The Distributed Computing Column

BY

Stefan Schmid

University of Vienna
Währinger Strasse 29, AT - 1090 Vienna, Austria
schmiste@gmail.com

Near-Additive Spanners and Near-Exact Hopsets, A Unified View

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

Given an *unweighted* undirected graph $G = (V, E)$, and a pair of parameters $\epsilon > 0$, $\beta = 1, 2, \ldots$, a subgraph $G' = (V, H)$, $H \subseteq E$, of G is a $(1+\epsilon, \beta)$-*spanner* (aka, a *near-additive spanner*) of G if for every $u, v \in V$,

$$d_{G'}(u, v) \leq (1+\epsilon) d_G(u, v) + \beta \, .$$

It was shown in [25] that for any n-vertex G as above, and any $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, there exists a $(1+\epsilon, \beta)$-spanner G' with $O_{\epsilon,\kappa}(n^{1+1/\kappa})$ edges, with

$$\beta = \beta_{EP} = \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa - 2} \, .$$

This bound remains state-of-the-art, and its dependence on ϵ (for the case of small κ) was shown to be tight in [3].

Given a *weighted* undirected graph $G = (V, E, \omega)$, and a pair of parameters $\epsilon > 0$, $\beta = 1, 2, \ldots$, a graph $G' = (V, H, \omega')$ is a $(1+\epsilon, \beta)$-*hopset* (aka, a *near-exact hopset*) of G if for every $u, v \in V$,

$$d_G(u, v) \leq d^{(\beta)}_{G \cup G'}(u, v) \leq (1+\epsilon) d_G(u, v) \, ,$$

where $d^{(\beta)}_{G \cup G'}(u, v)$ stands for a β-(hop)-bounded distance between u and v in the union graph $G \cup G'$. It was shown in [22] that for any n-vertex G and ϵ and κ as above, there exists a $(1+\epsilon, \beta)$-hopset with $\tilde{O}(n^{1+1/\kappa})$ edges, with $\beta = \beta_{EP}$.

Not only the two results of [25] and [22] are strikingly similar, but so are also their proof techniques. Moreover, Thorup-Zwick's later construction of near-additive spanners [41] was also shown in [24, 29] to provide hopsets with analogous (to that of [41]) properties.

In this survey we explore this intriguing phenomenon, sketch the basic proof techniques used for these results, and highlight open questions.

1 Introduction

1.1 Spanners

Given an undirected unweighted n-vertex graph $G = (V, E)$, and a pair of paremeters $\alpha \geq 1, \beta \geq 0$, a subgraph $G' = (V, H)$, $H \subseteq E$, of G is called an (α, β)-*spanner* of G, if for every pair $u, v \in V$ of vertices, we have $d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta$. Here d_G (respectively, d_H) stands for the distance in G (respectively, in H). If $\beta = 0$, the spanner is called *multiplicative*, and if $\alpha = 1$, the spanner is called *additive*. A graph $G' = (V, H, \omega)$ (possibly weighted, even when the original graph $G = (V, E)$ is unweighted) is called an (α, β)-*emulator* of G, if for every pair $u, v \in V$, we have $d_G(u, w) \leq d_H(u, v) \leq \alpha d_G(u, v) + \beta$.

Althofer et al. [6], improving upon an earlier work by Peleg and Schaeffer [37], showed that for every n-vertex undirected (possibly weighted) graph $G = (V, E)$, and any parameter $\kappa = 1, 2, \ldots$, there exists a $(2\kappa - 1)$-spanner with at most $n^{1+1/\kappa}$ edges. This bound is known to be tight under Erdos' girth conjecture (see, e.g., [40], Section 5), and is unconditionally tight up to a leading constant coefficient in the stretch.

A large body of literature exploring constructions of spanners in various computational settings was developed throughout the years [2, 16, 5, 31, 18, 25, 20, 11, 26, 41, 34, 35, 36, 43, 9, 17, 38, 13, 1, 3, 23, 21]. Algorithms for constructing purely additive spanners were given in [5, 18, 25, 9, 13]. Specifically, Aingworth et al. and Dor et al. [5, 18] devised an algorithm constructing additive 2-spanners with $\tilde{O}(n^{3/2})$ edges, and additive 4-emulators with $\tilde{O}(n^{4/3})$ edges. Elkin and Peleg [25] shaved polylogarithmic factors from these size bounds via different constructions; their size bounds are $O(n^{3/2})$ and $O(n^{4/3})$, respectively. Baswana et al. [9] devised a construction of additive 6-spanners with $O(n^{4/3})$ edges, and Chechik [13] devised a construction of additive 4-spanners with $\tilde{O}(n^{7/5})$ edges.

In [25] Elkin and Peleg also devised the first construction of *near-additive* spanners, i.e., $(1 + \epsilon, \beta)$-spanners, which are together with near-exact hopsets, constitute the main topic of the current survey. Specifically, they showed that for any $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, there exists $\beta(\epsilon, \kappa)$ (denoted also β_{EP}) such that for any n-vertex unweighted undirected graph there exists a $(1 + \epsilon, \beta)$-spanner with $O_{\epsilon,\kappa}(n^{1+1/\kappa})$ edges.

Note that in this result, unlike in the aforementioned tradeoff for multiplicative spanners [37, 6], both the multiplicative stretch $1 + \epsilon$ and the exponent of the number of edges $1 + 1/\kappa$ can *simultaneously* be made as small as one wishes, at the expense of increasing the additive error term β. This additive term behaves as

$$\beta_{EP}(\epsilon, \kappa) = \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa - 2},$$

and it is still the state-of-the-art bound.

Observe also that this result means also that distances larger than some constant threshold can be approximated arbitrarily well using arbitrarily sparse spanners. The threshold increases, of course, as the approximation factor and the exponent of the number of edges decrease.

At the beginning near-additive spanners were often viewed as a stepping stone towards the "real thing", that is, purely additive spanners. However, Abboud and Bodwin [1], relying on earlier lower bounds for distance preservers [12], showed that one cannot in general have purely additive spanners with constant (or even polylogarithmic) error term β and size $o(n^{4/3})$. Therefore, near-additive spanners is the *best one can hope for*!

Near-additive spanners were intensively studied in the last two decades [25, 20, 26, 41, 34, 35, 36, 23, 4, 21]. In [20] Elkin devised the first efficient algorithm for constructing them. This algorithm provides $(1+\epsilon,\beta)$-spanners with $\tilde{O}_{\epsilon,\kappa,\rho}(n^{1+1/\kappa})$ edges in centralized time $O(|E|\cdot n^{\rho})$, with $\beta_E = (\kappa/\epsilon)^{O(\frac{\log\kappa}{\rho})}$, where $\rho > 0$ is an additional parameter that controls the running time. Improved variants of this algorithm, as well as efficient implementations of it in distributed and streaming settings, were devised in [26]. Both these algorithms [20, 26] build upon ideas from a seminal algorithm of Cohen [15] for constructing hopsets. (See more about it in Section 1.2.)

Another remarkable algorithm for constructing near-additive spanners and emulators was devised by Thorup and Zwick [41]. The main feature of their construction is that it provides a *universal* near-additive spanner, i.e., the same spanner applies *simultaneously* for all values of $\epsilon > 0$. Putting it differently, their algorithm accepts as input an n-vertex graph $G = (V, E)$ and a parameter $\kappa = 2, 3, \ldots$, (but it does *not* accept ϵ as a part of the input), and it constructs for it a spanner $G' = (V, H)$, $H \subseteq E$, $|H| = O_{\kappa}(n^{1+1/\kappa})$, which constitutes a $(1+\epsilon,\beta(\epsilon,\kappa))$-spanner for all $\epsilon > 0$ simultaneously. The additive term in their construction behaves as $\beta = \beta_{TZ} = O\left(\frac{\kappa}{\epsilon}\right)^{\kappa-1}$, i.e., it is much higher than β_{EP}. On the other hand, they have also devised a *universal emulator* whose additive term is the same as in the *spanner* of [25].

Interestingly, the universality of Thorup-Zwick's construction enables one to obtain spanners and emulators with a *sublinear* additive error term. For concreteness, let us consider a pair of vertices $u, v \in V$ with $d_G(u,v) = d$. We can set

$$\epsilon = \frac{\log\kappa}{d^{\frac{1}{\log\kappa-1}}} .$$

Then we have

$$d_H(u,v) \leq d(1+\epsilon) + \left(\frac{\log\kappa}{\epsilon}\right)^{\log\kappa-2} \leq d + O\left(\log\kappa \cdot d^{1-\frac{1}{\log\kappa-1}}\right) .$$

Note that the additive error $O\left(\log\kappa \cdot d^{1-\frac{1}{\log\kappa-1}}\right)$ is sublinear in the original distance $d = d_G(u,v)$, and this property holds for all pair of vertices.

Pettie [34, 35, 36] improved the construction of universal spanners of [41]. His algorithm constructs universal $(1+\epsilon,\beta)$-spanners with $O_{\kappa}(n^{1+1/\kappa})$ edges and

$$\beta = \beta_{Pet} = \beta_{EP}^{\log_{4/3}2} \approx \beta_{EP}^{2.41} .$$

Devising universal spanners with additive error that matches the additive error of spanners of [25] (i.e., closing the gap between β_{Pet} and β_{EP}) is an open problem. The algorithm of Pettie [34, 35, 36] is based on a combination of Thorup-Zwick's construction of emulators with a construction of distance preservers from [12].

Finally, Abboud et al. [4] showed a lower bound on the size-stretch tradeoff of near-additive spanners. They showed that any construction of $(1+\epsilon,\beta)$-spanners with $O(n^{1+1/\kappa})$ edges must have

$$\beta_{ABP} = \Omega\left(\frac{1}{\epsilon\cdot\log\kappa}\right)^{\log\kappa-2} .$$

Note that while this lower bound is tight for a very small $\epsilon > 0$ and constant κ, it is meaningless when $\epsilon \geq \frac{1}{\log\kappa}$. So, in particular, it is wide open if one can achieve near-additive spanners with β

that depends *polynomially* on κ. (The state-of-the-art dependence is $\beta_{EP} = (\log \kappa)^{\log \kappa} = \kappa^{\log\log\kappa}$, i.e., it is slightly superpolynomial in κ.)

We note that if one allows a larger but still constant multiplicative stretch, then $(O(1),\beta)$-spanners with $\tilde{O}_\kappa(n^{1+1/\kappa})$ edges with $\beta = poly(\kappa)$ were devised by Pettie [34]. Improved and generalized bounds along these lines were given in [19, 10].

1.2 Near-Exact Hopsets

Given an undirected weighted n-vertex graph $G = (V, E, \omega)$, and a pair of parameters $\alpha \geq 1$ and $\beta = 1, 2, \ldots$, a graph $G' = (V, H, \omega')$, $H \cap E = \emptyset$, is called an (α,β)-*hopset* of G if for every pair $u, v \in V$ of vertices we have

$$d_G(u, v) \leq d^{(\beta)}_{G\cup G'}(u, v) \leq \alpha \cdot d_G(u, v) \ .$$

Here $\tilde{G} = G \cup G'$ is the union graph of G and G', i.e., $G \cup G' = (V, E \cup H, \tilde{\omega})$, where for every edge $e \in E$, $\tilde{\omega}(e) = \omega(e)$, and for every $e \in H$, $\tilde{\omega}(e) = \omega'(e)$. Also, $d^{(\beta)}_{\tilde{G}}$ stands for a β-*bounded* distance function in $\tilde{G}$, i.e., $d^{(\beta)}_{\tilde{G}}(u, v)$ is the length of the shortest $u - v$ path in $\tilde{G}$ that contains at most β hops. The parameter β is called the *hopbound* of the hopset G', and α is called the *stretch* of the hopset.

Hopsets turn our to be extremely useful for exact and approximate distance-related computations in distributed, dynamic, parallel and streaming settings. They also constitute fascinating combinatorial objects of independent interest.

Exact hopsets (i.e., hopsets with $\alpha = 1$) were implicitly studied by Ullman and Yannakakis [42], by Klein and Sairam [32] and by Shi and Spencer [39]. *Near-exact* hopsets, i.e., hopsets with $\alpha = 1 + \epsilon$, for an arbitrarily small $\epsilon > 0$, were introduced in a seminal paper by Cohen [15]. For an input undirected possibly weighted n-vertex graph, Cohen's algorithm constructs $(1 + \epsilon,\beta)$-hopsets of size $\tilde{O}(n^{1+1/\kappa})$, and with a polylogarithmic hopbound β. Specifically,

$$\beta_{Coh} = O\left(\frac{\log n}{\epsilon}\right)^{O(\log\kappa)} \ .$$

Additional constructions of near-exact hopsets were given by Bernstein [8], Henzinger et al. [27], and by Miller et al. [33]. The hopset of [8] has hopbound $O(\log n \cdot (1/\epsilon)^\kappa)$, and size $\tilde{O}(\kappa \cdot n^{1+1/\kappa} \cdot \log \Lambda)$, where Λ is the aspect ratio of the graph.[1] The hopsets of Henzinger et al. [27, 28] have hopbound $exp\{\tilde{O}_\epsilon(\sqrt{\log n \cdot \log\log n})\}$ and size $n \cdot exp\{\tilde{O}_\epsilon(\sqrt{\log n \cdot \log\log n})\} \cdot \log^{O(1)} \Lambda$. The hopsets of [33] have hopbound n^γ, for an arbitrarily small constant $\gamma > 0$, and size $O(n)$.

The first construction of hopsets with *constant* hopbound (and non-trivial size guarantee) was given by the current authors in [22]. Specifically, we showed there that for any pair of parameters $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, there exists $\beta = \beta(\epsilon, \kappa) = \beta_{EP}$, such that for every undirected weighted n-vertex graph $G = (V, E, \omega)$, there exists a $(1 + \epsilon,\beta)$-hopset with $O(n^{1+1/\kappa} \cdot \log n)$ edges. Note the striking similarity between this result and the result of Elkin and Peleg [25] concerning near-additive spanners. Remarkably, not just the results are similar, but also their proofs are closely related. We will elaborate on this relationship below.

Interestingly, the same phenomenon occurs for the Thorup-Zwick's construction [41] of near-additive spanners and emulators as well. In [24, 29] it was shown that Thorup-Zwick's

[1] The *aspect ratio* Λ of a weighted graph $G = (V, E, \omega)$ is the ratio betwen $\max_{u,v} d_G(u, v)$ and $\min_{u\neq v} d_G(u, v)$.

construction not only gives rise to universal near-additive emulators, but also to *universal* hopsets. Specifically, the algorithm of [24, 29, 41], given an input n-vertex graph and a parameter κ, constructs a hopset of size $O(n^{1+1/\kappa})$, which serves as a $(1+\epsilon,\beta)$-hopset with $\beta = \beta_{EP}(\epsilon,\kappa) = \left(\frac{\log\kappa}{\epsilon}\right)^{\log\kappa-2}$, *simultaneously* for all $\epsilon > 0$.

1.3 Discussion

With these results in mind, it is natural to wonder why are near-additive spanners and near-exact hopsets that similar? After all, there are some apparent significant differences. First, near-additive spanners apply to unweighted graphs[2], while near-exact hopsets apply to weighted graphs. Second, the meaning of the parameter β is very different. For spanners it is the additive error term, while for hopsets it is the hopbound. Third, spanners are subgraphs of the input graph, while hopsets are sets of edges that are not present in the original graph. (This distinction becomes blurred if one considers emulators instead of spanners. Nevertheless, an emulator, like a spanner, is used on its own, while hopset is used together with the edges of the original graph.)

We will next shortly discuss these techniques for spanners and hopsets. In the technical part of this survey we will sketch proofs of these results, and highlight the similarities and differences between them.

As was discussed above, there are three main approaches to building near-additive spanners and near-exact hopsets. The first one is the superclustering and interconnection approach, which was introduced by [25] in the context of near-additive spanners, and used in [22] for constructing hopsets. The second one, closely related to the first one, is the universal extension of the superclustering and interconnection approach. It was introduced by [41] in the context of spanners and emulators, and used in [24, 29] in the context of hopsets. The third one, based on neighborhood covers, was originated in Cohen's construction of hopsets [15]. It was later used in [20, 26] for building near-additive spanners. Because of space considerations, we will focus on the first two approaches in this survey.

The superclustering and interconnection approach, which we describe in detail in Section 2, works roughly as follows. It proceeds for $\ell = \log\kappa$ phases, indexed $0, 1, \ldots, \ell-1$. (Throughout the survey, we assume, for simplicity, that $\log\kappa = \log_2\kappa$ is an integer. This has only a very minor impact on the cited bounds.) On phase 0, its input partition $\mathcal{P}_0 = \{\{v\} \mid v \in V\}$ is the collection of singletons. One uses a sequence of degree thresholds, $deg_0, deg_1, \ldots, deg_{\ell-1}$, the simplest of which is $deg_i = n^{\frac{2^i}{\kappa}}$ [25], and a sequence of distance thresholds $\delta_i = (1/\epsilon)^i$. Each cluster C of the input partition $\mathcal{P}_i$ that has at least deg_i other clusters of $\mathcal{P}_i$ at distance at most δ_i from it, becomes *superclustered*, i.e., merged into a next-level cluster, a cluster of $\mathcal{P}_{i+1}$. Spanning trees of superclusters are added into the spanner/hopset. (On phase $\ell-1$, the superclustering step is skipped, and the algorithm proceeds directly to the interconnection step.)

In [25], this is done directly. Specifically, one initializes the set $\mathcal{U}_i$ of uncovered clusters as $\mathcal{P}_i$. Then one iteratively finds such clusters $C \in \mathcal{P}_i$ with many uncovered nearby clusters, creating superclusters around them, and marking them as covered. For reasons of efficiency and parallelism, in [22, 23], this is done by sampling clusters of $\mathcal{P}_i$ with probability $\frac{1}{deg_i}$, and creating superclusters around the sampled clusters. In [21] the same step is performed by computing ruling sets.

At any rate, once we are done with superclustering, we move to the interconnection step. On

[2] Even though there are some results [20, 19] about weighted graphs as well.

this step pairs of clusters are interconnected, i.e., shortest paths between them (or direct edges, in the case of hopsets/emulators) are inserted into the spanner (respectively, hopset/emulator).

The stretch analysis of this construction considers a pair $u, v \in V$ of vertices, and a shortest path $\pi(u, v)$ between them. The path is partitioned into segments of length $\delta_{\ell-1} = (1/\epsilon)^{\ell-1}$. On each such a segment $x-y$ one identifies the leftmost and the rightmost $\mathcal{P}_{\ell-1}$ clusters C_L and C_R. The substitute spanner's path $\pi'(x, y)$ that the stretch analysis finds uses a direct $C_L - C_R$ shortest path. The latter was inserted into the spanner on the $(\ell-1)$st phase, because $d_G(C_L, C_R) \leq \delta_{\ell-1}$. Then the stretch analysis zooms in into the $x - C_L$ subpath, and into the $C_R - y$ subpath. Both these subpaths are free of $\mathcal{P}_{\ell-1}$ clusters, and the stretch analysis exploits this to provide small-stretch substitute spanner's paths for them.

We note that the radii of C_L and C_R are both $O((1/\epsilon)^{\ell-2})$, i.e., by one order of magnitude (that is, by a factor of $1/\epsilon$) smaller than the length of the segment $x-y$. The same phenomenon occurs on lower levels of stretch analysis as well, i.e., in segments of the subpaths $x - C_L$ and $C_R - y$. These segments are of length $(1/\epsilon)^{\ell-2}$, while the maximum radius of a cluster that appears on these segments is $O((1/\epsilon)^{\ell-3})$, etc. Hence, roughly speaking, we have multiplicative stretch of $1+\epsilon$ on every level of stretch analysis, and thus the overall stretch is $1 + O(\epsilon \cdot \ell)$. The additive error term stems from the fact that $d_G(u, v)$ might be shorter than $\delta_{\ell-1} = (1/\epsilon)^{\ell-1}$. In this case one would not be able to charge the radii of C_L and C_R to the length of the segment $x - y$, and the additive term accounts for that.

The construction of hopsets that employs the superclustering and interconnection method [22] proceeds along very similar lines. In its simplest form it builds a separate hopset H_j for each distance scale $[2^j, 2^{j+1})$, for $j = 0, 1, \ldots, \lceil \log \Lambda \rceil$. For each fixed j, the hopset H_j takes care of pairs $u, v \in V$ of vertices with $d_G(u, v) \in [2^j, 2^{j+1})$. The ultimate hopset is $H = \bigcup_{j=0}^{\lceil \log \Lambda \rceil} H_j$.

We then define a distance unit $\gamma = \frac{2^j}{(1/\epsilon)^{\ell-1}} = \epsilon^{\ell-1} 2^j$. (As we aim at hopbound of $(1/\epsilon)^{\ell-1}$, one can assume that $2^j \geq (1/\epsilon)^{\ell-1}$.) Then the distance thresholds δ_i are defined as $\gamma \cdot (1/\epsilon)^i$, i.e., in the same way as for near-additive spanners, except for scaling by a factor of γ. We then conduct the same superclustering and interconnection algorithm as for the spanner's construction, with the same degree thresholds, and distance thresholds δ_i as above. The only difference is that instead of inserting shortest paths between pairs of vertices z, z' into the spanner, here we insert direct hopset edges (z, z') of weight $d_G(z, z')$. (This also happens in the construction of emulators.)

The stretch analysis of the resulting hopset is also conducted very similarly to the case of spanners. There are some technicalities that have to do with the fact that we deal with weighted graphs in the case of hopsets, and thus we may not be able to partition the shortest path into segments of length precisely $\delta_{\ell-1} = (1/\epsilon)^{\ell-1}\gamma$. However, one can easily overcome these difficulties. (See Section 2.1.2.) Another difference is that one can use edges of the original graph in the substitute path in the case of hopsets, while this is not the case for spanners. This actually makes the stretch analysis easier in the former case. Finally, in the case of hopsets one also needs to carefully analyze the number of hops used in the substitute path. Intuitively, the shortest path $\pi(u, v)$ is partitioned to $\approx (1/\epsilon)^{\ell-1}$ subsegments of length γ, and for each of them $O(1)$ hops suffice. Thus, the hopbound is, up to rescaling of ϵ, equal to $O((1/\epsilon)^{\ell-1})$.

Next we overview the construction of Thorup-Zwick's emulators [41] and hopsets of [24, 29], while focusing on their relationship to the superclustering and interconnection method. As was already mentioned, these emulators and hopsets can be viewed as a scale-free version of spanners and hopsets from [25, 22].

the algorithm of [41] works as follows. It defines $A_0 = V$, and for $i = 0, 1, \ldots, \ell-1$, vertices

of A_{i+1} are obtained from those of A_i by sampling each $v \in A_i$ independently with probability $\frac{1}{deg_i}$. The sequence $deg_0, deg_1, \ldots, deg_{\ell-1}$ of degree thresholds is defined exactly as in [25].

Given this hierarchy of subsets $V = A_0 \supseteq A_1 \supseteq \ldots \supseteq A_{\ell-1}$, the algorithm defines for every vertex $v \in A_i$, $i < \ell - 1$, its *pivot* $p(v)$ to be the closest A_{i+1}-vertex to v. (Ties are broken arbitrarily but consistently.) We also define the *bunch* of $v \in A_i$ by

$$Bunch(v) = \{u \in A_i \mid d_G(v,u) < d_G(v, A_{i+1})\} .$$

It is the set of all vertices of A_i that are closer to v than the pivot of v. For any $v \in A_{\ell-1}$, its bunch is defined as the entire $A_{\ell-1}$.

The algorithm then inserts into the emulator H (and into the hopset) the edges $\bigcup_{i=0}^{\ell-1}\{(v,u) \mid v \in A_i, u \in Bunch(v)\}$, and also the edges $\bigcup_{i=0}^{\ell-2}\{(v,p(v))\}$. This completes the description of the construction. Intuitively, the edges $\bigcup_{i=0}^{\ell-2}\{(v,p(v))\}$ are *superclustering* edges, i.e., edges that connect an i-level cluster center to its $(i+1)$st level parent. The edges of $\bigcup_{i=0}^{\ell-1}\{(v,u) \mid v \in A_i, u \in Bunch(v)\}$ are *interconnection* edges, i.e., edges that connect pairs of cluster centers of the same level.

For the stretch analysis, we consider a pair $u, v \in V$ of vertices, at distance $d = d_G(u,v)$ from one another. If we analyze H as an emulator, then we partition a shortest path $\pi(u,v)$ between them into segments of length $\delta_{\ell-1} = (1/\epsilon)^{\ell-1}$. (Recall than $\epsilon > 0$ is not a parameter of the algorithm in these scale-free constructions. Rather, it is a parameter of the *analysis*, which applies for any $\epsilon > 0$.) These segments are then partitioned into $1/\epsilon$ subsegments of length $\delta_{\ell-2}$, and those are again partitioned to $1/\epsilon$ subsegments, up until we reach single edges. (To analyze H as a hopset, we partition $\pi(u,v)$ of length d into $\approx 1/\epsilon$ segments of length $\approx d \cdot \epsilon$, and each of them into $\approx 1/\epsilon$ segments of length $\approx d \cdot \epsilon^2$, up until the bottom level, where each subsegment has length $\approx d \cdot \epsilon^{\ell-1}$.)

Now a segment $x - y$ of level i (i.e., of length δ_i) is called *successful*, if it admits a substitute path of length $1 + O(i \cdot \epsilon)$ between its endpoints in the emulator. For an unsuccessful segment, it can be argued that its endpoints x, y admit nearby $(i+1)$st pivots x', y', respectively. This is argued by an induction on i. The base case follows from definitions of pivots and bunches (with stretch 1). For the induction step, the analysis considers i-level subsegments of an $(i+1)$st level segment. If they are all successful, then we get a stretch of $1 + O(i \cdot \epsilon)$ for the entire segment, and we are done. Otherwise, we consider the leftmost and the rightmost unsuccessful subsegments $x_L - y_L$ and $x_R - y_R$. (The case that there is just one unsuccessful subsegment is even simpler. See Section 3.) By the induction hypothesis, there are $(i+1)$st level pivots z_L, z_R, with z_L being close to x_L and z_R to y_R. Now either $z_R \in Bunch(z_L)$, and so the edge (z_L, z_R) is in the emulator. We then obtain a short substitute emulator's $x - y$ path, that consists of the subpaths $x - x_L$, $x_L - z_L$, $z_L - z_R$, $z_R - y_R$, and finally, $y_R - y$. Otherwise, there is a nearby $(i+1)$nd pivot z to z_L, which, by triangle's inequality, is also close to x. This completes the inductive proof.

This inductive statement is used with $i = \ell - 1$. At this level all segments are successful, just because $A_\ell = \emptyset$. Hence the emulator provides stretch $1 + O(\epsilon \cdot \ell)$. In the case of hopsets one needs also to carefully count the number of hops, but other than that the stretch analysis proceeds along the same lines.

1.4 Organization

In Section 2 we describe the superclustering and interconnection approach in more detail. In Section 3 we do so for its scale-free extension.

2 Superclustering and Interconnection

This section is devoted to the superclusterig and interconnection method of constructing near-additive spanners and hopsets [25, 22, 21]. We start (Section 2.1) with describing the construction of near-additive spanners, and then proceed (Section 2.2) to hopsets.

2.1 Spanners

2.1.1 Algorithm

Let Q be the *ground* partition of the graph $G = (V, E)$, i.e., $Q = \{C_1, C_2, \ldots, C_q\}$, for some integer $q \geq 1$, is a collection of pairwise disjoint clusters such that $V = \bigcup_{C \in Q} C$. Moreover, each cluster $C \in Q$ has a designated center r_C, and the *radius* of the partition, i.e., the maximum radius of one of its clusters (with respect to its designated center) is

$$Rad(C) = \max_{u \in C}\{d_{G(C)}(r_C, u)\} \leq \kappa - 1 \ ,$$

for a parameter κ that controls the stretch-size tradeoff of the ultimate spanner.

The *supergraph* $\mathcal{G} = (Q, \mathcal{E})$ induced by the ground partition is defined by

$$\mathcal{E} = \{(C, C') \mid C \neq C', C, C' \in Q, \exists (v, v') \in E, v \in C, v' \in C'\} \ .$$

The ground partition Q has the property that the supergraph $\mathcal{G}$ is sparse, i.e., $|\mathcal{E}| = O(n^{1+1/\kappa})$.

Moreover, the ground partition Q has the property that $Q = \bigcup_{i=0}^{\kappa-1} Q_i$, where all clusters in Q_i have radius i, contain at least $n^{i/\kappa}$ vertices each, and have at most $n^{(i+1)/\kappa}$ "outgoing" neighbors (so that the total number of neighboring clusters is $O(n^{1+1/\kappa})$).

For convenience, we will assume that κ is of the form $\kappa = 2^\ell - 1$, for an integer ℓ. It is easy to adapt the constructions to the case of a general integer parameter κ. We partition the set of indices $\{0, 1, \ldots, \kappa - 1\}$ into subsets $\{0\}, \{1, 2\}, \{3, 4, 5, 6\}, \ldots, \{2^{\ell-1} - 1, 2^{\ell-1}, \ldots, 2^\ell - 2\}$. Let $\hat{Q}_0 = Q_0$, $\hat{Q}_1 = Q_1 \cup Q_2, \ldots$, and $\hat{Q}_{\ell-1} = Q_{2^{\ell-1}-1} \cup \ldots \cup Q_{2^\ell-2}$.

Constructions of such ground partitions are well-known, and can be found, e.g., in [37, 7, 30, 25]. We note that modern constructions of near-additive spanners [23, 21] that follow the superclustering and interconnection paradigm manage to bypass ground partitions altogether. However, the original construction of [25] that does use them is somewhat simpler.

The spanner H is initialized to contain the union of BFS spanning trees of all clusters C of the ground partition Q. For each cluster $C \in Q$, the BFS tree is rooted in its designated center r_C. We also insert into the spanner one edge $e = (u, v)$ for each pair of neighboring clusters $C, C' \in Q$ (i.e., e.g., $u \in C$, $v \in C'$). The overall number of edges inserted to the spanner so far is $O(n^{1+1/\kappa})$. (See [37, 7, 30, 25].)

The algorithm itself proceeds for ℓ phases. At the beginning of each phase i, $0 \leq i \leq \ell - 1$, we have the input partition $\mathcal{P}_i$ of clusters. For $i \leq \ell - 2$, the phase i *superclusters* some of these clusters into larger clusters (aka *superclusters*). The resulting partition $\hat{\mathcal{P}}_i$, union with the collection $\hat{Q}_{i+1}$ of clusters from the ground partition, forms the input for the next phase $i + 1$. (On phase 0, the input is $\mathcal{P}_0 = \hat{Q}_0 = Q_0$.) Some other clusters of $\mathcal{P}_i$ are not involved in superclustering. The set of these clusters is called $\mathcal{U}_i$, the set of unsuperclustered clusters of phase i. On the last phase $i = \ell - 1$, the superclustering step is omitted, and we define $\mathcal{U}_{\ell-1} = \mathcal{P}_{\ell-1}$.

On all phases $i = 0, 1, \ldots, \ell - 1$, the unsuperclustered clusters (the set $\mathcal{U}_i$) of this phase proceed to the *interconnection* step. On the interconnection step shortest paths between nearby clusters of $\mathcal{U}_i$ are added into the spanner H. An invariant of the algorithm is that each of the superclusters of $\mathcal{P}_i$ has size at least $n^{\frac{2^i-1}{\kappa}}$, for all $i = 0, 1, \ldots, \ell - 1$. At the beginning of phase i, the partition $\mathcal{P}_i$ is created as a union of $\hat{\mathcal{P}}_{i-1}$ (the output of phase $i-1$) with $\hat{\mathcal{Q}}_i$. Recall that each of the clusters of $\hat{\mathcal{Q}}_i$ has size at least $n^{\frac{2^i-1}{\kappa}}$ as well.

The algorithm also employs sequences of degree and distance thresholds. On each phase i, it uses the parameters $deg_i = n^{\frac{2^i}{\kappa}}$ as *degree threshold* and $\delta_i = (1/\epsilon)^i$ as *distance threshold*.

Next, we take a closer look on each particular phase $i = 0, 1, \ldots, \ell - 2$. (The last phase $i = \ell - 1$ is slightly different, as it has no superclustering step.) The algorithm checks if there exists a cluster $C \in \mathcal{P}_i$, such that at distance at most δ_i (in G) from C, there are at least deg_i uncovered clusters $C' \in \mathcal{P}_i$. (Initially all clusters of $\mathcal{P}_i$ are uncovered, i.e., $\mathcal{U}_i$ is initialized as $\mathcal{P}_i$.) If there is such a cluster C, then the algorithm creates a supercluster $\hat{C}$ around it, centered at C. This supercluster includes all the other uncovered clusters $C' \in \mathcal{P}_i$ at distance at most δ_i (in G) from C. Shortest paths between C and each of these clusters C' are added to the spanner H. Finally, C and all these clusters C' that are merged into $\hat{C}$ are removed from $\mathcal{U}_i$, i.e., they are marked as covered. Then the algorithm iterates. The resulting collection of superclusters $\hat{C}$ created in this way is the aforementioned set $\hat{\mathcal{P}}_i$. Together with $\hat{\mathcal{Q}}_{i+1}$ it constitutes the collection $\mathcal{P}_{i+1}$, which serves as input to phase $i+1$. This concludes the superclustering step of phase i.

On the interconnection step of phase i, each pair of clusters of $\mathcal{U}_i$ that are at distance at most δ_i from one another in G, are interconnected with one another via shortest paths. These shortest paths are added into the spanner H.

The last phase $i = \ell - 1$ is special, because the overall number of clusters in the input collection $\mathcal{P}_{\ell-1}$ is at most $O(n^{\frac{\kappa+1}{2\kappa}})$. Consequently, one interconnects all pairs of nearby clusters of $\mathcal{P}_{\ell-1}$ (i.e., clusters at distance at most $\delta_{\ell-1}$ from one another in G).

This concludes the description of the algorithm.

2.1.2 Analysis

We next sketch its analysis.

Size: We start with the size analysis. The number of edges inserted into H during the initialization step is, as was mentioned above, $O(n^{1+1/\kappa})$. This follows from properties of the ground partition [37].

Consider some fixed phase $i = 0, 1, \ldots, \ell - 1$. It can be easily seen inductively that each cluster $C \in \mathcal{P}_i$ has size at least $n^{\frac{2^i-1}{\kappa}}$. Since they are disjoint, we conclude that $|\mathcal{P}_i| \leq n^{1-\frac{2^i-1}{\kappa}}$. Recall that $deg_i = n^{\frac{2^i}{\kappa}}$. Hence the number of paths inserted into the spanner by the interconnection step of phase i is at most $|\mathcal{P}_i| \cdot deg_i \leq n^{1+1/\kappa}$. Each path contains at most δ_i edges (because we connect nearby clusters). Thus, the number of edges inserted on this step is $O(\delta_i \cdot n^{1+1/\kappa})$.

Consider the superclustering step (for $i < \ell - 1$). Let $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_p$ be the set of created superclusters. For each index $j \in [p]$, let C_j be the center cluster of $\hat{C}_j$, i.e., the cluster around which the supercluster $\hat{C}_j$ was created. Also, let $C_{j1}, C_{j2}, \ldots, C_{j(q_j)}$ denote the other clusters superclustered into the supercluster $\hat{C}_j$. Then the collection of edges

$$\{(C_j, C_{j1}), (C_j, C_{j2}), \ldots, (C_j, C_{j(q_j)})\} \mid 1 \leq j \leq p\}$$

forms a forest (a collection of disjoint stars), and thus contains less than n edges. For each of these edges, at most δ_i edges of the respective shortest path between C_j, for some $1 \leq j \leq p$,

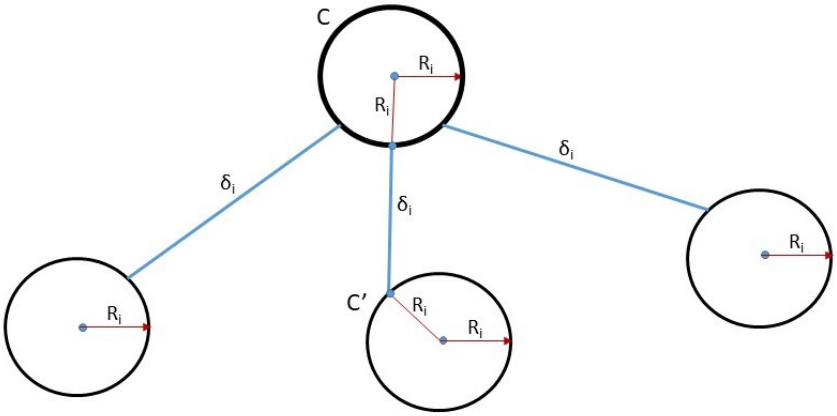

Figure 1: A drawing justifying the inequality $R_{i+1} \leq 3R_i + \delta_i$, where R_i is the radius of a level i cluster, and δ_i is the bound on the search distance at level i.

and some C_{jh}, for some $1 \leq h \leq q_j$, are inserted into the spanner H. Thus, the superclustering step of phase i inserts into the spanner $O(\delta_i \cdot n)$ edges. Thus, altogether phase i inserts into the spanner $O(\delta_i \cdot n^{1+1/\kappa})$ edges. Hence overall

$$|H| = O(n^{1+1/\kappa}) \cdot \sum_{i=0}^{\ell-1} \delta_i = O_{\epsilon,\kappa}(n^{1+1/\kappa}) \, .$$

(Recall that $\delta_i = (1/\epsilon)^i$.)

Stretch: Next we outline the stretch analysis of this construction.

Note that $\mathcal{U}_0 \cup \mathcal{U}_1 \cup \ldots \mathcal{U}_{\ell-1}$ is a partition of V. Let $\mathcal{U}^{(i)} = \bigcup_{j=0}^{i} \mathcal{U}_j$, for all $i \in [0, \ell-1]$. Observe also that all clusters of $\mathcal{U}_0$ are singletons, i.e., their radius (denoted R_0) is 0. Each of the clusters C in $\mathcal{U}^{(\ell-1)}$ has a designated center r, and the *radius* of C, denoted $Rad(C)$, is defined as $\max_{v \in C} d_H(r, v)$.

Generally, observe that the radius R_{i+1} of a cluster $\hat{C} \in U_{i+1}$, for some $0 \leq i \leq \ell - 2$, is at most $3R_i + \delta_i = 3R_i + (1/\epsilon)^i$. (Here R_i is the maximum radius of a cluster in $\mathcal{U}_i$.) See Figure 1 for an illustration.

Thus, $R_1 = 1$, and generally, for $0 \leq i \leq \ell - 2$, we have

$$R_{i+1} = \sum_{j=0}^{i} 3^{i-j}(1/\epsilon)^j < 2\left(\frac{1}{\epsilon}\right)^i \, ,$$

assuming $\epsilon < 1/6$.

Let $u, v \in V$ be a pair of vertices, and let $\pi(u, v)$ be a fixed shortest $u - v$ path in G. We partition it into segments of length $(1/\epsilon)^{\ell-1}$, except the last segment that may be shorter. Consider a particular fixed segment $x - y$ of this path, of length at most $(1/\epsilon)^{\ell-1} = \delta_{\ell-1}$.

It is convenient to imagine the path $\pi(u, v)$ and the subpath $\pi(x, y)$ as going from left to right, with u and x being the left endpoints and v and y being the right endpoints of their respective paths. Let z and w be the leftmost and the rightmost $\mathcal{U}_{\ell-1}$-clustered vertices on $\pi(x, y)$. (We assume that they exist. It is possible that $z = w$. If no such a vertex exists, the analysis is

actually simpler, as will be further indicated below.) Let $C_z, C_w \in \mathcal{U}_{\ell-1}$ be the clusters of z and w, respectively, i.e., $z \in C_z$, $w \in C_w$.

Then $d_G(C_z, C_w) \leq d_G(z, w) \leq (1/\epsilon)^{\ell-1}$, and thus, a shortest path $\tilde{\pi}$ between C_z and C_w was inserted into the spanner H. It follows that there exist vertices $\tilde{z} \in C_z$, $\tilde{w} \in C_w$, such that $\tilde{\pi}$ is the shortest $\tilde{z} - \tilde{w}$ path. Since spanning trees of radius at most $R_{\ell-1}$ for each cluster $C \in \mathcal{U}_{\ell-1}$ are contained in the spanner H, we conclude that

$$d_H(z, w) \leq d_G(C_z, C_w) + 4 \cdot R_{\ell-1} \leq (1/\epsilon)^{\ell-1} + 8 \cdot (1/\epsilon)^{\ell-2} .$$

Let z' (respectively, w') be the left (resp., right) neighbor of z (resp., w) on $\pi(x, y)$, if exists. Since z' and z belong to neighboring clusters of the ground partition Q, there is a path in H of length at most $4(\kappa - 1) + 1$ between them. Thus,

$$d_H(z', w') \leq (1/\epsilon)^{\ell-1} + 8 \cdot ((1/\epsilon)^{\ell-2} + \kappa) .$$

For simplicity, we suppress the term κ in this expression, as it is dominated by $(1/\epsilon)^{\ell-2} = (1/\epsilon)^{\log(\kappa+1)-2}$.

So the overall overhead so far that the spanner's path incurs in comparison to the original shortest path is $O((1/\epsilon)^{\ell-2})$, for each segment of length $(1/\epsilon)^{\ell-1}$. This amounts to the multiplicative stretch of $1 + O(\epsilon)$. The last segment of the path $\pi(u, v)$, which may be of length smaller than $(1/\epsilon)^{\ell-1}$, is responsible for the additive stretch of $O((1/\epsilon)^{\ell-2})$.

But we are not yet done. The spanner's path still needs to reach from x to z' and from w' to y. Observe that both these subsegments contain only vertices clustered at $\mathcal{U}^{(\ell-2)}$ (i.e., they are not clustered in $\mathcal{U}_{\ell-1}$).[3] We partition each of them into subsegments of length $\delta_{\ell-2} = (1/\epsilon)^{\ell-2}$ each, except the last subsegment that may be shorter.

On each such a subsegment $x' - y'$, we find the leftmost and the rightmost $\mathcal{U}_{\ell-2}$-clustered vertices $z_{\ell-2}$ and $w_{\ell-2}$. Let $z'_{\ell-2}$ be the left neighbor of $z_{\ell-2}$, and $w'_{\ell-2}$ be the right neighbor of $w_{\ell-2}$ along the path. The respective clusters $C(z_{\ell-2}), C(w_{\ell-2}) \in \mathcal{U}_{\ell-2}$ containing $z_{\ell-2}$ and $w_{\ell-2}$, respectively, have radius at most $R_{\ell-2} \leq 2(1/\epsilon)^{\ell-3}$. Hence an analogous computation to the one we did above for the $z - w$ path shows that the spanner H contains a $z'_{\ell-2} - w'_{\ell-2}$ path of length at most $d_G(z'_{\ell-2}, w'_{\ell-2}) + 8 \cdot ((1/\epsilon)^{\ell-3} + \kappa/2)$. (The second term is $\kappa/2$ and not κ, because clusters of the ground partition that may end up in a $\mathcal{U}_{\ell-2}$ cluster belong to $\hat{Q}_{\ell-2}$, and thus their radii are at most $\kappa/2$.) In other words, on each subsegment of length $(1/\epsilon)^{\ell-2}$, the spanner's path pays an overhead of $8 \cdot ((1/\epsilon)^{\ell-3} + \kappa/2)$, i.e., another multiplicative factor of $1 + O(\epsilon)$.

We then proceed by zooming in into subsegments between x' and $z'_{\ell-2}$, and between $w'_{\ell-2}$ and y', They are $\mathcal{U}^{(\ell-3)}$-clustered, and thus analogous considerations can be applied to them. Ultimately, this stretch analysis accumulates an overhead of $O(\epsilon)$-fraction of the length of the original path for ℓ times, leading to an overall multiplicative stretch of $1 + O(\epsilon \cdot \ell)$. The additive error of the spanner manifests itself on the last segment $x - y$ of the partition of $\pi(u, v)$ into segments of length $\delta_{\ell-1} = (1/\epsilon)^{\ell-1}$ is of length much smaller than $(1/\epsilon)^{\ell-1}$. Then the spanner's path pays an overhead of $O(R_{\ell-1}) = O((1/\epsilon)^{\ell-2})$, and this overhead cannot be charged to edges of the segment $x - y$, because the latter segment is too short.

To summarize, the spanner provides a stretch of $(1 + O(\epsilon \cdot \ell), (1/\epsilon)^{\ell-2})$. By rescaling, i.e.,

[3] The case that the entire $x - y$ path $\pi(x, y)$ has no $\mathcal{U}_{\ell-1}$-clustered vertices is actually a special case of the case considered here.

setting $\epsilon' = O(\epsilon \cdot \ell)$, one obtains a $(1+\epsilon', O\left(\frac{\ell}{\epsilon'}\right)^{\ell-2})$-spanner. Hence we have additive term

$$\beta = O\left(\frac{\log \kappa}{\epsilon'}\right)^{\log(\kappa+1)-2}.$$

We conclude this section with the following theorem:

Theorem 2.1. *[25] For every pair of parameters $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, there exists $\beta = \beta(\epsilon, \kappa) = O\left(\frac{\log \kappa}{\epsilon}\right)^{\log(\kappa+1)-2}$, such that for every unweighted undirected n-vertex graph $G = (V, E)$ there exists a $(1+\epsilon, \beta)$-spanner with $O_{\epsilon,\kappa}(n^{1+1/\kappa})$ edges.*

We note that if one is interested in an *emulator* as opposed to spanner, one can use the very same construction, but every time it inserted a shortest path between a pair of clusters C, C' into the spanner, the emulator will include a weighted edge between their respective centers r_C and $r_{C'}$ of weight equal to the distance $d_G(r_C, r_{C'})$ between these centers. It is easy to verify that the resulting emulator will have size $O_\kappa(n^{1+1/\kappa})$ (as opposed to $O_{\epsilon,\kappa}(n^{1+1/\kappa})$), i.e., its size will no longer depend on ϵ. One can also obtain this property for spanners constructed via the superclustering and interconnection approach, but via a slightly more involved construction that involves distance preservers [12], and with a slightly inferior additive error β [23].

2.2 Hopsets

In this section we argue that the same approach of superclustering and interconnection can be used to produce hopsets, with parameters similar to those of the corresponding near-additive spanners.

The algorithm produces a separate hopset for each distance scale. Assume that the smallest edge weight is 1, and let the aspect ratio Λ denote the maximum distance between a pair of vertices u, v in the input weighted undirected graph $G = (V, E, \omega)$. Our ultimate hopset H will be the union of single-scale hopsets H_i, where for each $i = 0, 1, \ldots, \lambda = \lceil \log_2 \Lambda \rceil$, H_i is the hopset that takes care of pairs of vertices u, v with $d_G(u, v) \in [2^i, 2^{i+1})$. (The last scale will always contain pairs with distance exactly Λ as well.) In [22] we showed that one can get rid of the dependence on the aspect ratio in the size of the ultimate hopset H. Here, however, we will describe a simpler construction in which $|H_i| = O_\kappa(n^{1+1/\kappa})$,[4] for every scale $i \in [0, \lambda]$, and thus, $|H| = O_\kappa(\log \Lambda \cdot n^{1+1/\kappa})$.

We fix a scale i, denote $R = 2^i$, and construct a hopset H_i that takes care of pairs of vertices u, v with $d_G(u, v) \in [R, 2R)$. From now on it will be referred to as $H' = H_i$.

We initialize $\mathcal{P}_0 = \{\{v\} \mid v \in V\}$ as the partition of V into singletons. (In [23] it was shown one can start with a partition into singletons when building near-additive spanners as well.) We initialize the set of uncovered clusters as $\mathcal{U}_0 \leftarrow \mathcal{P}_0$. Let $\delta_0 = R \cdot \epsilon^{\ell-1}$. Generally, we define $deg_i = n^{\frac{2^i}{\kappa}}$, for all $i \in [0, \ell-1]$, exactly as in the construction of near-additive spanners in Section 2.1.1. Also, we set $\delta_i = \delta_0/\epsilon^i$, for all $i \in [0, \ell-1]$. In particular, $\delta_{\ell-1} = R$. The way to think of it is that $\delta_0 = R \cdot \epsilon^{\ell-1}$ is the "distance unit" of the construction. Scaling down by the distance unit, one obtains the same sequence of distance thresholds as in Section 2.1.1.

Returning to the superclustering step of phase 0, if the algorithm finds an uncovered cluster $C \in \mathcal{U}_0$ with at least $deg_0 = n^{1/\kappa}$ other uncovered clusters $C' \in \mathcal{U}_0$ at distance at most δ_0 from it

[4] Specifically, $|H_i| = O(\log \kappa \cdot n^{1+1/\kappa})$. One can also eliminate the leading factor of $\log \kappa$ [22].

in G, then it creates a supercluster $\hat{C}$ out of them centered at the center r_C of C. (For a singleton cluster $C = \{v\}$, the center r_C is set as v.) The supercluster is created by adding into it C, and the nearby clusters $C' \in \mathcal{U}_0$ (at distance at most δ_0 from C in G). One also adds to the hopset the edges $\{(r_C, r_{C'}) \mid C' \in \hat{C}\}$, with weights $\omega((r_C, r_{C'})) = d_G(r_C, r_{C'})$. The cluster C and the clusters C' as above are then removed from $\mathcal{U}_0$, i.e., they are marked as covered. We then proceed to constructing the next supercluster in the same manner. The superclustering step (of phase 0) proceeds iteratively up until no additional supercluster can be created.

The set $\mathcal{U}_0$ of remaining *unclustered* clusters is then the input to the interconnection step of phase 0. On this step each pair of nearby clusters $C, C' \in \mathcal{U}_0$ (i.e., $d_G(C, C') \leq \delta_0$) is interconnected by a direct hopset edge $(r_C, r_{C'})$ between their respective centers. Its weight is also set as $d_G(r_C, r_{C'})$, This concludes the description of phase 0.

The resulting collection $\mathcal{P}_1$ of superclusters created on phase 0 is the input for phase 1. Phase 1 proceeds in the same way (as phase 0), except that its degree and threshold parameters are $deg_1 = n^{\frac{2^1}{\kappa}}$ and $\delta_1 = \delta_0/\epsilon^1$. This is also the case for phases $i \geq 1$, that have $deg_i = n^{\frac{2^i}{\kappa}}$ and $\delta_i = \delta_0/\epsilon^i$. When the algorithm reaches phase $\ell - 1$, all clusters of $\mathcal{P}_{\ell-1}$ are already of the size at least $n^{\frac{2^{\ell-1}-1}{\kappa}} = n^{\frac{\kappa-1}{2\kappa}}$. (By the same argument as in Section 2.1.2.) Hence $|\mathcal{P}_{\ell-1}| \leq n^{\frac{\kappa+1}{2\kappa}}$ (because the clusters are disjoint). So the superclustering step of phase $\ell - 1$ is skipped. Instead the algorithm proceeds directly to interconnecting all pairs of clusters of $\mathcal{P}_{\ell-1}$. (We also set $\mathcal{U}_{\ell-1} = \mathcal{P}_{\ell-1}$, to reflect the intuition that all clusters of phase $\ell-1$ are uncovered.) By "interconnecting" a pair (C, C') of clusters, we again mean inserting into the hopset the edge $(r_C, r_{C'})$ between their respective centers, with weight $\omega((r_C, r_{C'})) = d_G(r_C, r_{C'})$.

This concludes the description of the algorithm. The analysis of $|H'|$ (the size analysis) is analogous to the size analysis of the near-additive spanner from Section 2.1.2. We omit it. The size bound is $|H| = O_\kappa(n^{1+1/\kappa})$. We next sketch the analysis of its stretch and hopbound.

Consider a pair $u, v \in V$ of vertices such that $d_G(u, v) \in [R, 2R]$, and let $\pi = \pi(u, v)$ be a shortest path between them. Observe that the sets $\{\mathcal{U}_i \mid 0 \leq i \leq \ell - 1\}$ form a partition of V, exactly as in the case of near-additive spanners. We also write $\mathcal{U}^{(i)} = \bigcup_{j \leq i} \mathcal{U}_j$, for all $i \in [0, \ell - 1]$.

Next we provide upper bounds R_i on the radii of clusters of $\mathcal{U}_i$. Clusters of $\mathcal{U}_0$ are singletons, and thus $R_0 = 0$. In general, for $i \in [0, \ell - 2]$, we have $R_{i+1} = 3R_i + \delta_i$. Hence we have

$$R_{i+1} = \sum_{j=0}^{i} 3^j \delta_{i-j} = \delta_0/\epsilon^i \sum_{j=0}^{i} (3\epsilon)^j \leq \delta_0/\epsilon^i \frac{1}{1 - 3\epsilon} .$$

For $\epsilon \leq 1/6$, we have $R_{i+1} \leq 2\delta_0/\epsilon^i$. In particular, $R_{\ell-1} \leq 2\delta_0(1/\epsilon)^{\ell-2} = 2R\epsilon$. (Recall that $R = \delta_0/\epsilon^{\ell-1}$.)

Moreover, it is easy to verify (by induction on i) that the radius of each cluster of U_i, for all $i \in [0, \ell - 1]$, is attained by at most i hops.

Let z and w be the leftmost and the rightmost $\mathcal{U}_{\ell-1}$-clustered vertices on π, respectively. Let C_z and C_w be the $\mathcal{U}_{\ell-1}$-clusters of z and w, respectively, and let r_z and r_w denote their respective centers. Observe that the hopset H' contains a $z - w$ path obtained by going from z to r_z in $\ell - 1$ hops or less, from r_z to r_w via direct edge of H', and from r_w to w in at most $\ell - 1$ additional hops. The length of this path is at most

$$2R_{\ell-1} + d_G(r_z, r_w) \leq 2R_{\ell-1} + d_G(z, w) + 2R_{\ell-1} = d_G(z, w) + 8\delta_0 \cdot (1/\epsilon)^{\ell-2} .$$

Moreover, let z' be the left neighbor of z on π, and w' be the right neighbor of w on π. Then the

path in $G \cup H'$ between z' and w', that starts with G-edge (z', z), then takes the above hopset's $z - w$ path, and finally uses the G-edge (w, w'), has length at most $d_G(z', w') + 8\delta_0 \cdot (1/\epsilon)^{\ell-2}$, and uses at most $2 + 2(\ell - 1) + 1 = 2\ell + 1$ hops. Hence the overhead of $8\delta_0 \cdot (1/\epsilon)^{\ell-2}$ can be charged to the length of $\pi(u, w)$, which is at least $R = \delta_0 \cdot (1/\epsilon)^{\ell-1}$. This is a multiplicative overhead of $1 + 8\epsilon$.

Note also that the segments $u - z'$ and $w' - v$ of $\pi(u, v)$ contain vertices which are all clustered in $\mathcal{U}^{(\ell-2)}$. (In other words, no vertex in these subpaths is $\mathcal{U}_{\ell-1}$-clustered.) We divide these segments into subsegments of length at most $R \cdot \epsilon = \delta_0 \cdot (1/\epsilon)^{\ell-2} = \delta_{\ell-2}$.

In the case of hopsets this step requires more care than in the case of near-additive spanners, because in the latter case we dealt with *unweighted* graphs. Here the first segment starts at $u = x_0$, and ends in the last vertex $y = y_0$ along π such that $d_G(x_0, y_0) \leq R\epsilon$. If $d_G(x_0, y_0) < R\epsilon$, then the next segment starts in the right neighbor (with respect to π) x_1 of the vertex y_0. The edge (y_0, x_1) is called a *connecting* edge between the two consecutive segments $x_0 - y_0$ and $x_1 - y_1$. Otherwise (if $d_G(x_0, y_0) = R\epsilon$), we set $x_1 = y_0$. Then again y_1 is defined as the rightmost vertex with $d_G(x_1, y_1) \leq R\epsilon$ along π, etc. This process continues until we reach z'. The subpath between w' and v is divided into segments and connecting edges in the same manner.

In each such a segment (x, y), we find the leftmost and the rightmost $\mathcal{U}_{\ell-2}$-clustered vertices $z_{\ell-2}$ and $w_{\ell-2}$. Let C_z and C_w be their respective clusters, and r_z and r_w be their respective cluster centers. Observe that the distance between C_z and C_w is at most $R\epsilon = \delta_{\ell-2}$, and thus their centers r_z and r_w were interconnected by a direct hopset edge (r_z, r_w) in the interconnection step of phase $\ell - 2$. The length of this hopset edge is

$$\omega(r_z, r_w) = d_G(r_z, r_w) \leq d_G(r_z, z_{\ell-2}) + d_G(z_{\ell-2}, w_{\ell-2}) + d_G(w_{\ell-2}, r_w) \leq d_G(z_{\ell-2}, w_{\ell-2}) + 2R_{\ell-2} \ .$$

Hence there is a $z - w$ path in the hopset that goes from z to r_z, uses the edge (r_z, r_w), and goes from r_w to w. Its length is at most

$$d_G(z_{\ell-2}, w_{\ell-2}) + 4R_{\ell-2} \leq d_G(z_{\ell-2}, w_{\ell-2}) + 8\delta_0(1/\epsilon)^{\ell-3} \ ,$$

and it uses at most $2(\ell - 2) + 1 = 2\ell - 3$ hops. Define $z'_{\ell-2}$ to be the left neighbor of $z_{\ell-2}$ on π, and $w'_{\ell-2}$ to be the right neighbor of $w_{\ell-2}$ on π. Then we also obtain a $z'_{\ell-2} - w'_{\ell-2}$ path in $G \cup H'$ with at most $2\ell - 1$ hops and length at most $d_G(z'_{\ell-2}, w'_{\ell-2}) + 8\delta_0(1/\epsilon)^{\ell-3}$. We charge the overhead of $8\delta_0(1/\epsilon)^{\ell-3}$ to the length $\delta_0 \cdot (1/\epsilon)^{\ell-2} = R\epsilon$ of the segment between x and y.[5] As a result the incurred stretch is at most $1 + 8\epsilon$ (on top of the stretch $1 + 8\epsilon$ incurred on the top-most, $(\ell - 1)$st, level of the stretch analysis).

The number of hops incurred on the $(\ell - 2)$nd level of the stretch analysis can be upper-bounded as follows. There are $1/\epsilon$ segments, and on each of them $2\ell - 1 = O(\ell)$ hops are used. (In addition, one hop per segment is used for connecting edges, but this is swallowed in the O-notation.) Hence the number of hops used by this level of stretch analysis is $O(\ell/\epsilon)$.

We then continue the stretch analysis in the same way, by zooming in into each of the subsegments $x - z'_{\ell-2}$ and $w'_{\ell-2} - y$. Their vertices are $\mathcal{U}^{(\ell-3)}$-clustered, and thus on the next level of the stretch analysis we consider the leftmost and the rightmost $\mathcal{U}_{\ell-3}$-clustered vertices on each subsegment of length at most $R \cdot \epsilon^2 = \delta_{\ell-3}$, etc.

The overall accumulated stretch on all the ℓ levels of the stretch analysis is thus $1 + 8\epsilon\ell$, and the overall number of hops can be crudely upper-bounded by $O(\ell) \cdot (1/\epsilon)^{\ell-1}$. (To see this upper bound, observe that eventually we partition the path into $O((1/\epsilon)^{\ell-1})$ segments. The

[5] Strictly speaking, one needs also include the connecting edge incident on y in the segment.

above analysis shows that on each segment at most $O(\ell)$ hops are used. To have a more precise bound, one notes that in fact on lower levels of the stretch analysis less hops per segment are used. This leads to a bound of $O((1/\epsilon)^{\ell-1})$.)

We also remark that no additive error is present here, even though the last segment may be shorter than $R\epsilon$. This is because the entire path that we consider has length $\Theta(R)$, and thus the additive error of the last segment is swalllowed in the multiplicative stretch of $1 + O(\epsilon)$. (This is unlike the case of near-additve spanners, where the original path may be very short, and so the additive error cannot be charged to the length of the original path.)

We now rescale $\epsilon' = O(\epsilon\ell)$, and obtain stretch of $1 + \epsilon'$ and hopbound $\beta = O\left(\frac{\log\kappa}{\epsilon'}\right)^{\log(\kappa+1)-2}$. We summarize the result in the next theorem.

Theorem 2.2. *[22] For every pair of parameters $\epsilon > 0$ and $\kappa = 1, 2, \ldots$, there exists $\beta = \beta(\epsilon, \kappa) = O\left(\frac{\log\kappa}{\epsilon}\right)^{\log(\kappa+1)-2}$, such that for every weighted undirected n-vertex graph $G = (V, E, \omega)$, there exists a $(1 + \epsilon, \beta)$-hopset with $O_{\epsilon,\kappa}(n^{1+1/\kappa} \log \Lambda)$ edges.*

As was remarked above, the $\log \Lambda$ factor in the size can be replaced by $\log n$ (see [22]). Note the striking similarity between Theorems 2.1 and 2.2.

3 Scale-Free Hopsets and Spanners

In this section we present a universal extension of constructions from [25, 22], described in Section 2. They were developed in [41, 24, 29].

3.1 Construction

Let $G = (V, E)$ be a graph with n vertices (possibly with non-negative weights $w : E \to \mathbb{R}$ on the edges). Fix an integer parameter $\kappa \geq 1$ (it will be convenient to assume $\kappa = 2^\ell - 1$ for some integer $\ell \geq 1$). Denote $\ell = \log(\kappa + 1)$. Let $A_0, \ldots, A_\ell$ be sets of vertices such that $A_0 = V$, $A_\ell = \emptyset$, and for $0 \leq i \leq \ell - 2$, A_{i+1} is created by sampling independently every vertex of A_i with probability $q_i = n^{-2^i/\kappa}$.

For every $v \in V$ and $0 \leq i \leq \ell - 1$, define the pivot $p_i(v)$ as the closest vertex in A_i to v, breaking ties in a consistent matter. For every $0 \leq i \leq \ell - 1$ and every $u \in A_i \setminus A_{i+1}$ define the bunch

$$Bunch(v) = B(u) = \{v \in A_i \ : \ d_G(u, v) < d_G(u, A_{i+1})\} \cup \{p_{i+1}(u)\} .$$

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 the level $i + 1$ pivot. We then define the emulator (and the hopset) $H = \{(u, v) \ : \ u \in V, v \in B(u)\}$, where the length $\omega'(u, v)$ of the edge (u, v) is set as $d_G(u, v)$.

Size Analysis. Fix any $0 \leq i \leq \ell - 2$ and $u \in A_i \setminus A_{i+1}$, and consider the expected size of $B(u)$. If one orders the vertices of A_i by their distance to u, then $B(u)$ contains the prefix of all the vertices in that ordering until the first one sampled to A_{i+1}. As this is a geometric random variable with parameter q_i, its expectation is $1/q_i = n^{2^i/\kappa}$. In addition, each vertex is connected to at most ℓ pivots, adding a term of ℓn.

Note that each $v \in V$ is included in A_i with probability $\prod_{j=0}^{i-1} q_j$. These choices are independent for different vertices, so the expected size of A_i is:

$$N_i := \mathbb{E}[|A_i|] = n \prod_{j=0}^{i-1} q_j = n^{1-(2^i-1)/\kappa} .$$

So for each $0 \leq i \leq \ell-2$, we have $N_i/q_i = n^{1+1/\kappa}$. In addition, $N_{\ell-1} = n^{1-(2^{\ell-1}-1)/\kappa} = n^{(1+1/\kappa)/2}$, and it can be checked that

$$\mathbb{E}[|A_{\ell-1}|^2] \leq O(n^{1+1/\kappa}) .$$

Note that for $u \in A_{\ell-1}$ we have $B(u) = A_{\ell-1}$, thus the expected size of the hopset H is

$$\sum_{i=0}^{\ell-2} N_i/q_i + E[|A_{k-1}|^2] + \ell n = O(\log \kappa \cdot n^{1+1/\kappa}) .$$

We remark that a more refined choice for the probabilities q_i (and connecting to just 1 pivot, rather than all of them), can lead to size $O(n^{1+1/\kappa})$, essentially without affecting the other parameters.

3.2 Stretch Analysis of the Emulator

In this section we show that the edge set H constructed above can serve as a universal near-additive emulator for G.

Consider a pair of vertices $u, v \in V$. Let $\pi(u, v)$ be a shortest $u - v$ path. For some $\epsilon > 0$, we partition the path into segments of length $(1/\epsilon)^{\ell-1}$, except the last segment that may be shorter. Each such a segment $x - y$ will be called a *level-*$(\ell - 1)$ segment. It will be further subdivided into level-$(\ell - 2)$ segments of length $(1/\epsilon)^{\ell-2}$, etc. In general, for any $0 \leq i \leq \ell - 1$, level-i segments have length $(1/\epsilon)^i$.[6]

Lemma 3.1. *There exist two universal constants $c, c' > 0$, such that for any i, $0 \leq i \leq \ell - 1$, any i-level segment $x - y$ is either* successful*, i.e., satisfies*

$$(1)\ \ d_H(x, y) \leq d_G(x, y) + c \cdot i \cdot (1/\epsilon)^{i-1} ,$$

or fails*, i.e., satisfies*

$$(2)\ \ d_G(x, p_{i+1}(x)) \leq c' \cdot (1/\epsilon)^i .$$

Proof: The proof is by induction on i.

Base: $(i = 0)$

Level $i = 0$ segments have length 1, i.e., $(x, y) \in E$ is an edge. If $x \in A_1$, then $p_1(x) = x$, and so the segment fails ($d_G(x, p_1(x)) = 0$). Otherwise $x \in A_0 \setminus A_1$.

Then either $(x, y) \in H$, and then the segment is successful, as condition (1) holds with 1 at the right-hand-side. Or, alternatively, $(x, y) \notin H$, i.e., $y \notin Bunch(x)$. But then $d_G(x, p_1(x)) \leq d_G(x, y) = 1$, proving condition (2) (i.e., the segment fails).

Step:

[6] Except possibly one level-i subsegment of the possibly shorter level-$(\ell - 1)$ segment; but this technicality has no real effect on the analysis.

Suppose that the assertion holds for all level-i segments, for some $0 \le i \le \ell - 2$. Consider a level-$(i+1)$ segment $x - y$. If all its level-i subsegments are successful, then we concatenate the emulator's substitute paths for them. In the case that all its level-i subsegments have length exactly $(1/\epsilon)^i$, the length of the resulting path in the emulator can be bounded by

$$\begin{aligned} d_H(x,y) &\le 1/\epsilon \cdot ((1/\epsilon)^i + c \cdot i \cdot (1/\epsilon)^{i-1}) \\ &= (1/\epsilon)^{i+1} + c \cdot i \cdot (1/\epsilon)^i \,. \end{aligned}$$

In the general case, one obtains here an upper bound of $d_G(x,y) + c \cdot i \cdot (1/\epsilon)^i$, by essentially the same argument. Hence in this case the segment $u - v$ is successful as well.

Otherwise there are some failing level-i subsegments of $x - y$. Let $x_L - y_L$ and $x_R - y_R$ be the leftmost and the rightmost such subsegments. Let $z_L = p_{i+1}(x_L)$, $z_R = p_{i+1}(y)$. By the inductive hypothesis, we have $d_G(x_L, z_L), d_G(y_R, z_R) \le c' \cdot (1/\epsilon)^i$. Then

$$\begin{aligned} d_G(z_L, z_R) &\le d_G(z_L, x_L) + d_G(x_L, y_R) + d_G(y_R, z_R) \\ &\le d_G(x_L, y_R) + 2c' \cdot (1/\epsilon)^i \,. \end{aligned}$$

Observe that the edges (x_L, z_L), (y_R, z_R) belong to the emulator H. If $(z_L, z_R) \in H$ as well, then

$$\begin{aligned} d_H(x_L, y_R) &\le d_H(x_L, z_L) + d_H(z_L, z_R) + d_H(z_R, y_R) \\ &= d_G(x_L, z_L) + d_G(z_L, z_R) + d_G(z_R, y_R) \\ &\le d_G(x_L, y_R) + 2(d_G(x_L, z_L) + d_G(z_R, y_R)) \;\le\; d_G(x_L, y_R) + 4c' \cdot (1/\epsilon)^i \,. \end{aligned}$$

Also, note that each of the level-i segments of the subpaths $x - x_L$ and $y_R - y$ of the segment $x - y$ are successful, and there are

$$\frac{d_G(x, x_L) + d_G(y_R, y)}{(1/\epsilon)^i}$$

such segments. Hence

$$\begin{aligned} d_H(x, x_L) + d_H(y_R, y) &\le \frac{d_G(x, x_L) + d_G(y_R, y)}{(1/\epsilon)^i} \cdot ((1/\epsilon)^i + c \cdot i \cdot (1/\epsilon)^{i-1}) \\ &= (d_G(x, x_L) + d_G(y_R, y)) \cdot (1 + c \cdot i \cdot \epsilon) \,. \end{aligned}$$

Thus we have

$$\begin{aligned} d_H(x,y) &\le d_H(x, x_L) + d_H(x_L, y_L) + d_H(y_R, y) \\ &\le (d_G(x, x_L) + d_G(y_R, y)) \cdot (1 + c \cdot i \cdot \epsilon) + d_G(x_L, y_R) + 4c' \cdot (1/\epsilon)^i \\ &\le d_G(x,y)(1 + c \cdot i \cdot \epsilon) + 4c' \cdot (1/\epsilon)^i \,. \end{aligned}$$

Observe that $d_G(x,y) = (1/\epsilon)^{i+1}$, i.e., $4c'(1/\epsilon)^i = 4c'\epsilon \cdot d_G(u,v)$. Hence in this case

$$d_H(x,y) \le (1 + c \cdot i \cdot \epsilon + 4c' \cdot \epsilon) \cdot d_G(x,y) \,.$$

For $c = 4c'$, we obtain that

$$d_H(x,y) \le (1 + c(i+1)\epsilon)(1/\epsilon)^{i+1} \,,$$

and thus the segment $x-y$ is successful.

Otherwise $(z_L, z_R) \notin H$, i.e., $z_R \notin Bunch(z_L)$. Then we have

$$d_G(z_L, p_{i+1}(z_L)) \leq d_G(z_L, z_R) \leq d_G(x_L, y_R) + 2c'(1/\epsilon)^i \ .$$

Hence

$$\begin{aligned} d_G(x, p_{i+1}(x)) &\leq d_G(x, p_{i+1}(z_L)) \leq d_G(x, x_L) + d_G(x_L, z_L) + d_G(z_L, p_{i+1}(z_L)) \\ &\leq d_G(x, x_L) + c' \cdot (1/\epsilon)^i + d_G(x_L, y_R) + 2c'(1/\epsilon)^i \\ &\leq d_G(x, y) + 3c'(1/\epsilon)^i = (1/\epsilon)^{i+1} + 3c'(1/\epsilon)^i \ \leq \ c'(1/\epsilon)^{i+1} \ . \end{aligned}$$

The last inequality holds for $c' \geq \frac{1}{1-3\epsilon}$. Hence if we set $c' \geq 2$, it holds for all $\epsilon < 1/6$.

This completes the proof. ∎

Observe that an $(\ell-1)$-level segment $x-y$ cannot fail, and thus we have $d_H(x,y) \leq d_G(x,y)(1 + c(\ell-1)\cdot\epsilon$. By concatenating the emulator's substitute paths for all the segments, we obtain that for any $u, v \in V$,

$$d_H(u,v) \leq (1 + c(\ell-1)\cdot\epsilon) d_G(u,v) + O(c(\ell-1)(1/\epsilon)^{\ell-1}) \ .$$

(Exactly as in Section 2, the additive term is because of the last segment.)

By rescaling $\epsilon' = c(\ell-1)\epsilon$, we obtain that the stretch of the emulator is $(1+\epsilon, O\left(\frac{\log\kappa}{\epsilon}\right)^{\log(\kappa+1)-2})$. Note also that this construction does not accept ϵ as a parameter, and thus applies to all $\epsilon > 0$.

We summarize this analysis with the following theorem due to [41]. (The proof that we provided is however different from the original proof from [41].)

Theorem 3.2. *[41] For any $\kappa = 1, 2, \ldots$, and any n-vertex graph $G = (V, E)$, the graph $G' = (V, H, \omega')$ constructed as above is a $(1+\epsilon, \beta(\epsilon,\kappa))$-emulator for G with $O_\kappa(n^{1+1/\kappa})$ edges, for all $\epsilon < 1/6$, where $\beta = \beta_{EP}$.*

3.3 Stretch Analysis of the Hopset

In this section we show that the very same edge set H constructed in the beginning of this section provides a $(1+\epsilon, \beta)$-hopset (naturally, of the same size), even for weighted graphs.

Again, consider a shortest $u-v$ path $\pi(u,v)$. Denote $L = \omega(\pi(u,v))$. We partition it into $1/\epsilon$ segments of length $L\epsilon$ each. (Suppose for simplicity that it can be divided into segments of precisely this length. If it is not the case, it can be taken care of, essentially without affecting the analysis.) Those segments are again subdivided to $1/\epsilon$ subsegments of length $L \cdot \epsilon^2$ each, etc, for $\ell - 1$ levels. Segments of length $(L \cdot \epsilon^{\ell-1}) \cdot (1/\epsilon)^i$ are the i-level segments. So, in a sense, $\gamma = L \cdot \epsilon^{\ell-1}$ is the "distance unit" of the construction. See also Section 2.

One can assume that all weights are greater or equal to 1. Assume also that $L \geq (1/\epsilon)^{\ell-1}$. If it is not the case, the graph G itself has a $u-v$ path of length $d_G(u,v)$ with at most $(1/\epsilon)^{\ell-1}$ hops.

The next lemma is completely analogous to Lemma 3.1.

Lemma 3.3. *There exist universal constants $c, c' > 0$ such that for any $0 \leq i \leq \ell-1$, any i-level segment $x-y$ is either* successful*, i.e., satisfies*

$$(1) \ \ d_H^{(1/\epsilon)^i)}(x,y) \leq \gamma \cdot \left((1/\epsilon)^i + ci \cdot (1/\epsilon)^{i-1}\right) \ ,$$

or fails*, i.e., satisfies*

$$(2) \ \ d_G(x, p_{i+1}(x)) \leq \gamma \cdot c' \cdot (1/\epsilon)^i \ .$$

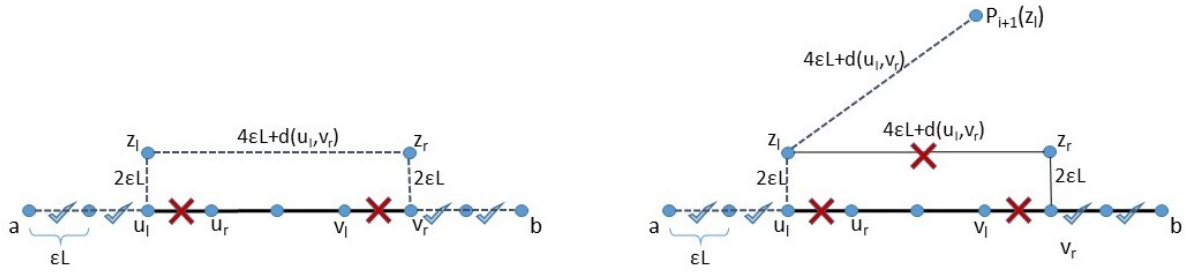

Figure 2: An illustration for the $1+\epsilon$ stretch version. Above are the two cases when considering an interval $[a,b]$ of length L at level i, which is divided to $1/\epsilon$ sub-intervals (the case when all sub-intervals are successful is omitted). The dashed line represents the path in $G \cup H$ we find. On the left is the case that some sub-intervals failed, and there is an hopset edge between the level i pivots of the leftmost and rightmost failed intervals' endpoints; in this case we have a $1/\epsilon^i$-hops path with stretch $1+ci\epsilon$. The other case is that there is no such edge, but then we see a level $i+1$ pivot at distance at most $c'L$.

Proof: The proof is again by induction on i.

Base: $(i=0)$

If $x \in A_1$, then the segment satisfies (2) with 0 in the right-hand-side. Otherwise $x \in A_0 \setminus A_1$. If $(x,y) \in H$ then the segment is successful. Otherwise, $d_G(x,p_1(x)) \leq d_G(x,y) = \gamma$, and the segment fails. In both cases the assertion of the lemma holds.

Step: The proof of the induction step is completely analogous to the proof of the induction step of Lemma 3.1, except that all expressions need to be scaled up by a factor of γ. An illustration is provided in Figure 2.

∎

Lemma 3.3 implies the following theorem.

Theorem 3.4. *[24, 29] For any $\kappa = 1,2\ldots$, and any n-vertex weighted graph $G=(V,E,\omega)$, the graph $G'=(V,H,\omega')$ constructed above is a $(1+\epsilon,\beta)$-hopset for G with $O_\kappa(n^{1+1/\kappa})$ edges, and $\beta=\beta_{EP}$.*

4 Conclusions and Open Problems

As we have seen, there is a striking similarity not just between the results concerning near-additive spanners for unweighted graphs and near-exact hopsets for weighted ones, but also between the techniques used to construct them and to analyze these constructions. Specifically, the superclustering and interconnection approach (see Section 2) due to [25] gives rise to very similar constructions of these two objects [25, 22], and this is also the case with its scale-free extension due to [41] (see [24, 29] and Section 3).

The situation is similar in the case of Cohen's approach [15] that relies on pairwise covers [14, 2]. This approach also gives rise to closely related constructions and analyses for both near-exact hopsets [15] and near-additive spanners [20, 26]. This approach was left out of the scope of the current survey.

A very interesting open problem is to explain the relationship between near-additive spanners and near-exact hopsets rigorously, i.e., by providing a reduction between these two objects.

Another major open question is to determine the correct dependency of β on ϵ and κ for both spanners and hopsets. Can one achieve β polynomial in κ for near-additive spanners and/or near-exact hopsets?

Numerous related open problems arise if one allows a larger stretch than $1+\epsilon$. Currently there are known constructions with stretch $3+\epsilon$ and β polynomial in κ [34, 19, 10]. Can this be achieved with stretch smaller than 3? What is the right three-way tradeoff between the sparsity parameter κ, the multiplicative stretch α and the hop-bound (or additive stretch) β?

We have also pointed out that the current state-of-the-art constructions of *universal* near-additive spanners (see Section 3) lag behind their non-universal counterparts. Specifically, the state-of-the-art bound on the parameter β in the universal constructions [23, 34] is $\beta_{EP}^{\log_{4/3} 2}$, where $\beta_{EP} = \left(\frac{\log \kappa}{\epsilon}\right)^{\log \kappa - 2}$ is the state-of-the-art bound for non-universal constructions [25]. Narrowing this gap, or proving a lower bound precluding this, is an open problem.

In this survey we focused on *existential*, i.e., combinatorial properties of near-additive spanners and near-exact hopsets. However, for many applications it is important to compute them efficiently in various computational models. For example, in the centralized model of computation one introduces a control parameter $\rho > 0$, and can obtain $(1+\epsilon, \beta)$-spanners with $O_{\epsilon,\kappa}(n^{1+1/\kappa})$ edges and

$$\beta = \left(\frac{\log \kappa\rho + 1/\rho}{\epsilon}\right)^{\log \kappa\rho + 1/\rho}$$

in time $O(|E| \cdot n^{\rho})$ [22, 23, 20, 26]. The tradeoff looks similarly in other models of computation, i.e., the overhead of n^{ρ} in the running time at the expense of larger β is persistent. Improving upon this tradeoff is an open problem. Its positive resolution is likely to lead to improved algorithms for the computation of approximate shortest paths, distributed routing tables, parallel distance oracles, and other applications.

Finally, in many applications of hopsets one needs not just approximate distances, but also paths that implement these distances. For this aim, *path-reporting* hopsets, i.e., hopsets from which approximate paths can be readily retrieved were introduced in [22]. Their parameters are, however, somewhat inferior to those of their non-path-reporting counterparts. Devising path-reporting hopsets with improved parameters is also an interesting open problem.

5 Acknowledgement

This research was supported by the ISF grant No. (2344/19) and (1817/17), and by BSF grant No. 2015813.

References

[1] Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *CoRR*, abs/1511.00700, 2015.

[2] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear cost sequential and distribured constructions of sparse neighborhood covers. In *34th Annual Sym-*

posium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993, pages 638–647, 1993.

[3] 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.

[4] Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. *SIAM J. Comput.*, 47(6):2203–2236, 2018.

[5] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

[6] Ingo Althöfer, Gautam Das, David P. Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In *SWAT*, pages 26–37, 1990.

[7] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discrete Math.*, 5(2):151–162, 1992.

[8] 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.

[9] S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. New constructions of (α,β)-spanners and purely additive spanners. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005.

[10] Uri Ben-Levy and Merav Parter. New (α, β) spanners and hopsets. *CoRR*, abs/1907.11402, 2019.

[11] S. Baswana and S. Sen. A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Proc. 30th Intl. Colloq. on Automata, Languages and Programming (ICALP)*, 2003.

[12] D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 660–669, 2005.

[13] Shiri Chechik. New additive spanners. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 498–512, 2013.

[14] 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.

[15] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 16–26, 1994.

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

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

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

[19] Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and PRAM distance oracles in weighted graphs. *CoRR*, abs/1907.11422, 2019.

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

[21] Michael Elkin and Shaked Matar. Near-additive spanners in low polynomial deterministic CONGEST time. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 531–540, 2019.

[22] Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. In *57th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, NJ, USA, October 2016*, 2016.

[23] Michael Elkin and Ofer Neiman. Efficient algorithms for constructing very sparse spanners and emulators. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 652–669, 2017.

[24] Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and distributed routing with low memory. *CoRR*, abs/1704.08468, 2017.

[25] M. Elkin and D. Peleg. $(1+\epsilon,\beta)$-spanner constructions for general graphs. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 173–182, 2001.

[26] M. Elkin and J. Zhang. Efficient algorithms for constructing $(1+\epsilon,\beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18:375–385, 2006.

[27] 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.

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

[29] Shang-En Huang and Seth Pettie. Thorup-zwick emulators are universally optimal hopsets. *CoRR*, abs/1705.00327, 2017.

[30] S. Halperin and U. Zwick. An optimal randomised logarithmic time connectivity algorithm for the EREW PRAM. *J. Comput. Syst. Sci.*, 53(3):395–416, 1996.

[31] S. Halperin and U. Zwick. Unpublished. 1996.

[32] 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.

[33] 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.

[34] S. Pettie. Low distortion spanners. In *Proc. 34th Int'l Colloq. on Automata, Languages, and Programming (ICALP)*, pages 78–89, 2007.

[35] Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 253–262, New York, NY, USA, 2008. ACM.

[36] Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010.

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

[38] L. Roditty, M. Thorup, and U. Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Proc. 32nd Int'l Colloq. on Automata, Lang., and Prog. (ICALP)*, pages 261–272, 2005.

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

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

[41] M. Thorup and U. Zwick. Spanners and emulators with sublinear distance errors. In *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 802–809, 2006.

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

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