta}{\Delta-5} D_{i-1}$. Define $m = \max\{\Delta D_{i-1}, d_G(x,y)\}$. Then at least one of the following holds:*

1. $d_H(x,y) \leq (3 + \frac{16}{\Delta-5})m$.
2. $d_H(x, p_{i+1}(x)) \leq \frac{4\Delta}{\Delta-5} D_i$.

The main difference in the proof is in (15), which is replaced by

$$d_H(p_{i+1}(x), p_{i+2}(p_{i+1}(x))) \leq 2 d_G(p_{i+1}(x), p_{i+1}(y)) \ ,$$

since we use half-bunches. One can then follow the calculations in the proof of Lemma 15, and check that the altered constants used in the 2 cases above suffice.

The proof of Theorem 7 is the same as the proof of Theorem 16, the only differences are taking $\Delta = 5 + \frac{16}{\varepsilon}$ (which affects the value of β) and using the bounds of Lemma 21 rather than of Lemma 15.