Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths

Michael Elkin* and Ofer Neiman*
*Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
Email: {elkinm,neimano}@cs.bgu.ac.il

***Abstract*—A (β, ϵ)-hopset for a weighted undirected n-vertex graph $G = (V, E)$ is a set of edges, whose addition to the graph guarantees that every pair of vertices has a path between them that contains at most β edges, whose length is within $1 + \epsilon$ of the shortest path. In her seminal paper, Cohen [8, JACM 2000] introduced the notion of hopsets in the context of parallel computation of approximate shortest paths, and since then it has found numerous applications in various other settings, such as dynamic graph algorithms, distributed computing, and the streaming model.**

Cohen [8] devised efficient algorithms for constructing hopsets with *polylogarithmic* in n number of hops. Her constructions remain the state-of-the–art since the publication of her paper in STOC'94, i.e., for more than two decades.

In this paper we exhibit the first construction of sparse hopsets with a *constant number of hops*. We also find efficient algorithms for hopsets in various computational settings, improving the best known constructions. Generally, our hopsets strictly outperform the hopsets of [8], both in terms of their parameters, and in terms of the resources required to construct them.

We demonstrate the applicability of our results for the fundamental problem of computing approximate shortest paths from s sources. Our results improve the running time for this problem in the parallel, distributed and streaming models, for a vast range of s.

***Keywords*-hopsets; shortest paths; graphs**

I. Introduction

A. Hopsets, Setting and Main Results

We are given an n-vertex weighted undirected graph $G = (V, E, \omega)$. Consider another graph $G_H = (V, H, \omega_H)$ on the same vertex set V. Define the union graph $G' = G \cup G_H$, $G' = (V, E' = E \cup H, \omega')$, where $\omega'(e) = \omega_H(e)$ for $e \in H$, and $\omega'(e) = \omega(e)$ for $e \in E \setminus H$. For a positive integer parameter β, and a pair $u, v \in V$ of distinct vertices, a β-*limited distance* between u and v in G', denoted $d^{(\beta)}_{G'}(u, v)$, is the length of the shortest u-v path in G' that contains at most β edges (aka *hops*). For a parameter $\epsilon > 0$, and a positive integer β as above, a graph $G_H = (V, H, \omega_H)$ is called a (β, ϵ)-*hopset* for the graph G, if for every pair $u, v \in V$ of vertices, we have $d_G(u, v) \leq d^{(\beta)}_{G'}(u, v) \leq (1 + \epsilon) \cdot d_G(u, v)$. (Here $d_G(u, v)$ stands for the distance between u and v in G.) We often refer to the edge set H of G_H as the *hopset*. The parameter β is called the *hopbound* of the hopset.

Hopsets are a fundamental graph-algorithmic construct. They turn out to be extremely useful for computing approximate shortest paths, distances, and for routing problems in numerous computational settings, in which computing shortest paths with a limited number of hops is significantly easier than computing shortest paths with no limitation on the number of hops. A partial list of these settings includes distributed, parallel, streaming and centralized dynamic models.

Hopsets were explicitly introduced in Cohen's seminal STOC'94 paper [8]. Implicit constructions of hopsets were given already in the beginning of the 1990's by Ullman and Yannakakis [23], Klein and Sairam [16], Cohen [7], and Shi and Spencer [20]. Cohen [8] showed that for any parameters $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, and any n-vertex graph G, there exists a (β, ϵ)-hopset H with $|H| = \tilde{O}(n^{1+1/\kappa})$ edges,[1] where the hopbound β is polylogarithmic in n. Specifically, it is given by $\beta = \left(\frac{\log n}{\epsilon}\right)^{O(\log \kappa)}$. Algorithmically, she showed that given an additional parameter $\rho > 0$, (β, ϵ)-hopsets with

$$\beta_{Coh} = \left(\frac{\log n}{\epsilon}\right)^{O((\log \kappa)/\rho)} \tag{1}$$

can be computed in $O(|E| \cdot n^{\rho})$ time in the centralized model of computation, and in $O(\beta) \cdot \mathrm{polylog(n)}$ PRAM time, with $O(|E| \cdot n^{\rho})$ work. She used these hopsets' constructions to devise efficient parallel algorithms for computing $S \times V$ $(1+\epsilon)$-approximate shortest paths (henceforth, $(1+\epsilon)$-ASP). Her results for these problems remained the state-of-the-art in this context up until now, for over two decades.

Despite being a major breakthrough, Cohen's hopsets leave much to be desired. Indeed, the only general lower bound applicable to them is that of [5] (based on [24], [1]), asserting that there exist n-vertex graphs for which any (β, ϵ)-hopset requires $\Omega(n \cdot \log^{(\lfloor \beta/2 \rfloor)} n)$ edges, where $\log^{(t)} n$ stands for a t-iterated logarithm. Cohen [8] herself wrote in the introduction of her paper (the italics are in [8]):

"One intriguing issue is the *existence* question of sparse hop sets with certain attributes. In addition, we would like to construct them efficiently."

The same motif repeats itself in the concluding section of her paper, where she writes:

[1]The notation $\tilde{O}(f(n))$ stands for $O(f(n) \cdot \log^{O(1)} f(n))$.

"We find the existence of good hop sets to be an intriguing research problem on its own right."

In the more than twenty years that passed since Cohen's [8] paper was published (in STOC'94), numerous additional applications of hopsets were discovered, and also some new constructions of hopsets were presented. Most notably, Bernstein [4] and Henzinger et al. [14] devised new constructions of hopsets, and used them for maintaining approximate shortest paths in dynamic centralized setting. Nanongkai [19] and Henzinger et al. [15] used hopsets for computing approximate shortest paths in distributed and streaming settings. Lenzen and Patt-Shamir [17] and the authors of the current paper [11] used them for compact routing. Miller et al. [18] devised new constructions of hopsets, and used them for approximate shortest paths in PRAM setting. The hopsets of [4], [14], [19], [15] have hopbound $2^{\tilde{O}(\sqrt{\log n})}$, and size $n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$, where Λ is the aspect ratio of the graph.[2] The hopsets of Miller et al. [18] have hopbound at least $\Omega(n^{\alpha})$, for a constant $\alpha > 0$, and linear size.

We will discuss these results of [4], [14], [19], [15], [18] in greater detail in the sequel. However, they all fail to address the fundamental challenge of Cohen [8], concerning the existence and efficient constructability of hopsets that are strictly and substantially superior to those devised in [8]. In this paper we build such hopsets. Specifically, for any $\epsilon > 0$, $\kappa = 1, 2 \ldots$, and any n-vertex graph $G = (V, E)$, we show that there exists a (β, ϵ)-hopset with $\beta = \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa}$, and $O(n^{1+1/\kappa} \cdot \log n)$ edges. Hence, these hopsets *simultaneously* exhibit arbitrarily small constant approximation factor $1 + \epsilon$, arbitrarily close to 1 constant exponent $1+1/\kappa$ of the hopset's size, and *constant* hopbound β. In all previous hopsets' constructions, the hopbound was at least *polylogarithmic* in n, in all regimes. Moreover, we devise efficient algorithms to build our hopsets in various computational models. Specifically, given a parameter $\rho > 0$ that controls the running time, our centralized algorithm constructs a (β, ϵ)-hopset with $O(n^{1+1/\kappa} \cdot \log n)$ edges in expected $O(|E| \cdot n^{\rho})$ time, with

$$\beta \;=\; O\left(\frac{1}{\epsilon} \cdot (\log \kappa + 1/\rho)\right)^{\log \kappa + O(1/\rho)} . \qquad (2)$$

Again, we can simultaneously have arbitrarily small constant approximation $1 + \epsilon$, arbitrarily close to 1 hopset's size exponent $1 + 1/\kappa$, arbitrarily close to $O(|E|)$ running time (in the sense $O(|E| \cdot n^{\rho})$, for an arbitrarily small constant $\rho > 0$), and still the hopbound β of the constructed hopset remains *constant*!

[2]The aspect ratio of a graph G is defined by the ratio of the largest distance to the smallest distance in G.

As was mentioned above, in [8] (the previous state-of-the-art), with the same approximation factor, hopset size and running time, the hopbound behaves as given in (1). Hence our result is stronger than that of [8] in a number of senses. First, the hopbound β_{Coh} is at least polylogarithmic, while ours is constant. Second, the exponent $O((\log \kappa)/\rho)$ of β_{Coh} is substantially larger than the exponent $\log \kappa + O(1/\rho)$ in our β.

The parameters of our hopset also strictly dominate those of the hopsets of [4], [14], [15]; see below. The hopsets' construction of [18] is incomparable to ours, as our hopbound is much better than $n^{\Omega(1)}$ of [18], but our hopset's size is at least $\Omega(n \log n)$, while the hopset of [18] has linear size. At its largest (i.e., when our hopset has size $O(n \log n)$), our hopbound is bounded by $O(\frac{\log \log n + 1/\rho}{\epsilon})^{\log \log n + O(1/\rho)}$. See Table I for a concise comparison of existing hopsets' constructions.

B. Hopsets in Parallel, Streaming and Distributed Models

We also devise efficient parallel, distributed and streaming algorithms for constructing hopsets with constant hopbound.

1) Hopsets in the Streaming Model: In the streaming model, the only previously known algorithm for constructing hopsets is that of [15]. Using $2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ passes over the stream, and $n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ space, their algorithm produces a hopset with hopbound $\beta = 2^{\tilde{O}(\sqrt{\log n})}$, and size $n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$. Our streaming algorithm constructs a hopset with β

$$\beta = O\left(\frac{\log \kappa + 1/\rho}{\epsilon \cdot \rho}\right)^{\log \kappa + O(1/\rho)} , \qquad (3)$$

i.e., it is independent of n. The expected size of the hopset is $O(n^{1+1/\kappa} \cdot \log n)$, and it uses space $O(n^{1+1/\kappa} \cdot \log^2 n)$. The number of passes is $O(n^{\rho} \cdot \beta)$. Also, by setting $\kappa = \Theta(\log n)$, $\rho = \sqrt{\frac{\log \log n}{\log n}}$, our result strictly dominates that of [4], [14], [15]; the hopbound and number of passes are essentially the same, while our space usage and hopset's size are significantly better.

2) Hopsets in the PRAM Model: In the PRAM model, Klein and Sairam [16] and Shi and Spencer [20] (implicitly) devised algorithms for constructing exact ($\epsilon = 0$) hopsets with hopbound $\beta = O(\sqrt{n})$ of linear size $O(n)$, in parallel time $O(\sqrt{n} \cdot \log n)$, and with $O(|E| \cdot \sqrt{n})$ work. (Work is the total number of operations performed by all processors during the algorithm.) Cohen [8] constructed (β, ϵ)-hopsets with size $n^{1+1/\kappa} \cdot (\log n)^{O((\log \kappa)/\rho)}$, with hopbound β_{Coh} given by (1), in parallel time $\left(\frac{\log n}{\epsilon}\right)^{O((\log \kappa)/\rho)}$, using $O(|E| \cdot n^{\rho})$ work. Her κ and ρ are restricted by $\kappa, 1/\rho = O(\log \log n)$, and thus the resulting hopset is never sparser than $n \cdot 2^{O(\frac{\log n}{\log \log n})}$.

Miller et al. [18] devised two constructions of linear-size (β, ϵ)-hopsets, but with very large β. One has $\beta = O_{\epsilon}(n^{\frac{4+\alpha}{4+2\alpha}})$, and running time given by the same expression,

Reference	Size	Hopbound	Run-time
[4], [14], [15]	$n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$	$2^{\tilde{O}(\sqrt{\log n})}$	
[18]	$O(n)$	n^{α}, $(\alpha = \Omega(1))$	
[8]	$O(n^{1+\frac{1}{\kappa}} \cdot \log n)$	$(\log n)^{O(\frac{\log \kappa}{\rho})}$	$\lvert E\rvert \cdot n^{\rho}$
This paper	$O(n^{1+\frac{1}{\kappa}} \cdot \log n)$	$O\left(\log \kappa + \frac{1}{\rho}\right)^{\log \kappa + O(\frac{1}{\rho})}$	$\lvert E\rvert \cdot n^{\rho}$
	$O(n \log n)$	$O(\log\log n + 1/\rho)^{\log\log n + O(1/\rho)}$	$\lvert E\rvert \cdot n^{\rho}$

Table I
COMPARISON BETWEEN (β, ϵ)-HOPSETS (NEGLECTING THE DEPENDENCY ON ϵ). WE NOTE THAT THE HOPSETS OF [4], [14], [15], [18] WERE DESIGNED FOR CERTAIN COMPUTATIONAL MODELS (I.E., DYNAMIC, STREAMING, DISTRIBUTED). THE SECOND ROW OF OUR RESULT FOLLOWS FROM THE FIRST ONE BY SUBSTITUTING $\kappa = \log n$.

and work $O(|E| \cdot \log^{3+\alpha} n)$, for a free parameter α. Another has $\beta = n^{\alpha}$, for a constant α, and running time given by the same expression, and work $O(|E| \cdot \log^{O(1/\alpha)} n)$.

Our algorithm has two regimes. In the first regime it constructs (β, ϵ)-hopsets with $\beta = \left(\frac{\log n}{\epsilon}\right)^{\log \kappa + O(1/\rho)}$, with expected size $O(n^{1+1/\kappa} \cdot \log n)$, in time $\left(\frac{\log n}{\epsilon}\right)^{\log \kappa + O(1/\rho)}$, using $O(|E| \cdot n^{\rho})$ work. This result strictly improves upon Cohen's hopset [8], as the exponent of β and of the running time in the latter is $O((\log \kappa)/\rho)$, instead of $\log \kappa + O(1/\rho)$ in our case. Also, the size of our hopset is smaller than that of [8] by a factor of $O(\log n)^{O((\log \kappa)/\rho)}$.

In the second regime our PRAM algorithm computes a hopset with constant (i.e., independent of n) hopbound β, but in larger parallel time. See Table II for a concise comparison of available PRAM algorithms.

C. Hopsets in Distributed Models

There are two distributed models in which hopsets were studied in the literature [14], [19], [15], [17], [11]. These are the Congested Clique model, and the CONGEST model. In both models every vertex of an n-vertex graph $G = (V, E)$ hosts a processor, and the processors communicate with one another in discrete rounds, via short messages. Each message is allowed to contain an identity of a vertex or an edge, and an edge weight, or anything else of no larger (up to a fixed constant factor) size.[3] On each round each vertex can send possibly different messages to its neighbors. The local computation is assumed to require zero time, and we are interested in algorithms that run for as few rounds as possible. (The number of rounds is called the *running time*.) In the Congested Clique model, we assume that all vertices are interconnected via direct edges. In the CONGEST model, every vertex can send messages only to its G-neighbors, but we also assume that there is an embedded "virtual" graph $G' = (V', E', \omega)$, $V' \subseteq V$, known locally to the vertices. (Every vertex $u \in V$ knows at the beginning of the computation if $u \in V'$, and if it is the case, then it also knows the identities of its G'-neighbors.) We remark that the assumption of embedded graph G' in the CONGEST model appears in previous papers on computing hopsets in distributed setting, that is, in [14], [19], [15], [17], [11]. It is motivated by distributed applications of hopsets, i.e., approximate shortest paths computation, distance estimation and routing, which require a hopset for a virtual graph embedded in the underlying network in the above way.

Henzinger et al. [15] devised an algorithm for constructing hopsets in the Congested Clique model. Their hopset has hopbound $\beta = 2^{\tilde{O}(\sqrt{\log n})}$, and size $n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$, where Λ is the aspect ratio of the embedded graph. The running time of their algorithm is $2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$.

Our algorithm, for parameters $\epsilon > 0$, $\rho > 0$, $\kappa = 2, 3, \ldots$, computes a hopset with β as in (3), with expected size $O(n^{1+1/\kappa} \cdot \log n)$, in $O(n^{\rho} \cdot \beta^2)$ rounds.

Comparing our result to that of [15], we first note that our hopset achieves a constant (i.e., independent of n) hopbound. Second, by setting $\kappa = \Theta(\log n)$, $\rho = \sqrt{\frac{\log\log n}{\log n}}$, we can have our hopbound and running time equal to $2^{\tilde{O}(\sqrt{\log n})}$, i.e., roughly the same as, but in fact, slightly better than, the respective bounds of [15]. Our hopset's size becomes then $O(n \cdot \log n)$, i.e., much closer to linear than $n \cdot 2^{\tilde{O}(\sqrt{\log n})}$ of the hopset of [15].

The situation is similar in the CONGEST model. Denote by $m = |V'|$ the size of the vertex set of the embedded graph G'. The algorithm of [15] computes a hopset with the same hopbound and size as in the Congested Clique model (with n replaced by m), and it does so in $(D + m) \cdot 2^{\tilde{O}(\sqrt{\log m})} \cdot \log \Lambda$ time, where Λ is the aspect ratio of G', and D is the hop-diameter of G.[4] Our algorithm computes a hopset with (constant) hopbound given by (3), expected size $O(m^{1+1/\kappa} \cdot \log m)$, in $O((D + m^{1+\rho}) \cdot \beta \cdot m^{\rho})$ time. (In the full version we provide stronger but more complicated bounds.) Again, our hopset can have constant hopbound, while that of [15] is $2^{\tilde{O}(\sqrt{\log m})}$. Also, by setting $\kappa = \Theta(\log m)$, $\rho = \sqrt{\frac{\log\log m}{\log m}}$, we obtain a result, which strictly dominates that of [15].

[3]Typically, in the CONGEST model only messages of size $O(\log n)$ bits are allowed, but edge weights are restricted to be at most polynomial in n. Our definition is geared to capture a more general situation, when there is no restriction on the aspect ratio. Hence results achieved in our more general model are more general than previous ones.

[4]The hop-diameter of a graph is the maximum hop-distance between two vertices. The hop-distance between a pair u, v of vertices is the minimal number of hops in a path between them.

Reference	Size	β = Hopbound	Time	Work
[16], [20]	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n}\log n)$	$O(\|E\| \cdot \sqrt{n})$
[18]	$O(n)$	$O(n^{\frac{4+\alpha}{4+2\alpha}})$	$O(n^{\frac{4+\alpha}{4+2\alpha}})$	$O(\|E\| \cdot \log^{3+\alpha} n)$
	$O(n)$	$O(n^{\alpha})$ $(\alpha \geq \Omega(1))$	$O(n^{\alpha})$	$O(\|E\| \cdot \log^{O(1/\alpha)} n)$
[8]	$n^{1+1/\kappa} \cdot (\log n)^{O(\frac{\log \kappa}{\rho})}$	$(\log n)^{O(\frac{\log \kappa}{\rho})}$	$(\log n)^{O(\frac{\log \kappa}{\rho})}$	$O(\|E\| \cdot n^{\rho})$
This paper	$O(n^{1+\frac{1}{\kappa}} \cdot \log n)$	$(\log n)^{\log \kappa + O(\frac{1}{\rho})}$	$(\log n)^{\log \kappa + O(\frac{1}{\rho})}$	$O(\|E\| \cdot n^{\rho})$
	$O(n^{1+\frac{1}{\kappa}} \cdot \log n)$	$O\left(\frac{\log \kappa + \frac{1}{\rho}}{\zeta}\right)^{\log \kappa + O(\frac{1}{\rho})}$	$O(n^{\zeta}) \cdot \beta$	$O(\|E\| \cdot n^{\rho+\zeta})$
	$O(n \cdot \log n)$	$O\left(\frac{\log \log n + 1/\rho}{\zeta}\right)^{\log \log n + O(1/\rho)}$	$O(n^{\zeta}) \cdot \beta$	$O(\|E\| \cdot n^{\rho+\zeta})$

Table II
COMPARISON BETWEEN (β, ϵ)-HOPSETS IN THE PRAM MODEL (NEGLECTING THE DEPENDENCY ON ϵ). THE HOPSETS OF [16], [20] PROVIDE EXACT DISTANCES. THE THIRD ROW OF OUR RESULT FOLLOWS FROM THE SECOND ONE BY SUBSTITUTING $\kappa = \log n$.

D. Applications

Our algorithms for constructing hopsets also give rise to improved algorithms for the problems of computing $(1+\epsilon)$-approximate shortest distances (henceforth, $(1+\epsilon)$-ASD) and paths (henceforth, $(1+\epsilon)$-ASP). In all settings, we consider a subset $S \subseteq V$ of origins, and we are interested in distance estimates or in approximate shortest paths for pairs in $S \times V$. Denote $s = |S|$.

Our PRAM algorithm for the $(1+\epsilon)$-ASP problem has running time $O\left(\frac{\log n}{\epsilon}\right)^{\log \kappa + 1/\rho + 1}$, and uses $O(|E| \cdot (n^{\rho} + s))$ work. Cohen's algorithm [8] for the same problem has (parallel) running time $O\left(\frac{\log n}{\epsilon}\right)^{O((\log \kappa)/\rho)}$, and has the same work complexity as our algorithm. Hence, both our and Cohen's algorithms achieve polylogarithmic time and near-optimal work complexity, but the exponent $\log \kappa + 1/\rho + 1$ of the logarithm in our result is significantly smaller than in Cohen's one. The latter is $O((\log \kappa)/\rho)$.

In the distributed CONGEST model, the hopset-based algorithm of [15] computes single-source $(1+\epsilon)$-ASP in $(D + \sqrt{n}) \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ time. Using it naively for $S \times V$ $(1+\epsilon)$-ASP results in running time of $(D + s \cdot \sqrt{n}) \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$. Using our hopsets we solve this problem in $(D + \sqrt{n \cdot s}) \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ time. Whenever $s = n^{\Omega(1)}$, we use our hopset with different parameters, and our running time becomes $\tilde{O}(D + \sqrt{n \cdot s}) \cdot \log \Lambda$. We also remark that if large messages of size n^{ρ} are allowed for some constant $\rho > 0$, then we can compute single-source $(1+\epsilon)$-ASP in $\tilde{O}(D + \sqrt{n}) \cdot \log \Lambda$ time, and in the Congested Clique model we obtain single source $(1+\epsilon)$-ASP in polylogarithmic time.

In the streaming model, Henzinger et al. [15] devised a single-source $(1+\epsilon)$-ASP streaming algorithm with $2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ passes, that uses $n \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$ space. To the best of our knowledge, the best-known streaming $S \times V$ $(1+\epsilon)$-ASP algorithm with this space requirement is to run the algorithm of [15] for each source separately, one after another. The resulting number of passes is $s \cdot 2^{\tilde{O}(\sqrt{\log n})} \cdot \log \Lambda$. Our algorithm for this problem builds a hopset, whose parameters depend on s. As a result, our algorithm has an improved number of passes, particularly when s is large (we also avoid the dependence on Λ). Our space usage is only $\tilde{O}(n)$ for $(1+\epsilon)$-ASD. Moreover, allowing larger space we can get optimal $O(s)$ passes for any fixed $\epsilon > 0$ whenever $s = n^{\Omega(1)}$. See the full version [10] for the precise results.

E. Overview of Techniques

In this section we sketch the main ideas used in the hopsets' constructions of [8], in [4], [19], [14], [15], and in our constructions.

Cohen's algorithm [8] starts with constructing a pairwise cover $\mathcal{C}$ of the input graph [6], [2]. This is a collection of small-diameter clusters, with limited intersections, and such that for any path π of length at most W, for a parameter W, all vertices of π are clustered in the same cluster. For each cluster $C \in \mathcal{C}$, the algorithm inserts into the hopset a star $\{(r_C, u) \mid u \in C\}$ connecting the center r_C of C with every other vertex of C. In addition, it adds to the hopset edges connecting centers of large clusters with one another, and recurses on small clusters.

This powerful approach has a number of limitations. First, the collection of star edges itself contains $O(\kappa \cdot n^{1+1/\kappa})$ edges, where κ is a parameter, which controls the hopset's size. Each level of the recursion increases the exponent of the number of edges in the hopset by roughly a factor of $\kappa \cdot n^{1/\kappa}$, and as a result, the hopset of [8] cannot be very sparse. Second, each distance scale $[2^k, 2^{k+1}]$, $k = 0, 1, 2, \ldots$, requires a separate hopset, and as a result, a separate collection of covers. This increases the hopset's size even further, but in addition, a hopset of scale $k+1$ in Cohen's algorithm is computed using hopsets of all the lower scales. This results in accumulation of error, i.e., if the error incurred by each hopset computation is $1+\epsilon$, the approximation factor of the ultimate hopset becomes $(1+\epsilon)^{\log \Lambda}$. After rescaling $\epsilon' = \epsilon \log \Lambda$, one obtains a hopbound of roughly $(1/\epsilon)^{\ell} = O(\frac{\log \Lambda}{\epsilon'})^{\ell}$, where ℓ is the number of levels of the recursion. As a result, the hopbound in [8] is at least polylogarithmic in n.

Another line of works [4], [14], [15], [19] is based on the distance oracles and emulators[5] of Thorup and Zwick [21], [22]. They build a hierarchy of sampled sets $V = A_0 \supset A_1 \supset \dots A_{k-1} \supset A_k = \emptyset$, where for any $i = 1, \dots, k-1$, each vertex $v \in A_{i-1}$ joins A_i independently at random with probability $n^{-1/k}$. For each vertex $v \in V$, one can define the *TZ cluster* $C(v)$ by $C(v) = \bigcup_{i=0}^{k-1}\{u \mid u \in A_i, d_G(u,v) < d_G(u, A_{i+1})\}$. Thorup and Zwick [22] showed that for *unweighted* graphs $H = \{(v,u) \mid u \in C(v)\}$ is a $(1+\epsilon,\beta)$-emulator with $O(k \cdot n^{1+1/k})$ edges, and $\beta = O(k/\epsilon)^k$. Bernstein and others [4], [14], [15], [19] showed that a closely related construction provides a hopset. Specifically, they set $k = \Theta(\sqrt{\log n})$, and build TZ clusters with respect to $2^{\tilde{O}(\sqrt{\log n})}$-limited distances. This results in a so-called *restricted hopset*, i.e., a hopset H_1 that handles $2^{\tilde{O}(\sqrt{\log n})}$-limited distances. Consequently, all nearly shortest paths with N hops in G, for some N, translate now into nearly shortest paths (incurring an approximation factor of $1+\epsilon$ of H_1) with $\frac{N}{2^{\tilde{O}(\sqrt{\log n})}}$ hops in $G \cup H_1$. Nanongkai [19] called this operation a *hop reduction*, as this essentially reduces the maximum number of hops from $n-1$ to $n/2^{\sqrt{\log n}}$. Then the hop reduction is repeated for $\sqrt{\log n}$ times, until a hopset for all distances is constructed.

This scheme appears to be incapable of providing very sparse hopsets, as just the invocation of Thorup-Zwick's algorithm with $\kappa = \Theta(\sqrt{\log n})$ gives $n \cdot 2^{\Omega(\sqrt{\log n})}$ edges. In addition, the repetitive application of hop reduction blows up the hopbound to $2^{\Omega(\sqrt{\log n})}$, i.e., the large hopbound appears to be inherent in this approach.

Our approach combines techniques from [12] for constructing $(1+\epsilon,\beta)$-spanners in unweighted graphs with those of [8], and with a suit of new ideas. To build their spanners, [12] start with constructing an Awerbuch-Peleg's partition $\mathcal{P} = \{C_1, \dots, C_q\}$ [3] of the vertex set V into disjoint clusters of small diameter. (This partition satisfies an additional property which is irrelevant to this discussion.) It then sets a distance threshold δ_1 and a degree threshold deg_1. Every cluster $C \in \mathcal{P}$ that has at least deg_1 unclustered clusters $C' \in \mathcal{P}$ in its δ_1-vicinity creates a supercluster which contains C and these clusters. (At the beginning all clusters are unclustered. Those that join a supercluster become clustered.) This *superclustering step* continues until no additional superclusters can be formed. All the remaining unclustered clusters which are at pairwise distance at most δ_1 are now interconnected by shortest paths in the spanner. This is the *interconnection* step of the algorithm. Together the superclustering and interconnection steps from a single *phase* of the algorithm. Once the first phase is over, the same process (interleaving superclustering and interconnection) is repeated with new distance and degree thresholds δ_2 and deg_2, respectively, on the set of superclusters of the previous phase. The sequences $\delta_1, \delta_2, \dots$ and $deg_1, deg_2, \dots$ are set carefully to optimize the parameters of the resulting spanner.

The basic variant of our hopset construction considers each distance scale $[2^k, 2^{k+1}]$, $k = 0,1,2,\dots$, separately (w.l.o.g we assume all weights are at least 1). Instead of Awerbuch-Peleg's partition, we use the partition $\mathcal{P} = \{\{v\} \mid v \in V\}$ into single vertices. We set the distance threshold δ_1 to roughly $2^k/\beta = 2^k/(1/\epsilon)^\ell$, where ℓ is the number of phases of the algorithm, and raise it by a factor of $1/\epsilon$ on every phase. The degree thresholds are also set differently from the way they were set in [12]. This is because, intuitively, the hopset contains less edges than the spanner, as the hopset can use a single edge where a spanner needs to use an entire path. Hence the degree sequence that optimizes the hopset's size is different than the one that optimizes the spanner's size.

The superclustering and interconnection steps are also implemented in a different way than in [12], because of efficiency considerations. The algorithm of [12] is not particularly efficient, and there are no known efficient streaming, distributed or parallel implementation of it. [6] On phase i we sample clusters $C \in \mathcal{P}$ independently at random with probability $1/deg_i$. The sampled clusters create superclusters of radius δ_i around them. Then the unclustered clusters of $\mathcal{P}$ which are within distance $\delta_i/2$ from one another are interconnected by hopset edges. Note that here the superclustering distance threshold and the interconnection distance thresholds differ by a factor of 2. This ensures that all involved Dijkstra explorations can be efficiently implemented. We also show that the overhead that this factor introduces to the resulting parameters of our hopset is insignificant.

Our approach (interleaving superclustering and interconnection steps) to constructing hopsets was not previously used in the hopsets' literature [8], [4], [14], [19], [15]. Rather it is adapted from [12]. The latter paper deals with nearly-additive spanners for unweighted graphs. We believe that realizing that the technique of [12] can be instrumental for constructing drastically improved hopsets, and adapting that technique from the context of near-additive spanners for unweighted graphs to the context of hopsets for general graphs is our main technical contribution.

To construct a hopset for *all* scales, in the centralized setting we simply take the union of the single-scale hopsets. In parallel, distributed and streaming settings, however, Dijkstra explorations for large scales could be too expensive. To remedy this, we rely on lower-scales hopsets for computing the current scale, like in Cohen's algorithm. On the other

[5]A graph $G' = (V', E', \omega')$ is called a $(1+\epsilon,\beta)$-*emulator* of an unweighted graph $G = (V,E)$, if $V \subseteq V'$, and for every pair of $u, v \in V$ of vertices, it holds that $d_{G'}(u,v) \le d_G(u,v) \le (1+\epsilon)d_G(u,v) + \beta$. If G' is a subgraph of G, then G' is called a $(1+\epsilon,\beta)$-*spanner* of G.

[6]The algorithms of [9], [13] that construct $(1+\epsilon,\beta)$-spanners in distributed and streaming settings are not based on superclustering and interconnection technique. Rather they are based on a completely different approach, reminiscent to that of [8], i.e., they build covers, and recurse in small clusters.

hand, a naive application of this approach results in polylogarithmic hopbound β. To achieve *constant* (i.e., independent of n) hopbound β, we compute in parallel hopsets for many different scales, using the same low-scale hopset for distance computations. This results in a much smaller accumulation of error than in Cohen's scheme, but requires larger running time. (Roughly speaking, computing a scale-t hopset using scale-s hopset, for $t > s$, requires time proportional to 2^{t-s}.) We carefully balance this increase in running time with other parameters, to optimize the attributes of our ultimate hopset.

Finally, one needs to replace the logarithmic dependence on the aspect ratio Λ, by the same dependence on n. Cohen's results [8] do not have this dependence, as they rely on a PRAM reduction of Klein and Sairam [16]. However, Klein and Sairam [16] (see also [7] for another analysis) analyzed this reduction for single-source distance estimation, while in the hopset's case one needs to apply it to all pairs. The distributed and streaming hopsets' constructions [4], [14], [15], [19] all have a dependence on $\log \Lambda$.

We develop a new analysis of Klein-Sairam's reduction, which applies to the hopsets' scenario. We also show that the reduction can be efficiently implemented in distributed and streaming settings.

II. Preliminaries

Let $G = (V, E)$ be a weighted graph on n vertices with diameter Λ, we shall assume throughout that edge-weights are positive integers. Let d_G be the shortest path metric on G, and let $d_G^{(t)}$ be the t-limited distance, that is, for $u, v \in V$, $d_G^{(t)}(u, v)$ is the minimal length of a path between u, v that contains at most t edges (set $d_G^{(t)}(u, v) = \infty$ if there is no such path). Note that $d_G^{(t)}$ is not a metric.

III. Hopsets

We show here the centralized (i.e., sequential) construction of our hopsets, and defer the other results to the full version.

Let $G = (V, E)$ be a weighted graph on n vertices with diameter Λ. Fix parameters $\kappa \geq 1$, $0 < \epsilon < 1/10$ and $1/\kappa \leq \rho < 1/2$. Recall that κ governs the sparsity of the hopset, ϵ governs its stretch, and ρ governs the running time. The parameter β, which governs the hopbound, will be determined later as a function of $n, \Lambda, \kappa, \rho, \epsilon$. ($\Lambda$ stands for the aspect ratio.) We build separately a hopset H_k for every distance range $(2^k, 2^{k+1}]$, for $k \leq \log \Lambda$. We will call such a hopset H_k a *single-scale* hopset.

Denote $\hat{R} = 2^{k+1}$. For $\hat{R} \leq \beta = (1/\epsilon)^\ell$, where ℓ is the number of levels of the construction (to be determined), an empty hopset $H_k = \emptyset$ does the job. (Indeed, for $d_G(u, v) \leq \hat{R} \leq \beta$, we have $d_{G \cup H_k}^{(\beta)}(u, v) = d_G^{(\beta)}(u, v) = d_G(u, v)$, for $H_k = \emptyset$.) Hence we assume that $k > \log \beta - 1$, i.e., $\hat{R} > \beta$.

The algorithm initializes the hopset H_k as an empty set, and proceeds in phases. It starts with setting $\hat{\mathcal{P}}_0 = \{\{v\} \mid v \in V\}$ to be the partition of V into singleton clusters. The partition $\hat{\mathcal{P}}_0$ is the input of phase 0 of our algorithm. More generally, $\hat{\mathcal{P}}_i$ is the input of phase i, for every index i in a certain appropriate range, which we will specify in the sequel.

Throughout the algorithm, all clusters C that we will construct will be centered at designated centers r_C. In particular, each singleton cluster $C = \{v\} \in \hat{\mathcal{P}}_0$ is centered at v. We define $Rad(C) = \max\{d_{G(C)}(r_C, v) \mid v \in C\}$, and $Rad(\hat{\mathcal{P}}_i) = \max_{C \in \hat{\mathcal{P}}_i}\{Rad(C)\}$.

All phases of our algorithm except for the last one consist of two steps. Specifically, these are the *superclustering* and the *interconnection* steps. The last phase contains only the interconnection step, and the superclustering step is skipped. We also partition the phases into two *stages*. The first stage consists of phases $0, 1, \ldots, i_0 = \lfloor \log(\kappa\rho) \rfloor$, and the second stage consists of all the other phases $i_0 + 1, \ldots, i_1$ where $i_1 = i_0 + \left\lceil \frac{\kappa+1}{\kappa\rho} \right\rceil - 2$, except for the last phase $\ell = i_1 + 1$. The last phase will be referred to as the *concluding phase*.

Each phase i accepts as input two parameters, the distance threshold parameter δ_i, which determines the range of the Dijkstra explorations, and the degree parameter deg_i, which determines the sampling probability. The difference between stage 1 and 2 is that in stage 1 the degree parameter grows exponentially, while in stage 2 it is fixed. The distance threshold parameter grows in the same steady rate (increases by a factor of $1/\epsilon$) all through the algorithm.

The distance thresholds' sequence is given by $\alpha = \epsilon^\ell \cdot \hat{R}$, $\delta_i = \alpha(1/\epsilon)^i + 4R_i$, where $R_0 = 0$ and $R_{i+1} = \delta_i + R_i = \alpha(1/\epsilon)^i + 5R_i$, for $i \geq 0$. It follows that $R_1 = \alpha$, and by estimating the recurrence we obtain $R_i \leq 2 \cdot \alpha \cdot (1/\epsilon)^{i-1}$. The degree sequence in the first stage of the algorithm is given by $deg_i = n^{2^i/\kappa}$, for $i = 0, 1, \ldots, i_0$. We then use $deg_i = n^\rho$ in all subsequent phases $i_0 + 1, \ldots, i_1$. Finally, on phase $\ell = i_1 + 1$ we perform just the interconnection step. Note that $\ell \geq 2$ since $\rho < 1/2$.

Next we take a closer look on the execution of phase i, $i = 0, 1, 2, \ldots, \ell - 1$. At the beginning of the phase we have a collection $\hat{\mathcal{P}}_i$ of clusters, of radius $2\alpha \cdot (1/\epsilon)^{i-1}$, for $i \geq 1$, and radius 0 for $i = 0$. (It will be shown in Claim III.1 that $Rad(\hat{\mathcal{P}}_i) \leq R_i = 2\alpha \cdot (1/\epsilon)^{i-1}$, for all $i = 0, 1, \ldots, \ell$.) Each of these clusters is now sampled with probability $1/deg_i$, i.a.r.. The resulting set of sampled clusters is denoted $\mathcal{S}_i$. We then initiate a single Dijkstra exploration in G rooted at the set $Roots = \{r_C \mid C \in \mathcal{S}_i\}$ of cluster centers of sampled clusters. The Dijkstra exploration is conducted to depth δ_i. Let F_i denote the resulting forest.

Let $C' \in \hat{\mathcal{P}}_i \backslash \mathcal{S}_i$ be a cluster whose center $r_{C'}$ was reached by the exploration, and let r_C, for some cluster $C \in \mathcal{S}_i$, be the cluster center such that $r_{C'}$ belongs to the tree of F_i rooted at r_C. We then add an edge $(r_C, r_{C'})$ of weight $\omega(r_C, r_{C'}) = d_G(r_C, r_{C'})$ into the hopset H_k, which we are now constructing. A supercluster $\hat{C}$ rooted at $r_{\hat{C}} = r_C$ is now created. It contains all vertices of C and of clusters C' as above. This completes the description of the superclustering step. The resulting set $\hat{\mathcal{S}}_i$ of superclusters becomes the next level partition $\hat{\mathcal{P}}_{i+1}$, i.e., we set $\hat{\mathcal{P}}_{i+1} \leftarrow \hat{\mathcal{S}}_i$.

Claim III.1. *Fix any cluster $C \in \hat{\mathcal{P}}_i$ with center r_C. Then for any $u \in C$ there is a path in H_k of at most i edges from r_C to u of length at most R_i.*

Proof: The proof is by induction on i, the basis $i = 0$ holds as C is a singleton. Assume it holds for i, and fix any $\hat{C} \in \hat{\mathcal{P}}_{i+1}$ and $u \in \hat{C}$. Recall that $\hat{C}$ consists of a sampled cluster $C \in \mathcal{S}_i$, and clusters $C' \in \hat{\mathcal{P}}_i$ for which the Dijkstra exploration to range δ_i from r_C reached their center $r_{C'}$. Assume $u \in C'$ (the case where $u \in C$ is simpler). Then by induction there is a path of length at most R_i from $r_{C'}$ to u in H_k of i hops, and by construction we added the edge $(r_C, r_{C'})$ of weight $d_G(r_C, r_{C'})$ into the hopset H_k. This implies a path of $i+1$ hops and length at most

$$\delta_i + R_i = R_{i+1} \ .$$

■

Let $\hat{\mathcal{U}}_i$ denote the set of $\hat{\mathcal{P}}_i$ clusters which were not superclustered into $\hat{\mathcal{S}}_i$ clusters. These clusters are involved in the interconnection step. Specifically, each of the cluster centers r_C, $C \in \hat{\mathcal{U}}_i$, initiates now a separate Dijkstra exploration to depth $\frac{1}{2}\delta_i = \frac{1}{2}\alpha \cdot (1/\epsilon)^i + 2R_i$. For any cluster center $r_{C'}$ of a cluster $C' \in \hat{\mathcal{U}}_i$ such that $r_{C'}$ was discovered by an exploration originated at r_C, we now insert an edge $(r_C, r_{C'})$ into the hopset, and assign it weight $\omega(r_C, r_{C'}) = d_G(r_C, r_{C'})$. This completes the description of the interconnection step.

Lemma III.2. *For any vertex $v \in V$, the expected number of explorations that visit v at the interconnection step of phase $0 \le i \le i_1$ is at most deg_i.*

Proof: For $0 \le i \le i_1$, assume that there are l clusters of $\hat{\mathcal{P}}_i$ within distance $\delta_i/2$ from v. If at least one of them is sampled to $\mathcal{S}_i$, then no exploration will visit v (since in the superclustering phase the sampled center will explore to distance δ_i, and thus all these l cluster will be superclustered into some cluster of $\hat{\mathcal{S}}_i$). The probability that none of them is sampled is $(1 - 1/deg_i)^l$, in which case we get that l explorations visit v, so the expectation is $l \cdot (1 - 1/deg_i)^l \le deg_i$ for any l. ■

We analyze the number of clusters in collections $\hat{\mathcal{P}}_i$ in the following lemma.

Lemma III.3. *Assuming $n^\rho = \omega(1)$, with high probability, for every $i = 0, 1, \ldots, i_0 + 1$ we have*

$$|\hat{\mathcal{P}}_i| \ \le \ 2 \cdot n^{1 - \frac{2^i - 1}{\kappa}} \ , \qquad (4)$$

and for $i = i_0 + 2, \ldots, i_1 + 1$,

$$|\hat{\mathcal{P}}_i| \ \le \ 2 \cdot n^{1 + 1/\kappa - (i - i_0)\rho} \ .$$

Proof: For the first assertion, the probability that a vertex $v \in V$ will be a center of a cluster in $\hat{\mathcal{P}}_i$ is $\prod_{j=0}^{i-1} 1/deg_j = n^{-(2^i - 1)/\kappa}$. Thus the expected size of $\hat{\mathcal{P}}_i$ is $n^{1-(2^i-1)/\kappa}$, and as these choices are made independently, by Chernoff bound,

$$\begin{aligned} \Pr[|\hat{\mathcal{P}}_i| \ge 2\mathbb{E}[|\hat{\mathcal{P}}_i|]] &\le exp\{-\Omega(\mathbb{E}[|\hat{\mathcal{P}}_i|])\} \\ &= exp\{-\Omega(n^{1-\frac{2^i-1}{\kappa}})\} \ . \end{aligned}$$

Since for $\rho < 1/2$ and $i \le i_0 + 1 = \lfloor \log \rho\kappa \rfloor + 1$, we have $n^{1-\frac{2^i-1}{\kappa}} \ge n^{1-2\rho} = \omega(\log n)$, we conclude that whp for all $0 \le i \le i_0 + 1$, $|\hat{\mathcal{P}}_i| \le 2n^{1-\frac{2^i-1}{\kappa}}$. In particular, $|\hat{\mathcal{P}}_{i_0+1}| = O(n^{1-\rho+1/\kappa})$.

For the second assertion, consider any $i \in [i_0 + 2, i_1 + 1]$, the expected size of $\hat{\mathcal{P}}_i$ is

$$\begin{aligned} \mathbb{E}[|\hat{\mathcal{P}}_i|] &= n \cdot \prod_{j=0}^{i-1} 1/deg_j \le n^{1+1/\kappa-\rho-(i-1-i_0)\rho} \\ &= n^{1+1/\kappa-(i-i_0)\rho} \ . \end{aligned}$$

Since $n^{1+1/\kappa-(i-i_0)\rho} \ge n^\rho$ for any $i \le i_1$, by Chernoff bound with probability at least $1 - exp\{-\Omega(n^\rho)\}$ (which is $1 - o(1)$ by our assumption on n^ρ), we have

$$|\hat{\mathcal{P}}_i| \ \le \ 2 \cdot n^{1+1/\kappa-(i-i_0)\rho} \ .$$

■

This lemma implies that whp

$$\begin{aligned} |\hat{\mathcal{P}}_{i_1+1}| &\le O(n^{1+1/\kappa-(i_1+1-i_0)\rho}) \qquad (5) \\ &= O(n^{1+1/\kappa-(\lceil\frac{\kappa+1}{\kappa\rho}\rceil-1)\rho}) = O(n^\rho) \ . \end{aligned}$$

For the assumption of the Lemma above to hold, we will need to assume that $\rho \ge \frac{\log\log n}{2\log n}$, say. We will show soon that this assumption is valid in our setting.

The running time required to implement the single Dijkstra exploration in the superclustering of phase i is $O(|E| + n\log n)$, while in the interconnection step, by Lemma III.2 every vertex is expected to be visited by at most deg_i explorations, so the expected running time of phase $0 \le i \le i_1$ is $O(|E| + n\log n) \cdot deg_i$. Recall that in the last phase $i_1 + 1$ there is no superclustering step, but as (5) implies, there are whp only $O(n^\rho)$ clusters, so each vertex will be visited at most $O(n^\rho)$ times. Thus the total expected running time is

$$\begin{aligned} &O(|E| + n\log n) \cdot \left(\sum_{i=0}^{\ell-1}(deg_i) + n^\rho\right) \\ &= O(|E| + n\log n) \cdot \left(\sum_{i=0}^{i_0}(n^{2^i/\kappa}) + (i_1 - i_0)n^\rho\right) \\ &= O(|E| + n\log n) \cdot (n^{2^{i_0}/\kappa} + n^\rho/\rho) \\ &= O(|E| + n\log n) \cdot n^\rho/\rho \ . \end{aligned}$$

The size of the hopset H_k that was constructed by this algorithm is dominated by the number of edges inserted by the interconnection steps, since all the edges inserted at superclustering steps induce a forest. Due to Lemma III.2, the expected number of edges inserted by the interconnection step of phase i is at most $O(|\hat{\mathcal{P}}_i| \cdot deg_i) = O(n^{1+1/\kappa})$, for $i \le i_0$, and $\sum_{i=i_0+1}^{\ell+1} O(|\hat{\mathcal{P}}_i| \cdot deg_i) = O(n^{1+1/\kappa})$ edges on the later phases. Hence overall $\mathbb{E}(|H_k|) = O(n^{1+1/\kappa} \cdot \log \kappa)$. We remark that the factor $\log \kappa$ can be eliminated from the hopset size by using a refined degree sequence, at the cost of increasing the number of phases by 1 (this will increase the exponent of β by 1). We elaborate on this in the full version of the paper. Then the number of edges contributed to the hopset H_k by all interconnection steps becomes $O(n^{1+1/\kappa})$.

Next we analyze the stretch and the hopbound of H_k. Write $H = H_k$. Observe that, by Claim III.1, $Rad(\hat{\mathcal{U}}_0) = R_0 = 0$, and, for all $i \in [1, \ell]$, $Rad(\hat{\mathcal{U}}_i) \le Rad(\hat{\mathcal{P}}_i) \le R_i \le 2\alpha(1/\epsilon)^{i-1}$. Write $c = 2$. Note also that for any pair of distinct clusters $C, C' \in \hat{\mathcal{U}}_i$, for any i, which are at distance $d_G(C, C') \le \frac{1}{2}\alpha \cdot (1/\epsilon)^i$, it holds that $d_G(r_C, r_{C'}) \le d_G(C, C') + 2R_i \le \frac{1}{2}\alpha(1/\epsilon)^i + 2 \cdot R_i = \frac{1}{2}\delta_i$. Hence for every pair of clusters C, C' as above, an edge $(r_C, r_{C'})$ of weight $\omega(r_C, r_{C'}) = d_G(r_C, r_{C'})$ belongs to the hopset.

Observe that $\hat{\mathcal{U}} = \bigcup_{i=0}^{\ell} \hat{\mathcal{U}}_i$ is a partition of G. For any i, we denote $\hat{\mathcal{U}}^{(i)} = \bigcup_{j=0}^{i} \hat{\mathcal{U}}_j$.

Lemma III.4. *Let x, y be a pair of vertices with $d_G(x, y) \le \frac{1}{2}\alpha \cdot (1/\epsilon)^i$ and such that all vertices of a shortest path $\pi(x, y)$ in G between them are clustered in $\hat{\mathcal{U}}^{(i)}$, for some $i \le \ell$. Then it holds that*

$$d^{(h_i)}_{G \cup H}(x, y) \le d_G(x, y)(1 + 16c(i-1)\epsilon) + 8\alpha c(1/\epsilon)^{i-1} , \qquad (6)$$

with h_i given by $h_0 = 1$, and $h_{i+1} = (h_i + 1)(1/\epsilon + 2) + 2i + 5$.

Proof: The proof is by induction on i. The base case is the case $i = 0$.

Base case: We assume $d_G(x, y) \le \frac{1}{2}\alpha$ and all vertices of $\pi(x, y)$ are clustered in $\hat{\mathcal{U}}_0$, then there is an edge (x, y) in H with $\omega(x, y) = d_G(x, y)$, and indeed

$$d^{(h_0)}_{G \cup H}(x, y) = d_G(x, y) \le d_G(x, y)(1 - 16c\epsilon) + 8\alpha c(1/\epsilon)^{-1} .$$

Step: We assume the assertion of the lemma for some index i, and prove it for $i + 1$.
Consider first a pair u, v of vertices such that all vertices of $\pi(u, v)$ are clustered in $\hat{\mathcal{U}}^{(i)}$, for a fixed $i < \ell$, without any restriction on $d_G(u, v)$.

We partition $\pi(u, v)$ into segments $L_1, L_2, \ldots$ of length roughly $\frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$ each in the following way. The first segment L_1 starts at u, i.e., we write $u = u_1$. Given a left endpoint u_p, $p \ge 1$, of a segment L_p, we set the right endpoint v_p of L_p to be (if exists) the farthest vertex of $\pi(u, v)$ from u_p which is closer to v than u_p, and such that $d_G(u_p, v) \le \frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$.

If v_p does not exist then the pth segment L_p is declared as *void*, and we define $v_p = u_{p+1}$ to be the neighbor of u_p on $\pi(u, v)$ which is closer to v. If v_p does exist, then u_{p+1} is (if exists) the "right" neighbor of v_p on $\pi(u, v)$, i.e., the neighbor of v_p which is closer to v than v_p is. (It may not exist only if $v_p = v$.) Observe that in either case, if u_{p+1} exists then $d_G(u_p, u_{p+1}) > \frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$.

We also define *extended* segments $\hat{L}_p$ in the following way. If L_p is a void segment, then we define $\hat{L}_p = L_p$. Otherwise $\hat{L}_p$ is the segment of $\pi(u, v)$ connecting u_p with u_{p+1}, if u_{p+1} exists, and with v_p otherwise. (This may be the case only if $L_p = \hat{L}_p$ is the last, i.e., the rightmost, segment of the path $\pi(u, v)$.)

Observe that every non-void extended segment $\hat{L}_p$, except maybe the last one, has length at least $\frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$, and every segment L_p has length at most $\frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$.

Next we construct a path $\pi'(u, v)$ in $G \cup H$, which has roughly the same length as $\pi(u, v)$, but consists of much fewer hops. Consider a segment L_p, with left endpoint u_p and right endpoint v_p, and its extended segment $\hat{L}_p$ with right endpoint u_{p+1}. We define a *substitute* segment L'_p in $G \cup H$, connecting u_p with u_{p+1} with a few hops, and of roughly the same length.

If L_p is a void segment then L'_p is just the single edge (u_p, u_{p+1}), taken from $E = E(G)$. Observe that for a void segment,

$$\omega(\hat{L}_p) = \omega(L_p) = \omega(u_p, u_{p+1}) = d_G(u_p, u_{p+1}) .$$

Otherwise, if L_p is not a void segment, then $d_G(u_p, v_p) \le \frac{1}{2} \cdot \alpha \cdot (1/\epsilon)^i$. Observe also that since all vertices of $\pi(u, v)$ are $\hat{\mathcal{U}}^{(i)}$-clustered, this is also the case for the subpath $\pi(u_p, v_p)$. Hence the induction hypothesis is applicable to this subpath, and so there exists a path $\pi'(u_p, v_p)$ in $G \cup H$ with at most h_i hops, such that

$$\omega(\pi'(u_p, v_p)) \le d_G(u_p, v_p) \cdot (1 + 16c(i-1)\epsilon) + 8\alpha c(1/\epsilon)^{i-1} .$$

We define L'_p to be the concatenation of $\pi'(u_p, v_p)$ with the edge (v_p, u_{p+1}). (This edge is taken from G.) Since v_p lies on a shortest path between u_p and u_{p+1}, it follows that

$$\omega(L'_p) \le (1 + 16c(i-1)\epsilon) \cdot d_G(u_p, u_{p+1}) + 8\alpha c(1/\epsilon)^{i-1} ,$$

and L'_p contains up to $h_i + 1$ hops.

Finally, our ultimate path $\pi'(u, v)$ is the concatenation of all the substitute segments $L'_1 \circ L'_2 \circ \ldots \circ L'_q$, where $\pi(u, v) = \hat{L}_1 \circ \hat{L}_2 \circ \ldots \circ \hat{L}_q$. Since each extended segment has length at least $\frac{1}{2}\alpha \cdot (1/\epsilon)^{i-1}$ we conclude that

$$\begin{aligned} & d_{G\cup H}^{((h_i+1)\cdot\lceil\frac{d_G(u,v)}{\frac{1}{2}\alpha(1/\epsilon)^i}\rceil)}(u,v) \\ \leq\ & d_G(u,v)(1+16c(i-1)\epsilon) + \left\lceil \frac{d_G(u,v)}{\frac{1}{2}\alpha(1/\epsilon)^i} \right\rceil \cdot \frac{8\alpha c}{\epsilon^{i-1}} \\ \leq\ & d_G(u,v)\left(1+16c(i-1)\epsilon + \frac{8\alpha c/\epsilon^{i-1}}{\frac{1}{2}\alpha/\epsilon^i}\right) + \frac{8\alpha c}{\epsilon^{i-1}} \\ =\ & d_G(u,v)(1+16ci\epsilon) + \frac{8\alpha c}{\epsilon^{i-1}} \ . \qquad (7) \end{aligned}$$

Now consider x, y such that $d_G(x,y) \leq \frac{1}{2}\alpha \cdot (1/\epsilon)^{i+1}$ and such that $\pi(x,y)$ is $\hat{\mathcal{U}}^{(i+1)}$-clustered. Let z_1 and z_2 denote the leftmost and the rightmost $\hat{\mathcal{U}}_{i+1}$-clustered vertices on this path, and denote by C_1 and C_2 their respective clusters. Denote also $r_1 = r_{C_1}, r_2 = r_{C_2}$. (If z_1 and z_2 do not exist, then $\pi(x,y)$ is $\hat{\mathcal{U}}^{(i)}$-clustered, and this case was already taken care of.) Denote also by w_1 (resp., w_2) the neighbor of z_1 (resp., z_2) on the subpath $\pi(x,z_1)$ (resp., $\pi(z_2,y)$) of $\pi(x,y)$.

The path $\pi'(x,y)$ in $G\cup H$ between x and y is constructed in the following way. By (7), we can reach from x to w_1 while incurring a multiplicative stretch of $(1+16ci\cdot\epsilon)$ and an additive error of $8\cdot\alpha\cdot c\cdot(1/\epsilon)^{i-1}$, and using at most $b_1 = (h_i+1)\cdot\lceil\frac{d_G(x,w_1)}{\frac{1}{2}\alpha(1/\epsilon)^i}\rceil$ hops. The same is true for the pair w_2, y, except that the required number of hops is at most $b_2 = (h_i+1)\cdot\lceil\frac{d_G(w_2,y)}{\frac{1}{2}\alpha(1/\epsilon)^i}\rceil$. Finally, the path $\pi'(x,y)$ connects w_1 with w_2 via edges $(w_1,z_1),(z_2,w_2)$ that belong to $E(G)$, the edge (r_1,r_2) of the hopset H, and the paths $\pi(z_1,r_1),\pi(r_2,z_2)$ (each of $i+1$ hops) in H given by Claim III.1. Hence

$$\begin{aligned} & d_{G\cup H}^{(h_{i+1})}(x,y) \\ \leq\ & d_{G\cup H}^{(b_1)}(x,w_1) + d_G^{(1)}(w_1,z_1) + d_G^{(i+1)}(z_1,r_1) \\ & + d_H^{(1)}(r_1,r_2) + d_G^{(i+1)}(r_2,z_2) + d_G^{(1)}(z_2,w_2) \\ & + d_{G\cup H}^{(b_2)}(w_2,y) \\ \leq\ & (1+16ci\epsilon)d_G(x,w_1) + d_G(w_1,z_1) + R_{i+1} \\ & + (d_G(z_1,z_2)+2R_{i+1}) + R_{i+1} + d_G(z_2,w_2) \\ & + (1+16ci\epsilon)d_G(w_2,y) + 2(8\alpha c(1/\epsilon)^{i-1}) \\ \leq\ & (1+16ci\epsilon)d_G(x,y) + 4\alpha c(1/\epsilon)^i + 16\alpha c(1/\epsilon)^{i-1} \\ \leq\ & (1+16ci\epsilon)d_G(x,y) + 8\alpha c(1/\epsilon)^i \ , \end{aligned}$$

where the required number of hops indeed satisfies

$$\begin{aligned} & (h_i+1)\left(\lceil\frac{d_G(x,w_1)}{\frac{1}{2}\alpha\cdot(1/\epsilon)^i}\rceil + \lceil\frac{d_G(w_2,y)}{\frac{1}{2}\alpha\cdot(1/\epsilon)^i}\rceil\right) + 2i+5 \\ \leq\ & (h_i+1)\left(\frac{d_G(x,y)}{\frac{1}{2}\alpha(1/\epsilon)^i} + 2\right) + 2i + 5 \\ \leq\ & (h_i+1)(1/\epsilon+2) + 2i + 5 \\ =\ & h_{i+1} \ . \qquad (8) \end{aligned}$$

■

The recursive equation $h_{i+1} = (h_i+1)(1/\epsilon+2)+2i+5$ solves to $h_i \leq 3\cdot(1/\epsilon+2)^i$, for $\epsilon < 1/10$, i.e., $h_\ell \leq 3\cdot(1/\epsilon+2)^\ell$. Write $\zeta = 16c(\ell+1)\cdot\epsilon$ and $\beta = 2h_\ell + 1 \leq 6\cdot(1/\epsilon+2)^\ell + 1$.

Corollary III.5. *Let $x, y \in V$ be such that $d_G(x,y) \in (\hat{R}/2, \hat{R}]$. Then*

$$d_{G\cup H}^{(\beta)}(x,y) \leq (1+\zeta)\cdot d_G(x,y) \ .$$

Proof: Let $\pi(x,y)$ be the shortest path in G between x, y. By a similar (and simpler) argument to the one appearing in Lemma III.4, one can see that there exists an edge $(u,v) \in E$ such that both u, v are on $\pi(x,y)$, and also $d_G(x,u) \leq \hat{R}/2$ and $d_G(y,v) \leq \hat{R}/2$. Applying Lemma III.4 on these pairs with $i = \ell$, recalling that $\frac{1}{2}\alpha\cdot(1/\epsilon)^\ell = \hat{R}/2$ and that every vertex is clustered in $\hat{\mathcal{U}}^{(\ell)}$, it follows that

$$d_{G\cup H}^{(h_\ell)}(x,u) \leq d_G(x,u)(1+16c(\ell-1)\cdot\epsilon) + 8c\cdot\epsilon\cdot\hat{R} \ .$$

Similarly also

$$d_{G\cup H}^{(h_\ell)}(y,v) \leq d_G(x,u)(1+16c(\ell-1)\cdot\epsilon) + 8c\cdot\epsilon\cdot\hat{R} \ .$$

Since $\beta = 2h_\ell + 1$ and $(u,v) \in E$, we obtain

$$\begin{aligned} & d_{G\cup H}^{(\beta)}(x,y) \\ \leq\ & d_{G\cup H}^{(h_\ell)}(x,u) + d_G^{(1)}(u,v) + d_{G\cup H}^{(h_\ell)}(y,v) \\ \leq\ & (d_G(x,u)+d_G(u,v)+d_G(v,y))(1+16c(\ell-1)\epsilon) \\ & +16c\epsilon\hat{R} \\ \leq\ & d_G(x,y)\cdot(1+16c(\ell-1)\cdot\epsilon) + 32c\cdot\epsilon\cdot d_G(x,y) \\ =\ & d_G(x,y)\cdot(1+\zeta) \ . \end{aligned}$$

■

Recall that $\ell = \lfloor\log(\kappa\rho)\rfloor + \lceil\frac{\kappa+1}{\rho\kappa}\rceil - 1 \leq \log(\kappa\rho) + \lceil 1/\rho\rceil$ is the number of phases of the algorithm (for the sake of brevity, from now on we shall ignore the ceiling of $1/\rho$). When we rescale $\epsilon = \zeta$ as the strech factor then $\beta = O(\ell/\epsilon)^\ell = O\left(\frac{\log\kappa+1/\rho}{\epsilon}\right)^{\log\kappa+1/\rho}$.

Our ultimate hopset H is created by $H \leftarrow \bigcup_{k>\log\beta-1} H_k$, i.e., H is the union of up to $\lceil\log\Lambda\rceil$ hopsets, each of which takes care of its own distance range. As a result, the number of edges in H is $O(n^{1+1/\kappa}\cdot\log\Lambda)$, and its expected construction time is $O((|E|+n\log n)\cdot n^\rho/\rho\cdot\log\Lambda)$. In the full version of the paper we show how to remove the dependence on the aspect ratio Λ, and replace it with n, which yields the following result.

Theorem III.6. *For any graph $G = (V,E,\omega)$ with n vertices, $2 \leq \kappa \leq (\log n)/4$, $1/2 > \rho \geq 1/\kappa$, and $0 < \epsilon \leq 1$, our algorithm constructs a (β,ϵ)-hopset H with $O(n^{1+1/\kappa}\cdot\log n)$ edges in expectation, in time $O((|E| + n\log n)(n^\rho/\rho\cdot\log n))$, with $\beta = O\left(\frac{\log\kappa+1/\rho}{\epsilon}\right)^{\log\kappa+1/\rho}$.*

Finally, we note that our assumption that $\rho > \log\log n/(2\log n)$ is justified, as otherwise we get $\beta \geq n$, in which case an empty hopset will do. Also $\epsilon \leq 1$, because we rescaled it by a factor of $16c(\ell + 1) > 10$.

Acknowledgments

M.E. was supported by the ISF grant No. (724/15). O.N was supported in part by ISF grant No. (523/12), by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement $n°303809$, and by the BSF-NSF grant No. 2015813.

We are grateful to anonymous reviewers of FOCS'16 for comments, that helped improving the presentation.

References

[1] Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical report, 1987.

[2] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear cost sequential and distributed constructions of sparse neighborhood covers. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 638–647, 1993.

[3] Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 503–513, 1990.

[4] 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.

[5] T-H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 70–78, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[6] Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*, pages 648–658, 1993.

[7] Edith Cohen. Using selective path-doubling for parallel shortest-path computations. *J. Algorithms*, 22(1):30–56, 1997.

[8] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000.

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

[10] Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. *CoRR*, abs/1605.04538, 2016.

[11] Michael Elkin and Ofer Neiman. On efficient distributed construction of near optimal routing schemes, 2016. Accepted to PODC'16.

[12] Michael Elkin and David Peleg. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631, 2004.

[13] 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.

[14] 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.

[15] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. An almost-tight distributed algorithm for computing single-source shortest paths. 2016. To appear in STOC'16.

[16] Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *J. Algorithms*, 25(2):205–220, 1997.

[17] Christoph Lenzen and Boaz Patt-Shamir. Fast partial distance estimation and applications. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 153–162, 2015.

[18] 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.

[19] 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.

[20] Hanmao Shi and Thomas H. Spencer. Time-work tradeoffs of the single-source shortest paths problem. *J. Algorithms*, 30(1):19–32, 1999.

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

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

[23] Jeffrey D. Ullman and Mihalis Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991.

[24] Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 128–136, 1982.