New Additive Emulators

Shimon Kogan
Weizmann Institute of Science, Rehovot, Israel

Merav Parter
Weizmann Institute of Science, Rehovot, Israel

Abstract

For a given (possibly weighted) graph $G = (V, E)$, an additive emulator H is a weighted graph in $V \times V$ that preserves the (all pairs) G-distances up to a small additive stretch. In their breakthrough result, [Abboud and Bodwin, STOC 2016] ruled out the possibility of obtaining $o(n^{4/3})$-size emulator with $n^{o(1)}$ additive stretch. The focus of our paper is in the following question that has been explicitly stated in many of the prior work on this topic:

What is the minimal additive stretch attainable with linear size emulators?

The only known upper bound for this problem is given by an implicit construction of [Pettie, ICALP 2007] that provides a linear-size emulator with $+\widetilde{O}(n^{1/4})$ stretch. No improvement on this problem has been shown since then.

In this work we improve upon the long standing additive stretch of $\widetilde{O}(n^{1/4})$, by presenting constructions of linear-size emulators with $\widetilde{O}(n^{0.222})$ additive stretch. Our constructions improve the state-of-the-art size vs. stretch tradeoff in the entire regime. For example, for every $\epsilon > 1/7$, we provide $+n^{f(\epsilon)}$ emulators of size $\widetilde{O}(n^{1+\epsilon})$, for $f(\epsilon) = 1/5 - 3\epsilon/5$. This should be compared with the current bound of $f(\epsilon) = 1/4 - 3\epsilon/4$ by [Pettie, ICALP 2007].

The new emulators are based on an extended and optimized toolkit for computing weighted additive emulators with sublinear distance error. Our key construction provides a weighted modification of the well-known Thorup and Zwick emulators [SODA 2006]. We believe that this TZ variant might be of independent interest, especially for providing improved stretch for distant pairs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Spanners, Emulators, Distance Preservers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.85

Category Track A: Algorithms, Complexity and Games

Funding This project is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 949083), and by the Israeli Science Foundation (ISF), grant No. 2084/18.

1 Introduction

Emulators are well-studied compression schemes that approximately encode the distance metric of a (dense) undirected input graph $G = (V, E)$ by a sparse *weighted* graph $H \subseteq V \times V$. This extends the notion of spanners which are required to be subgraphs of G. Along with their spanner cousin, emulators admit a wide range of algorithmic applications, most notably in settings related to graph compression, routing schemes, distributed computing, and all pairs shortest paths approximation. The focus of this paper is in providing improved constructions for *additive emulators* which only allow for additive stretch. For a given unweighted n-vertex graph $G = (V, E)$, a graph $H \subseteq V \times V$ is an $f(d)$-emulator if $\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq f(\text{dist}_G(u, v))$ for every $u, v \in V$. An $f(d)$-emulator for $f(d) = d + \beta$ for some fixed β is denoted as *additive* emulator. There has been a long line of work on additive emulators, both from an upper bound and lower bound perspectives, see Table 1.

The first explicit construction for this setting obtained $+4$ emulators of size $O(n^{4/3})$ by Dor, Halperin and Zwick [11]. The question of whether sparser emulators exist for any constant additive stretch has been one of the most major open problems in the area. In their breakthrough result, Abboud and Bodwin [1] refuted this possibility by demonstrating that any emulator with $O(n^{4/3-\epsilon})$ edges might induce a polynomially large additive stretch of $\Omega(n^{\delta(\epsilon)})$, for any ϵ.

On the other side of the size vs. stretch tradeoff, additive emulators of *linear* size have in particular attracted a lot of attention over the years [18, 7, 10, 9, 1, 16, 2, 14, 17]. To this date, the best additive stretch known for linear size emulators is $\widetilde{O}(n^{1/4})$, as shown (implicitly) by an earlier work of Pettie [18]. Bodwin and Vassilevska Williams [10] designed linear-size spanners and emulators with additive stretch of $+\widetilde{O}(\sqrt{n})$ (resp., $+\widetilde{O}(n^{1/3})$). In a follow-up work [9] , they cleverly improved the spanner's stretch to the state-of-the-art bound of $+\widetilde{O}(n^{3/7})$; Unfortunately, their improved spanner constructions do not seem to imply improved bounds for emulators, and Pettie's result [18] remains the state-of-the-art.

In this paper we focus on the following basic graph compression problem which despite all efforts is still fairly open:

▶ **Question 1.1.** What is the minimal additive error that can be achieved with linear space?

This question on its various forms (e.g., spanners, emulators) has been raised in many of the prior work on the topic, see e.g., [10, 3], especially in light of the "4/3 barrier" of [1]. Indeed in their seminal lower bound paper, Abboud and Bodwin [1] explicitly asked:

> *Our work shows that polynomial additive error must be suffered in order to obtain near-linear size compression of graphs. Given this, it is natural to wonder* how much *polynomial error is necessary to obtain compression in this regime.*

While not much progress has been provided on the upper bound side, there has been more movement on the lower bound aspects of the problem. Abboud and Bodwin showed that any linear size emulator must suffer $\Omega(n^{1/22})$ additive stretch, in the worst case. Huang and Pettie [16] improved this bound to $+\Omega(n^{1/18})$. This was further improved by Lu, Wein, Vassilevska Williams, and Xu [17] to $+\Omega(n^{2/29})$. Very recently, Bodwin and Hoppenworth [8] provided an $+\Omega(n^{1/7})$ stretch lower bound for linear spanners, by extending the known obstacle product framework to support also non-layered graphs.

Our new constructions are built upon modifying and extending the existing constructions for emulators with sublinear additive stretch and weighted additive spanners. While these notions have been studied before, our primary conceptual contribution is in demonstrating their usefulness for computing additive emulators of unweighted graphs. We next discuss the prior work on each of these settings.

Sublinear additive stretch. Elkin and Peleg showed that the "4/3 barrier" could be broken if one allows a $(1+\epsilon)$ multiplicative stretch, in addition to a small additive stretch [15]. Thorup and Zwick gave an elegant construction of an $O(kn^{1+1/(2^{k+1}-1)})$-size emulator H with $O(1+\epsilon, O(k/\epsilon)^{k-1})$-type stretch[1]. Their emulator has the remarkable property that its stretch bound holds for every $\epsilon > 0$ *simultaneously*, as its size bound is independent in ϵ. For any distance d, choosing $\epsilon = k/d^{1/k}$ leads to an emulator with a sublinear additive stretch function $f(d) = d + O(d^{1-1/k} + 3^k)$. As noted in [2], an interesting open question is whether one can match this size-stretch tradeoff for spanners.

[1] I.e., for every $u, v \in V$, $\mathrm{dist}_H(u,v) \leq (1+\epsilon)\mathrm{dist}_G(u,v) + O(k/\epsilon)^{k-1}$.

Table 1 Upper and lower bounds for additive emulators. New bounds are marked in blue.

Emulator Size	Additive Stretch	Remark	Citation
$O(n^{3/2})$	2		[5]
$\widetilde{O}(n^{4/3})$	4		[11]
$\Omega(n^{1+1/k})$	$2k-1$		[20]
$O(n^{4/3-\epsilon})$	$\Omega(n^{\delta(\epsilon)})$		[1]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/2-3\epsilon/2})$	implicit	[7]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/3-2\epsilon/3})$		[10]
$\widetilde{O}(n^{1+\epsilon+o(1)})$	$O(n^{3/11-9\epsilon/11})$		[9]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/4-3\epsilon/4})$	implicit	[18]
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{1/5-3\epsilon/5})$	$\epsilon \geq 1/7$	**new**
$\widetilde{O}(n^{1+\epsilon})$	$O(n^{(25-87\epsilon)/112})$	$0 \leq \epsilon \leq 1/5$	**new**
$\widetilde{O}(n)$	$O(n^{2/9-1/1600-o(1)})$		**new**
$\widetilde{O}(n)$	$\Omega(n^{1/22})$		[1]
$\widetilde{O}(n)$	$\Omega(n^{1/18})$		[16]
$\widetilde{O}(n)$	$\Omega(n^{2/29})$		[17]

Weighted (near) additive stretch. Elkin, Gitlitz and Neiman [13] provided the first constructions of near-additive spanners for *weighted* graphs. Their algorithm extends the unweighted construction of near-additive spanners (e.g., by [15]) to provide stretch guarantees of $f(d) = (1+\epsilon)d + \beta W$ where W is the maximum edge weight. Ahmed et al. [3] extended the constructions of spanners with purely additive stretch to weighted graphs by an ingenious amortized argument (which plays a role in our constructions, as well). Consequently, they provide $+2W, +4W, +8W$ weighted spanners with $\widetilde{O}(n^{3/2})$, $\widetilde{O}(n^{7/5})$ and $\widetilde{O}(n^{4/3})$ edges, respectively. Elkin, Gitlitz and Neiman [12] improve the latter stretch bound to $(6+o(1))W$, nearly matching the unweighted result for $W = 1$. Note that the above mentioned constructions also provide a *local* stretch guarantee of $+\beta \cdot W_{s,t}$ for every s, t pair, where $W_{s,t}$ is the largest edge weight on an s-t shortest path.

1.1 New Results

We provide a positive progress for Question 1.1 by improving upon the long-standing bound of $+\widetilde{O}(n^{1/4})$ by Pettie [18] to an additive stretch of $+O(n^{0.222-o(1)})$. Our end result is:

▶ **Theorem 1.2.** *Any unweighted n-vertex graph $G = (V, E)$ admit a linear-size emulator with additive stretch $\widetilde{O}(n^{2/9-1/1600-o(1)})$.*

The main novel aspect of this result is in our approach, which draws an interesting connection between weighted additive emulators and unweighted emulators with polynomial additive stretch. The final additive bound $O(n^{0.222-o(1)})$ is obtained by taking a gradual approach, containing two major steps of optimizations.

To illustrate our new algorithmic approach, we start by presenting a very simple construction for recovering the state-of-the-art additive stretch of $+\widetilde{O}(n^{1/4})$. This construction is obtained by using in a black-box manner the recent constructions of *weighted additive*

spanners by Ahmed et al. [4] and Elkin, Gitlitz and Neiman [12] which provides an additive stretch $+\beta W$. While such an additive term might be undesirable in many settings, these constructions play a key role in providing additive emulators for *unweighted* graphs[2].

Improved Stretch vs. Size Bounds via Weighted Additive Emulators. A careful inspection of our $+\widetilde{O}(n^{1/4})$ additive construction reveals that our reduction yields in fact, a specialized *weighted* graph instance with several convenient properties. In particular, we enjoy the fact that our generated weighted graphs are in fact obtained from unweighted graphs, in the sense that the edge weights corresponds to distances, rather than being arbitrary. We then provide a designated construction of weighted additive emulators that takes advantage of these specialized weighted graph instances. This leads to a quite general construction which improves over the known bounds in the entire regime of sparsity, i.e., n to $n^{4/3}$:

▶ **Theorem 1.3.** *For any n-vertex graph $G = (V, E, \omega)$ where $\omega : E \to \{1, \ldots, W\}$ and $0 \leq \epsilon \leq \frac{1}{3}$, there exists a $+\widetilde{O}(W \cdot n^{f(\epsilon)})$ emulator H of size at most $\widetilde{O}(n^{1+\epsilon})$ where:*

$$f(\epsilon) = \begin{cases} (1-3\epsilon)/5 & \text{if } 1/7 \leq \epsilon \leq 1/3; \\ (9-31\epsilon)/40 & \text{if } 3/37 \leq \epsilon \leq 1/7; \\ (3-9\epsilon)/14 & \text{if } 1/15 \leq \epsilon \leq 3/37; \\ (25-87\epsilon)/112 & \text{if } 0 \leq \epsilon \leq 1/15. \end{cases}$$

Setting $\epsilon = 0$, provides a linear-size emulator with additive stretch $n^{25/112} \sim n^{0.223}$.

Discretization of the Thorup-Zwick (TZ) Emulator Construction. Our final emulator result of Theorem 1.2 is based on a rather involved discretization of the TZ emulator construction adapted for weighted graphs. The following (quite technically to state) result serves as the core component of the final linear-size emulator:

▶ **Theorem 1.4.** *For every n-vertex unweighted $G = (V, E)$ a constant integer $k \geq 3$ and integer $D \geq 1$, one can compute an emulator H with additive stretch $O(D^{1-1/(k-1)} \log n)$ for any distance $d = O(D \cdot \log n)$. The size of H is bounded by*

$$\widetilde{O}(n^{1+1/(2^{k+1}-1)} + n^{1+1/(2^k-1)}/(D^{(2^k-2k)/((2^k-1)k(k-1))}).$$

This should be compared with the original TZ construction that provides an additive stretch of $d^{1-1/k}$ using $n^{1+1/(2^{k+1}-1)}$ edges. Theorem 1.4 can also be shown to imply that the stretch function of the TZ emulator is optimal only for a restricted regime of distances. In particular, with a size bound of $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$, one can provide pairs at distances $d \geq n^{k^2/2^k}$ an additive stretch of $O(d^{1-1/(k-1)})$, rather than $O(d^{1-1/k})$ as provided by the TZ bounds, which might be of independent interest.

1.2 Technical Overview

Our $+\widetilde{O}(n^{0.222})$-additive linear-size emulator is obtained in a sequence of two intermediate results, that gradually take advantage of several interesting degrees of freedom in the current constructions of weighted (near) additive emulators. Our technique exhibits several directions

[2] While our constructions utilizes the $+\beta W$ stretch guarantees, it is unclear if the local $+\beta W_{s,t}$ stretch guarantees can be useful in our context, as well.

of optimizations in the emulator framework of Thorup and Zwick [19], which become useful in the context of designing additive emulators with a small polynomial stretch. Note that while all our constructions are implemented in polynomial time, in this paper we put emphasis on the stretch vs. size tradeoff.

Beginner: $+\widetilde{O}(n^{1/4})$ Additive Stretch. As a warmup to our approach, we provide in Sec. 2 a new proof technique to obtain $+\widetilde{O}(n^{1/4})$ emulators of linear size, which simplifies the (implicit) state-of-the-art construction of Pettie [18]. Interestingly, our argument follows immediately by the weighted $+O(W)$ additive spanners of Ahmed et al. [4] and Gitlitz, Elkin, Neiman [12] with $\widetilde{O}(n^{4/3})$ edges, where W is the maximum edge weight of the graph. This provides the starting indication for the potential connection between weighted additive emulators and purely additive emulators for unweighted graphs.

On a high level, the construction works by computing a weighted *net* graph G' for the given (unweighted) graph G, obtained by sampling each G-vertex independently with probability of $\Theta(1/n^{1/4})$. The edges of G' connect every pair of sampled vertices u, v provided that their G-distance is at most $\Theta(n^{1/4}\log n)$. The net edges are weighted by the G-distance between their endpoints. The output emulator is union of two spanners: (i) a $O(\log n)$ multiplicative spanner for G (see Lemma 1.7), and (ii) a $+O(W)$ additive spanner for G' where $W = \Theta(n^{1/4}\log n)$. It is easy to see that the size bound is (near) linear[3]. The stretch argument for nearby pairs u, v at G-distance $O(n^{1/4}\log n)$ follows by the addition of the $O(\log n)$ multiplicative spanner. The argument for distant pairs $\Omega(n^{1/4}\log n)$ follows by using the $+O(W)$ additive spanner for G'.

Intermediate : $+\widetilde{O}(n^{0.223})$ Additive Stretch. The essence of the above mentioned construction is to employ on a weighted additive algorithm on the computed (weighted) net graph G', in a black box manner. Our starting observation, to break the current $+\Theta(n^{1/4})$ barrier, is the following: while G' is indeed a weighted graph, it is obtained from a given *unweighted* base graph G. Therefore it might be possible to treat G' better than any arbitrary input weighted graph. More specifically, by including the TZ emulator for G, one can provide a *sublinear* stretch guarantee for any neighboring pairs in G'. This, in principle, is impossible, for general weighted graphs. Since the sublinear stretch guarantees of the TZ emulators require a superlinear size bound, we cannot employ them directly on G, but rather on a subsampled *net* of G. This sub-sampling immediately converts the unweighted input instance into a weighted instance. We therefore conclude that the key task should be concerned with providing sparse constructions of *weighted additive* emulators. Our core construction computes a superlinear-size emulator for any weighted graph whose weighted stretch and size guarantees depend on the input integer parameters D, k, as follows:

▶ **Theorem 1.5.** *Any n-vertex graph $G = (V, E, \omega)$ with max weight W and integers $k \geq 2, D \geq 1$ admits a $+O(WD)$ emulator of size $\widetilde{O}(n^{1+1/(2^{k+1}-1)} + n^{4/3}/(D^{4/3} + 2/(3k)))$.*

Theorem 1.5 serves as the key technical step in providing the improved additive stretch vs. size bounds in almost the entire regime of parameters (see Theorem 1.3). In particular, by using a suitable pre-sampling of a net graph G' and applying Thm. 1.5 on G', we obtain linear-size emulator with $+\widetilde{O}(n^{25/112})$ stretch. Moreover, for any $\epsilon > 1/7$, Thm. 1.5 allows us to provide an $+n^{f(\epsilon)}$ emulator with $n^{1+\epsilon}$ edges, where $f(\epsilon) = 1/5 - 3\epsilon/5$. This improves the state-of-the-art bounds of $1/4 - 3\epsilon/4$ due to [18].

[3] One can make it linear by reducing the sampling probability an $O(\log n)$ factor.

We prove 1.5 by presenting a three-step algorithm. The first step (which takes care of the short distances) include a weighted variant of the TZ emulators which for integer stretch k provides $f(d)$-emulator with $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ edges and $f(d) = d + d^{1-1/k}W^{1/k}$ for $d \geq W$ and $f(d) = O(W)$, otherwise. This variant can be obtained by a straightforward adaptation of the TZ construction to the weighted setting. In particular, setting $W = 1$ recovers the TZ bounds (see Thm. 1.8).

The second step is based on the useful tool of *light-initialization* introduced by Ahemd et al. [3] in the context of translating the existing constructions of additive spanners for unweighted graphs into suitable constructions for weighted graphs. For a given weighted graph G and integer parameter t, the t light-initialization is a subgraph H' of G containing the t-lightest (based on edge weight) edges[4] incident to each vertex in G. Ahemd et al. [3] provided a very elegant argument that in essence achieves the same net effect as obtained in the unweighted setting (where one simply adds t arbitrary edges per vertex): Specifically, the key property is that any u-v shortest path P that has misses ℓ edges in H' must contains $\Omega(t\ell)$ vertices that are incident to the vertices of P via the edges of H'. Our algorithm employs the light-initialization tool on sampled net G' of G, for a carefully chosen parameter t. Each G-vertx is sampled into the net G' with probability $\Theta(\log n/D)$. The third and last step further sub-samples the vertices of G' and adds the complete weighted graph on this sample to the output spanner.

The stretch analysis of this scheme has the following structure. First, using the TZ emulators allows us to satisfy the stretch for pairs at distance $O(WD\log n)$ in G. The focus is then on bounding the stretch for a pair of sampled vertices $u, v \in V(G')$. The argument considers a u-v shortest path P in G' and distinguishes between two cases: $|P \setminus H| \leq q$ for some chosen parameter q, and the complementarity case where $|P \setminus H| > q$. For the first case, we use the weighted-TZ spanner of G to obtain a sublinear stretch guarantee for every edge on $P \setminus H$, taking advantage of the fact that each such edge corresponds to a path in the original graph G. The benefit that we get from the sublinear stretch bounds allows us to accumulate it for each of the t missing edges.

To handle the complementary case where $|P \setminus H| > q$, we use H' to claim that the final sampled set V'' contains a pair u'', v'' that are sufficiently close to u and v. The stretch bound is provided by the addition of the edge (u'', v'') to the final emulator.

Advanced: $+\widetilde{O}(n^{0.222})$ Additive Stretch. Our last and most involved improvement performs a *root treatment* to the TZ emulator construction. Instead of using the weighted-TZ variant in a black-box manner on our weighted sampled graph, we provide a discretization variant for this algorithm in which we replace the continuous TZ stretch function by a step function. The latter provides worse bounds for nearby pairs, with the benefit of using fewer edges. More specifically, the construction is parameterized by integers D, k, p and show:

▶ **Theorem 1.6.** *For every n-vertex $G = (V, E, \omega)$ with maximum weight W, a constant integer $k \geq 3$, integer $D \geq 1$ and $p \in (0, 1)$, one can compute an emulator H with additive stretch $O(D^{1-1/(k-1)} \cdot W \log n/p)$ for any distance $d = O(D \cdot W \log n/p)$. The size of H is bounded by*

$$|H| = \widetilde{O}\left(n^{1+\frac{1}{2^{k+1}-1}} + (n \cdot p)^{1+\frac{1}{2^k-1}} / \left((1/p)^{\frac{2^k-2}{(2^k-1)k}} \cdot D^{\frac{2^k-2k}{(2^k-1)k(k-1)}}\right)\right) .$$

[4] Each vertex sorts its incident edges in increasing edge weight, and the t first edges in this ordering are taken.

Our optimized variant of the weighted TZ emulators is fitted to the setting where the given weighted graph provided as input to Theorem 1.6 is in fact a net graph G' that corresponds to some unweighted base graph[5] G. We then aim at exploiting the fact that the edges of G' corresponds to G-paths already in the construction of the weighted TZ emulators. To present our key ideas, we briefly describe the TZ algorithm. For a subset of vertices V' and probability q, let $V'[q]$ be the set of vertices obtained by sampling each vertex $v \in V'$ independently with probability of q.

For a given parameter k, the algorithm computes a hierarchy $V = V_0 \supset V_1 \supset V_2 \supset \ldots \supset V_{k-1}$ of levels, where $V_i = V_{i-1}[q_{i-1}]$ for $q_{i-1} = |V_{i-1}|/n^{1+1/(2^k-1)}$. For every vertex $v \in V_i$, its $(i+1)^{th}$ pivot $p_{i+1}(v)$ is the closest vertex to v in V_{i+1}. The bunch $B_i(v)$ contains all vertices in V_i that are closer to v than its pivot $p_{i+1}(v)$. The algorithm adds to the emulator the edges between each $v \in V_i$ to all vertices in its bunch $B_i(v)$. The weights of the edges are the G-distance of their endpoints. This is done for every $i \in \{1, \ldots, k-2\}$. Finally, all edges in $V_{k-1} \times V_{k-1}$ are added to the emulator.

Our adaptation to weighted net graphs G' (whose edges correspond to paths in a base graph G) computes a hierarchy of $2(k-1)$ levels: $V = V_{1/2} \supseteq V_1 \supseteq V_{3/2} \supseteq V_2 \supseteq \ldots \supseteq V_{k-1/2}$, where $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$ for every $j \in \{1, \ldots, 2(k-1)\}$. Hence, we have $k-1$ integral levels $V_1, \ldots, V_{k-1}$ and k "half"-levels $V_{1/2}, V_{3/2}, \ldots, V_{k-1/2}$. Intuitively, the "half" levels represent an intermediate step that re-scales the "aggregate" benefit obtained by the existence of the precomputed emulator H_0 that takes care of the short distances in G'. The selection of the sampling probabilities are made in a careful manner that depend on the properties of H_0. Once the hierarchy is computed, we have $k-1$ steps which mimic the TZ algorithm with one main distinction, we add to the emulator edges from the *half*-level $V_{i+1/2}$ to the next *integral*-level V_{i+1}.

That is, for every $i \in \{0, \ldots, k-2\}$ and every $u \in V_{i+0.5}$, the algorithm computes a pivot $p_{i+1}(v)$ (closest vertex in V_{i+1}) and a bunch $B_{i+0.5}(u)$, which consists of all $V_{i+0.5}$ vertices that are closer to u than its pivot $p_{i+1}(u)$. The edges in $\{u\} \times B_{i+0.5}(u)$ are added to the emulator. Finally, in the last half level $k-0.5$, we add all edges in $V_{k-0.5} \times V_{k-0.5}$.

Remark. We note that our approach for computing improved linear emulators of Thm. 1.2 can also be used to improve the general tradeoff provided in Thm. 1.3. The total improvement, however, is limited to a small $o(1)$ additive term, and therefore we make this extra effort only for linear size emulators. We also note that our approach for the latter could be further optimized by considering a large number of recursive sampling steps, but again the net effect on the stretch is negligible (in particular, an additional sampling step might reduce the stretch by an 0.0001 additive term).

Notations. For a possibly weighted graph G, let $\text{dist}_G(u, v)$ be the *length* of a shortest path from u to v. The length of a shortest-path Q is measured by the sum of its weighted edges. Let $|Q|$ be the number of edges on this path. We use $\widetilde{O}(\cdot)$ notation to hide polylogarithmic factors in n. For a set of elements X and $p \in [0, 1]$, let $X[p]$ be the set obtained by sampling each X-element independently with probability p.

For a given (possibly) weighted graph G and integer t, a subgraph $H \subseteq G$ is a *t-spanner* if $\text{dist}_H(u, v) \leq t \cdot \text{dist}_G(u, v)$ for every $u, v \in V$. Our constructions use the following algorithm as a subroutine, mainly for $t = O(\log n)$.

[5] I.e., in our constructions, the graph G provided as input to Theorem 1.6 is in fact a net graph G' of some base graph G.

▶ **Lemma 1.7** ([6]). *For every n-vertex (possibly weighted) graph G and a given integer $k \geq 1$, one can compute a $(2k-1)$-spanner $H \subseteq G$ with $|H| \leq n^{1+1/k}$ edges.*

▶ **Theorem 1.8** ([19]). *For every n-vertex unweighted graph G and a given integer $k \geq 1$, one can compute an emulator H with $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ edges, such that for every $u, v \in V$, it holds that $\mathrm{dist}_G(u,v) \leq \mathrm{dist}_H(u,v) \leq \mathrm{dist}_G(u,v) + (\mathrm{dist}_G(u,v))^{1-1/k}$.*

Roadmap. In Sec. 2, we present a simple approach to recover the state-of-the-art bound of $+\widetilde{O}(n^{1/4})$ additive emulator. Sec. 3 provides an improved emulator construction for the entire regime, proving Theorem 1.5 and consequently also Thm. 1.3. Finally, in Sec. 4 we provide the proof of the key result, Thm. 1.2.

2 Warmup: $+\widetilde{O}(n^{1/4})$ Linear Emulators

We start by presenting a simple construction of linear size $+\widetilde{O}(n^{1/4})$-emulators, which uses the following theorem for weighted additive spanners by [4] (recently improved by [12]).

▶ **Theorem 2.1** (Theorem 3 in [4]). *Any n-vertex weighted graph $G = (V, E, \omega)$ with max edge weight W admits a $+8W$ additive spanner $H \subseteq G$ with $O(n^{4/3})$ edges.*

Algorithm. The algorithm for computing $+\widetilde{O}(n^{1/4})$-emulator has two steps. The first step computes a $O(\log n)$-multiplicative spanner $H_1 \subseteq G$, which as we show later handles the short distances in G. The second step computes a net graph $G' = (V', E', \omega')$ defined over a sampled subset $V' = V[p]$ for $p = \log n / n^{1/4}$. The edge set E' consists of all pairs in $V' \times V'$ whose distance in G is at most $n^{1/4}$. The weights of the E' are taken to be the G-distances. Formally, $E' = \{(x,y) \in V' \times V' \mid \mathrm{dist}_G(x,y) \leq n^{1/4}\}$, $\omega((x,y)) = \mathrm{dist}_G(x,y), \forall (x,y) \in E'$. Note that, by definition, the maximum weight W' of G' is $O(n^{1/4})$. The algorithm then applies Theorem 2.1 to compute $+8W'$ emulator H_2 for G'. The output emulator is given by $H = H_1 \cup H_2$. This completes the description of the algorithm.

The size analysis is immediate as w.h.p. $|V'| = O(n^{3/4} \log n)$ and thus by Theorem 2.1 $|H_2| = \widetilde{O}(n)$. We now consider the stretch argument. Fix $u, v \in V$. Assume first that $\mathrm{dist}_G(u,v) \leq c \cdot n^{1/4}$ for some constant c. Then, by including the $O(\log n)$-multiplicative spanner H_1, we have that $\mathrm{dist}_H(u,v) = O(n^{1/4} \log n)$, as desired.

Consider the complementary case where $\mathrm{dist}_G(u,v) > c \cdot n^{1/4}$, and let P be a u-v shortest path in G. Let P', P'' be the $n^{1/4}$-length prefix (resp., suffix) of P. By the Chernoff bound, w.h.p., we have that there exists a sampled vertex $u' \in P' \cap V'$ and $v' \in P'' \cap V'$. By the previous argument (for short distances), it remains to show that $\mathrm{dist}_H(u', v') \leq \mathrm{dist}_G(u', v') + O(n^{1/4} \cdot \log n)$.

Observe that since every $n^{1/4}$-length consecutive segment on P contains, w.h.p., a sampled vertex in V', we have that $\mathrm{dist}_{G'}(u', v') = \mathrm{dist}_G(u', v')$. By the properties of H_2, we then have that $\mathrm{dist}_{H_2}(u', v') \leq \mathrm{dist}_{G'}(u', v') + 8W' = \mathrm{dist}_G(u', v') + 8n^{1/4}$. Overall, we have

$$\begin{aligned} \mathrm{dist}_H(u,v) &\leq \mathrm{dist}_{H_1}(u,u') + \mathrm{dist}_{H_2}(u',v') + \mathrm{dist}_{H_1}(v',v) \\ &\leq O(\log n)(\mathrm{dist}_G(u,u') + \mathrm{dist}_G(v',v)) + \mathrm{dist}_G(u',v') + 8n^{1/4} \\ &= \mathrm{dist}_G(u,v) + O(n^{1/4} \cdot \log n) \ . \end{aligned}$$

3 New Weighted Additive Emulators

3.1 The Core Construction

We start by presenting the key construction which for n-vertex weighted graphs provides emulators with $+O(WD)$ stretch and with $O(n^{4/3}/f(D))$ edges, for some monotone increasing function $f(\cdot)$. These emulators serve the basis for improved emulator constructions in wide range of parameters, and in particular computing linear emulators with improved additive stretch $+n^{0.222}$. We show:

▶ **Theorem 3.1.** *There is an algorithm* SuperLinEmulator *that given any n-vertex graph $G=(V,E,\omega)$ with maximum weight W, and integers $k\geq 2, D\geq 1$ computes a $+O(WD)$ emulator H of size $\widetilde{O}\left(n^{1+1/(2^{k+1}-1)}+\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.*

We start by presenting the two main tools used by Algorithm SuperLinEmulator.

Tool I: Weighted Near-Additive Emulator. We used the following adaptation of the Thorup and Zwick emulators to the weighted setting. An adaptation for universal emulators has been recently provided by Elkin, Gitlitz and Neiman [13]. In the full version, we show:

▶ **Lemma 3.2.** *There is an algorithm* WeightedTZEmulator *that for any n-vertex graph $G=(V,E,\omega)$ with maximum weight W, and any fixed integer $k\geq 2$ computes an $+f(d)$ emulator H of size $O(n^{1+1/(2^{k+1}-1)})$, where $f(d)=d+O(d^{1-1/k}\cdot W^{1/k})$ for any distance $d>W$, and $f(d)=d+O(W)$ for $d\leq W$.*

Tool II: Light Initialization. A t-light initialization of a weighted graph $G=(V,E,\omega)$, introduced by Ahmed et al. [4], is a subgraph $H\subseteq G$ obtained by including the t lightest edges incident to each vertex v (or all its edges when $\deg(v)\leq t$). Edge weight ties can be broken arbitrarily; Let Initialization be the algorithm that given the graph G and a parameter t, outputs the t-light initialization subgraph H. We say that v is a t-light neighbor of u if the edge (u,v) is among the t-lightest edges incident on u.

▶ **Theorem 3.3** (Theorem 5 in [4])**.** *Let $G=(V,E,\omega)$ be an undirected weighted graph and let $H=$ Initialization(G,t) for some input integer t. Then, for every shortest path $P_{u,v}$ that is missing ℓ edges in H (i.e., $|P_{u,v}\setminus H|=\ell$), there is a set of vertices $S\subseteq V$ such that (i) $|S|=\Omega(t\cdot\ell)$ and (ii) for every $a\in S$, there is a vertex $b\in P_{u,v}$ satisfying that a is a t-light neighbor of b.*

Tool III: Algorithm Net. Given an n-vertex weighted graph $G=(V,E,\omega)$ with maximum edge weight W and a probability $p\in(0,1)$, the algorithm Net(G,p) outputs a graph $G'=(V',E',\omega')$, denoted as a *net*, defined as follows. Let $V'=V[p]$ be a random sample of V, obtained by sampling each $v\in V$ independently with probability of p. Let $E'=\{(u,v)\in V'\times V' \mid \mathrm{dist}_G(u,v)\leq\Theta(\log n/p)\cdot W\}$ and $\omega'((u,v))=\mathrm{dist}_G(u,v)$ for every $(u,v)\in E'$. We use the following observation in our constructions:

▶ **Observation 3.4.** *Let $G'=(V',E',\omega')$ be the output net graph of Alg.* Net(G,p) *where $G=(V,E,\omega)$ is an n-vertex graph with maximum edge weight W. Then w.h.p., the following holds: (i) $|V'|=O(np\log n)$, (ii) for every $u,v\in V'$, $\mathrm{dist}_{G'}(u,v)=\mathrm{dist}_G(u,v)$, and (iii) the maximum edge weight of G' is bounded by $W'=\Theta(W\log n/p)$.*

Description of Alg. SuperLinEmulator. The algorithm has three main steps, each computes an emulator graph H_1, H_2, H_3 whose union provides the desired emulator. The first emulator H_1 is obtained by computing the weighted-variant of the Thorup-Zwick emulator using Lemma 3.2. As we will see in the analysis, this would provide the desired stretch for short distances. The second emulator H_2 is obtained by computing the t-initialization of some net graph G_2 for $t = n^{1/3}/D^{(1+2/k)/3}$. The net G_2 is defined by sampling a subset of vertices $V_2 = V[p_1]$ for $p_1 = 10 \log n/D$. The edges E_2 of the net G_2 are defined by connecting each pair $(u, v) \in V_2 \times V_2$ provided that $\text{dist}_G(u, v) \leq WD$, every edge (u, v) in G_2 is then weighted by the G-distance between its endpoints. Finally, the last emulator graph H_3 is obtained by adding all the weighted edges between a sampled set $V_3 = V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$. The weights are taken to be the G-distances between the endpoints. This completes the algorithmic description.

Algorithm 1 SuperLinEmulator(G, k, D).

Input: Graph $G = (V, E, \omega)$ with maximum edge weight W, integers k, D.
Output: A $+O(W \cdot D)$ emulator H of size $\widetilde{O}\left(n^{1+1/(2^{k+1}-1)} + \frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.

1. $H_1 \leftarrow$ WeightedTZEmulator(G, k) (using Lemma 3.2).
2. Let $G_2 = (V_2, E_2, \omega_2) \leftarrow$ Net(G, p_1) for $p_1 = 10 \log n/D$.
3. $H_2 \leftarrow$ Initialization(G_2, t) for $t = n^{1/3}/D^{(1+2/k)/3}$ (using Thm. 3.3).
4. Let $V_3 \leftarrow V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$;
5. Set $H_3 \leftarrow (V_3, V_3 \times V_3, \omega_3)$ where $\omega_3((u, v)) = \text{dist}_G(u, v)$ for every $(u, v) \in H_3$.
6. Output $H \leftarrow H_1 \cup H_2 \cup H_3$.

Size analysis. By Lemma 3.2, $|H_1| = O(n^{1+1/(2^{k+1}-1)})$. By the Chernoff bound, w.h.p $|V_2| = n \cdot p_1$ and $|H_2| = t \cdot |V_2| = \widetilde{O}\left(\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$. Finally, by the Chernoff bound, w.h.p, $|V_3| = |V_2| \cdot p_2$, and as $|H_3| = |V_3|^2$, we also get that $|H_3| = \widetilde{O}\left(\frac{n^{4/3}}{D^{4/3+2/(3k)}}\right)$.

Stretch analysis. We prove the following somewhat stronger lemma.

▶ **Lemma 3.5.** *Let H' be an emulator for G with maximum edge weight W such that for any u, v pair at G-distance at most WD, it holds that $\text{dist}_{H'}(u, v) \leq \text{dist}_G(u, v) + O(WD^{1-1/k})$. Then, for* every $u, v \in V$*, it holds that $\text{dist}_{H' \cup H_2 \cup H_3}(u, v) \leq \text{dist}_G(u, v) + O(WD)$.*

By Lemma 3.2, we then have that $\text{dist}_{H_1}(u, v) \leq \text{dist}_G(u, v) + O(WD^{1-1/k})$ for every u, v pair at G-distance at most WD, hence by taking $H' = H_1$, the stretch argument holds.

Proof of Lemma 3.5. Fix a pair $u, v \in V$ and first consider the simpler case where $\text{dist}_G(u, v) \leq WD$. By the properties of H', $\text{dist}_{H'}(u, v) \leq O(WD)$. From now on, we assume that $\text{dist}_G(u, v) > WD$. Let $P_{u,v}$ be the u-v shortest path in G, and let u', v' be a sampled vertex in V_2 in the $D/4$-hop prefix (resp., suffix) of the path. By the above, we have that $\text{dist}_G(u, u') \leq WD$ and $\text{dist}_G(v', v) \leq WD$, and therefore H' provides an additive $+O(WD)$ term for each of these distances.

Our next goal is to bound the u'-v' distance in H where $u', v' \in V_2$. It is easy to see that w.h.p., $\text{dist}_{G_2}(u', v') = \text{dist}_G(u', v')$, since each D-hop segment on the $P_{u,v}$ path contains a sampled vertex in V_2. Let P' be a u'-v' shortest path in G_2. We distinguish between two cases depending on the number of edges in $P_2 \setminus H_2$.

Case 1: $|P' \setminus H_2| \leq D^{1/k}$. Each edge (x, y) in $P' \subseteq G_2$ corresponds to an x-y shortest path where $\text{dist}_G(x, y) \leq WD$. Using the H' emulator, we have that for each such edge $(x, y) \in G_2$, $\text{dist}_{H'}(x, y) \leq \text{dist}_G(u, v) + O(D^{1-1/k} \cdot W)$. Since there are at most $D^{1/k}$, the total additive stretch introduced due to these edges is bounded by $O(D^{1/k} \cdot D^{1-1/k} \cdot W) = O(WD)$, as required. This is the critical point where we exploit the fact that the weighted edges of G_2 correspond to short paths in G.

Case 2: $|P' \setminus H_2| > D^{1/k}$. We next turn to consider the case where H_2 misses many edges from P'. Here we will exploit the expansion property guaranteed by the addition of the t-light initialization. Let P_1 (resp., P_2) be a prefix (resp., suffix) of P' for which H_2 misses exactly $D^{1/k}/2$. I.e., $|P_i \setminus H_2| = D^{1/k}/2$ for $i \in \{1, 2\}$. By Theorem 3.3 the following claims holds for every $i \in \{1, 2\}$: There exists a subset $S_i \subseteq V_2$ such that (i) $|S_i| = \Omega(t \cdot D^{1/k})$ and (ii) for every $a \in S_i$, there is a vertex $b_i \in P_i$ such that a is t-light neighbor of b_i. By the value of p_2, we get that w.h.p., there exists $s_i \in S_i \cap V_3$ for every $i \in \{1, 2\}$. Therefore, the emulator H_3 contains the edge (s_1, s_2) with weight $\text{dist}_G(s_1, s_2)$.

Since the maximum edge weight in G_2 is at most $W_2 = WD$, and $(b_i, s_i) \in H_2$, we have that $\text{dist}_H(b_1, s_1) + \text{dist}_H(s_2, b_2) = O(WD)$. By the triangle inequality,

$$\text{dist}_G(s_1, s_2) \leq \text{dist}_G(b_1, b_2) + O(WD) \ . \tag{3.1}$$

Since the segments $P'[u, b_1]$ and $P'[b_2, v]$, each has at most $D^{1/k}$ missing edges in H. Therefore, by applying the argument for Case 1, we have:

$$\text{dist}_H(u, b_i) \leq \text{dist}_G(u, b_i) + O(WD), \text{ for } i \in \{1, 2\} \ . \tag{3.2}$$

We are now ready to complete the stretch argument by showing:

$$\begin{aligned} \text{dist}_H(u', v') &\leq \text{dist}_H(u', b_1) + \text{dist}_H(b_1, s_1) + \text{dist}_H(s_1, s_2) + \text{dist}_H(s_2, b_2) + \text{dist}_H(b_2, v') \\ &\leq \text{dist}_G(u', b_1) + \text{dist}_G(b_1, b_2) + \text{dist}_G(b_2, v') + O(WD) \ , \end{aligned}$$

where the inequalities follow by plugging Eq. (3.1,3.2), and using the fact that as $(s_1, s_2) \in H_3$, by the triangle inequality $\text{dist}_H(s_1, s_2) = \text{dist}_G(s_1, s_2) \leq \text{dist}_G(b_1, b_2) + O(WD)$. ◀

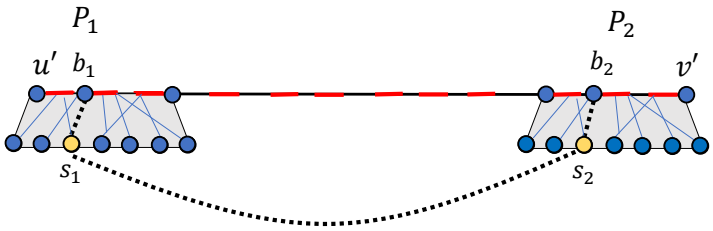

Figure 1 An illustration for the stretch argument of Alg. SuperLinEmulator. Shown is a u'-v' shortest path $P' \subseteq G_2$, the segments P_1, P_2 each containing $D^{1/k}/2$ missing edges w.r.t H_2. By the properties of the t-initialization procedure, each these segments contains a vertex b_1, b_2 with at least one sampled t-light neighbor, s_1, s_2. The added weighted edge (s_1, s_2) establishes the stretch guarantees.

3.2 Improved Additive Emulators

Our improved additive stretch bounds are provided by using Theorem 3.1 with two sparsity bounds determined by $k = 2, 3$. We have:

▶ **Corollary 3.6.** *For any n-vertex graph $G = (V, E)$ with max weight W, there exists a:*
1. *$+O(W \cdot n^{4/35})$ emulator of size $\widetilde{O}(n^{8/7})$, and*
2. *$+O(W \cdot n^{6/35})$ emulator of size $\widetilde{O}(n^{16/15})$.*

Proof. (1) follows by setting $k = 2$ and $D = n^{4/35}$ in Theorem 3.1, and (2) follows by setting $k = 3$ and $D = n^{6/35}$ in Theorem 3.1. ◀

Using these two emulator constructions, we show an improved stretch vs. size tradeoff in almost the entire regime of interest.

Proof of Thm. 1.3 for $0 \leq \epsilon \leq 1/15$ and $3/37 \leq \epsilon \leq 1/7$. We describe Algorithm ImprovedEmulator which given $G = (V, E, \omega)$ and $\epsilon \in [0, 1/15] \cup (3/37, 1/7]$, computes the desired emulator. The algorithm starts by computing a $O(\log n)$ multiplicative spanner H_0, which as always, takes care of the short distances in G. Next, the algorithm computes a net graph G' whose bounds depends on the value of ϵ, as follows. Define:

$$k_\epsilon = \begin{cases} 3, & \text{for } \epsilon \in [0, 1/15], \\ 2, & \text{for } \epsilon \in (3/37, 1/7] \ . \end{cases} \tag{3.3}$$

Let $n_\epsilon = n^{(1-1/2^{k_\epsilon+1})(1+\epsilon)}$ and $q = n/n_\epsilon$. Then, the net graph G' is obtained by applying Alg. $\mathsf{Net}(G, p)$ for $p = 10 \log n/q$. Finally, it applies Alg. SuperLinEmulator with the input G', k_ϵ and $D = (n_\epsilon)^{2k_\epsilon/35}$. This results in the emulator H_1. The output emulator is given by $H = H_0 \cup H_1$.

Algorithm 2 ImprovedEmulator(G, ϵ).

Input: Graph $G = (V, E, \omega)$ with maximum weight W, $\epsilon \in [0, 1/15] \cup (3/37, 1/7]$.
Output: $+\widetilde{O}(W \cdot n^{f(\epsilon)})$ emulator H of size $\widetilde{O}(n^{1+\epsilon})$.
1. $H_0 \leftarrow \mathsf{MultSpanner}(G, O(\log n))$.
2. Let $n_\epsilon = n^{(1-1/2^{k_\epsilon+1})(1+\epsilon)}$ and $q = n/n_\epsilon$ (see Eq. (3.3)).
3. $(G' = (V', E', \omega')) \leftarrow \mathsf{Net}(G, p)$ for $p = 10 \log n/q$.
4. $H_1 \leftarrow \mathsf{SuperLinEmulator}(G', k_\epsilon, D)$ for $D = (n_\epsilon)^{2k_\epsilon/35}$.
5. Output $H_0 \cup H_1$.

Analysis. We start with a stretch argument for a fixed pair $u, v \in V$. First, assume the more interesting case where $u', v' \in V'$. By the properties of H_1, the additive stretch is:

$$\widetilde{O}(D \cdot q \cdot W) = \widetilde{O}(n_\epsilon)^{2k_\epsilon/35} \cdot n^{1/2^{k_\epsilon+1}} \cdot n^{\epsilon(1/2^{k_\epsilon+1}-1)} \cdot W) = \widetilde{O}(W \cdot n^{f(\epsilon)}) \ . \tag{3.4}$$

Next assume that $\mathrm{dist}_G(u, v) \leq W \cdot q$. By adding the multiplicative spanner H_0, we have $\mathrm{dist}_{H_0}(u, v) \leq O(Wq \log n)$. Finally, assume that $\mathrm{dist}_G(u, v) \geq Wq$ and let $u', v' \in V'$ be the closest sampled vertex to u (resp., v) on the u-v shortest path. W.h.p., $\mathrm{dist}_G(u, u'), \mathrm{dist}_G(v, v') \leq Wq$ and therefore, $\mathrm{dist}_{H_0}(u, u'), \mathrm{dist}_{H_0}(v, v') \leq O(Wq \log n)$. Since w.h.p. $\mathrm{dist}_G(u', v') = \mathrm{dist}_{G'}(u', v')$, the stretch argument is completed by Eq. (3.4). The size bound follows by plugging $|H_0| = \widetilde{O}(n)$, and moreover, $|H_1| = \widetilde{O}(n^{1+\epsilon})$ by Corollary 3.6. We are now ready to complete the proof for the missing regimes.

Complete proof of Thm. 1.3. For the range $1/7 \leq \epsilon \leq 1/3$, the proof follows by letting $H = \mathsf{SuperLinEmulator}(G, k = 2, D)$ for $D = n^{(1-3\epsilon)/5}$. For the range $1/15 \leq \epsilon \leq 3/37$, the proof follows by letting $H = \mathsf{SuperLinEmulator}(G, k = 3, D)$ for $D = n^{(3-9\epsilon)/14}$. ◀

4 $+n^{0.222}$ Emulator of Linear Size

4.1 An Optimized Weighted Thorup-Zwick Emulator

In this section, we present an optimized variant of Thorup-Zwick that plays a key role in the construction of our linear additive emulator. We prove the following theorem which in particular implies Thm. 1.4.

▶ **Theorem 4.1.** *For every n-vertex $G = (V, E, \omega)$ with maximum weight W, a constant integer $k \geq 3$, integer $D \geq 1$ and $p \in (0, 1)$, there is an Algorithm* ImprovedTZEmulator *for computing an emulator H with additive stretch $O(D^{1-1/(k-1)} \cdot W \log n/p)$ for any distance $d = O(D \cdot W \log n/p)$. The size of H is bounded by*

$$|H| = \widetilde{O}\left(n^{1+\frac{1}{2^{k+1}-1}} + (n \cdot p)^{1+\frac{1}{2^k-1}} / \left((1/p)^{\frac{2^k-2}{(2^k-1)k}} \cdot D^{\frac{2^k-2k}{(2^k-1)k(k-1)}}\right)\right) .$$

We can also show interesting corollaries of Thm. 4.1 which demonstrate the sub-optimality of the TZ construction for a wide-range of distances. For example, the following holds:

▶ **Corollary 4.2.** *Every n-vertex unweighted graph G and given integer $k \geq 1$ admits an emulator H of size $\widetilde{O}(n^{1+1/(2^{k+1}-1)})$ such that pairs at distances $d \geq n^{k^2/2^k}$ have additive stretch of $O(d^{1-1/(k-1)})$.*

This should be compared with the additive stretch of $O(d^{1-1/k})$ provided by the TZ emulator (which also marks the state-of-the-art bounds). Thus, while the original TZ emulator is optimal for small distances as proven in [2], this optimality holds in a restricted range of distances, especially for non-constant values of k, e.g., $k = O(\log \log n)$. We now turn to prove Thm. 4.1 which constitutes the key technical contribution in the linear emulator construction.

Algorithm ImprovedTZEmulator. The algorithm starts by applying our weighted-variant of the Thorup-Zwick emulator to obtain $H_1 \leftarrow \mathsf{WeightedTZEmulator}(G, k)$, see Lemma 3.2. Next, it computes a net $G' = \mathsf{Net}(G, p)$ obtained by sampling each vertex in V into the net G' independently with probability p. By Obs. 3.4, we have that the maximum edge weight G' is bounded by $W' = \Theta(\log n \cdot W/p)$. In addition, w.h.p. it also holds that the G'-distances equal to the G-distances. The key technically involved step is in the computation of an additional emulator that we denoted by H_2 for G'. This emulator is computed by applying a new variant of the TZ emulator which takes advantageous of the fact that each weighted edge in G' corresponds to some path in a prior graph G, and more specifically, that there is a precomputed emulator (namely, H_1) that handles short distances in G'.

The construction builds a hierarchy $V = V_{1/2} \supseteq V_1 \supseteq V_{3/2} \supseteq V_2 \supseteq \ldots \supseteq V_{k-1/2}$, where $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$ for every $j \in \{1, \ldots, 2(k-1)\}$. Note that in contrast to the classic TZ emulator construction, our hierarchy has $2k-1$ levels: $k-1$ integral levels $V_1, \ldots, V_{k-1}$ and k "half"-levels $V_{1/2}, V_{3/2}, \ldots, V_{k-1/2}$. Intuitively, the "half" levels represent an intermediate

step that re-scales the extra-benefit obtained by the existence of the emulator H_1 (that takes care of short distances in G'). The definition of the sampling probabilities q_j is somewhat more involved compared to that of the classic construction. To define these probabilities, we need the following function definitions, for every integer $0 \leq i \leq k$:

$$h(i) = 1 - \frac{2^i - 1}{2^k - 1}, \qquad f(i) = \frac{2(2^k-1)i - 2k(2^i-1) - 2^k + 2^i}{(2^k-1)k} \quad \text{and} \quad g(i) = \frac{2^k - 2^i}{(2^k-1)k} \ . \tag{4.1}$$

The probabilities q_j for $j \in \{2i, 2i+1\}$ are chosen in order to satisfy the following, w.h.p., for every $i \in \{1, \ldots, k-1\}$:

$$|V_i| = \widetilde{O}\left(n^{h(i)} \cdot \Delta^{f(i)} \cdot \left(\frac{W'}{W}\right)^{g(i)}\right) \quad \text{and} \quad |V_{i+1/2}| = \widetilde{O}\left(|V_i| / \left(\Delta^{(i-1)/k} \cdot \left(\frac{W'}{W}\right)^{1/k}\right)\right) \ . \tag{4.2}$$

The sampling probabilities $q_{j=2i}$ for every $i \in \{1, \ldots, k-1\}$ have a simple to state expression:

$$q_{2i} = \Theta\left(\frac{\log n}{\Delta^{(i-1)/k} \cdot \left(\frac{W'}{W}\right)^{1/k}}\right) \ . \tag{4.3}$$

Let us give a concrete example for $k = 4$: For ease of notation, let $\widehat{W} = W'/W$.

1. $|V_{0.5}| = n$.
2. $|V_1| = \widetilde{O}\left(n^{14/15} \cdot \Delta^{2/15} \cdot \left(\widehat{W}\right)^{7/30}\right)$.
3. $|V_{1.5}| = \widetilde{O}\left(|V_1|/(\widehat{W})^{1/4}\right) = \widetilde{O}(n^{14/15} \cdot \Delta^{2/15} \cdot \left(\widehat{W}\right)^{7/30-1/4}\right)$.
4. $|V_2| = \widetilde{O}\left(n^{12/15} \cdot \Delta^{6/15} \cdot \left(\widehat{W}\right)^{1/5}\right)$.
5. $|V_{2.5}| = \widetilde{O}\left(|V_2|/(\Delta \cdot \widehat{W})^{1/4}\right) = \widetilde{O}\left(n^{12/15} \cdot \Delta^{6/15-1/4} \cdot \left(\widehat{W}\right)^{1/5-1/4}\right)$.
6. $|V_3| = \widetilde{O}\left(n^{8/15} \cdot \Delta^{13/30} \cdot \left(\widehat{W}\right)^{2/15}\right)$.
7. $|V_{3.5}| = \widetilde{O}\left(|V_3|/(\Delta^2 \cdot \widehat{W})^{1/4}\right) = \widetilde{O}\left(n^{8/15} \cdot \Delta^{13/30-1/2} \cdot \left(\widehat{W}\right)^{2/15-1/4}\right)$.

Given the q_j's probabilities, the algorithm proceeds in a very similar manner to the TZ emulator algorithm, with one main emphasis: There are $k-1$ phases in which we add to the emulator edges from the *half*-level $V_{i+1/2}$ to the next *integral*-level V_{i+1}. That is, no edges are added between V_{i+1} to $V_{i+1.5}$. For every $i \in \{0, \ldots, k-2\}$ and every $u \in V_{i+0.5}$, the algorithm computes a pivot $p_{i+1}(u)$ and a bunch $B_{i+0.5}(u)$, as follows. The pivot $p_{i+1}(u)$ is the closest[6] vertex to u in the next integral-level, V_{i+1}. The bunch $B_{i+0.5}(u)$ consists of all vertices in $V_{i+0.5}$ that are strictly closer to u than its pivot $p_{i+1}(u)$. The edges in $\{u\} \times B_{i+0.5}[u]$ are added to the emulator H_2, weighted by their G'-distances (which by Obs. 3.4(ii) also equal to the G-distances). Finally, all edges between the vertices in the last-half level $V_{k-0.5}$ are also added to H_2. The output emulator is given by $H_1 \cup H_2$.

The analysis is deferred to the full version.

[6] As usual, we can assume that the shortest-paths are unique.

Algorithm 3 ImprovedTZEmulator(G, k, p, D).

Input: Graph $G = (V, E, \omega)$ with maximum weight W and parameters $k \geq 3, D \geq 1$, $p \in (0, 1)$.
Output: $+O(D^{1-1/(k-1)} \cdot W \log n/p)$ emulator H for pairs at distance $d = O(D \cdot W \log n/p)$

1. $H_1 \leftarrow$ WeightedTZEmulator(G, k) (using Lemma 3.2), $H_2 \leftarrow \emptyset$.
2. $(G' = (V', E', \omega')) \leftarrow$ Net(G, p).
3. Set $\Delta = D^{1/(k-1)}$ and $V_{0.5} = V$.
4. For $j \in \{1, \ldots, 2(k-1)\}$ do: $V_{(j+1)/2} \leftarrow V_{j/2}[q_j]$, where q_j is defined based on Eq. (4.2).
5. For $i = 0$ to $k-2$ do:
   - For every $u \in V_{i+0.5}$ do:
     a. $p_{i+1}(u) = CLOSEST(V_{i+1}, u)$.
     b. $B_{i+0.5}(u) \leftarrow \{v \in V_{i+0.5} \mid \text{dist}_{G'}(u, v) < \text{dist}(u, p_{i+1}(u))\}$.
     c. $B_{i+0.5}[u] \leftarrow B_{i+0.5}(u) \cup \{p_{i+1}(u)\}$.
     d. $H_2 \leftarrow H_2 \cup (\{u\} \times B_{i+0.5}[u])$.
6. $H_2 \leftarrow H_2 \cup (V_{k-0.5} \times V_{k-0.5})$.
7. $H = H_1 \cup H_2$.

4.2 Improved Linear Emulators

This section is devoted to the proof of Theorem 1.2. Let ModifiedSuperLinEmulator be the same algorithm as SuperLinEmulator only that we omit the computation of H_1 in Step (1). We next present Algorithm ImprovedLinearEmulator that computes the desired emulators, as follows:

Algorithm 4 ImprovedLinearEmulator.

Input: An *unweighted* graph $G = (V, E)$ on n vertices.
Output: $+\widetilde{O}(n^{2/9-1/1600})$ emulator H of size $\widetilde{O}(n)$.
1. $H_0 \leftarrow$ MultSpanner$(G, k = O(\log n))$.
2. Set $p_1 = 10 \log n/n^{1/32}$, $p_2 = 10 \log n/n^{21/1060}$ and $D = n^{723/4240}$.
3. $(G_1 = (V_1, E_1, \omega_1)) \leftarrow$ Net(G, p_1).
4. $H_1' \leftarrow$ ImprovedTZEmulator$(G_1, k = 4, p_2, D = n^{723/4240})$.
5. $(G_2 = (V_2, E_2, \omega_2)) \leftarrow$ Net(G_1, p_2).
6. $H_2 \leftarrow$ ModifiedSuperLinEmulator$(G_2, k = 3, D)$.
7. Output $H_0 \cup H_1' \cup H_2$.

Algorithm 5 ModifiedSuperLinEmulator(G, k, D).

1. Let $G_2 = (V_2, E_2, \omega_2) \leftarrow$ Net(G, p_1) for $p_1 = 10 \log n/D$.
2. $H_2 \leftarrow$ Initialization(G_2, t) for $t = n^{1/3}/D^{(1+2/k)/3}$ (using Thm. 3.3).
3. Let $V_3 \leftarrow V_2[p_2]$ for $p_2 = 10 \log n/(t \cdot D^{1/k})$;
4. Set $H_3 \leftarrow (V_3, V_3 \times V_3, \omega_3)$ where $\omega_3((u, v)) = \text{dist}_G(u, v)$ for every $(u, v) \in H_3$.
5. Output $H \leftarrow H_2 \cup H_3$.

Size analysis. Clearly, $|H_0| = \widetilde{O}(n)$. By Theorem 4.1, we have that:

$$|H'_1| = \widetilde{O}\left((n \cdot p_1)^{1+\frac{1}{2^{4+1}-1}} + (n \cdot p_1 \cdot p_2)^{1+\frac{1}{2^4-1}} / \left((1/p_2)^{\frac{2^4-2}{(2^4-1)4}} \cdot D^{\frac{2^4-2\cdot 4}{(2^4-1)4(4-1)}}\right)\right) .$$

Therefore, $|H'_1| = \widetilde{O}(n^{(1-1/32-21/1060)(16/15)-(21/1060)\cdot(14/60)-(723/4240)\cdot(8/(15\cdot 4\cdot 3))}) = \widetilde{O}(n^1)$. Finally, by the proof of Thm. 3.1, it holds that $|H_2| = O((|V_2|)^{4/3}/D^{4/3+2/(3k)})$. Therefore,

$$|H_2| = (n \cdot p_1 \cdot p_2)^{4/3}/D^{4/3+2/9} = \widetilde{O}(n^{(1-1/32-21/1060)\cdot(4/3)-(4/3+2/9)\cdot(723/4240)}) = \widetilde{O}(n) .$$

Stretch analysis. Let W_1 (resp. W_2) be the maximum edge weight of G_1 (reps., G_2). By Obs. 3.4, we have that $W_1 = \widetilde{O}(1/p_1)$, $W_2 = \widetilde{O}(W_1 \cdot (1/p_2))$. We show that the additive stretch is $O(W_2 \cdot D) = O(n^{2/9-1/1600})$. By Obs. 3.4, the G_1-distances and G_2-distances, w.h.p., equal to the G-distances.

Case 1: Consider first a vertex pair $u', v' \in V_2$, we shall compute the stretch argument for the pair u', v'. By Lemma 3.5 with $H' = H'_1$ and $W = W_2$ we have that the additive stretch of the pair u', v' is given by:

$$\widetilde{O}(D \cdot W_2) = O(n^{2/9-1/1600}) . \tag{4.4}$$

Case 2: $\mathrm{dist}_G(u, v) \leq W_2$. By adding the multiplicative spanner H_0, $\mathrm{dist}_{H_0}(u, v) \leq O(W_2 \log n)$.

Case 3: $\mathrm{dist}_G(u, v) > W_2$. Let $u', v' \in V_2$ be the closest sampled vertex to u (resp., v) on the u-v shortest path in G. By the Chernoff bound, w.h.p., $\mathrm{dist}_G(u, u'), \mathrm{dist}_G(v, v') \leq W_2$ and therefore, by Case 2, $\mathrm{dist}_{H_0}(u, u'), \mathrm{dist}_{H_0}(v, v') \leq O(W_2 \log n)$. By Obs. 3.4, w.h.p., $\mathrm{dist}_G(u', v') = \mathrm{dist}_{G_2}(u', v')$, and the stretch argument is completed by Eq. (4.4) of Case 1.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 351–361. ACM, 2016.

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

3 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020.

4 Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen G. Kobourov, and Richard Spence. Weighted additive spanners. In Isolde Adler and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science – 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers*, volume 12301 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2020.

5 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev 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 John R. Gilbert and Rolf G. Karlsson, editors, *SWAT 90, 2nd Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 11-14, 1990, Proceedings*, volume 447 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 1990.

7 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (alpha, beta)-spanners. *ACM Trans. Algorithms*, 7(1):5:1–5:26, 2010.

8 Greg Bodwin and Gary Hoppenworth. New additive spanner lower bounds by an unlayered obstacle product. *CoRR*, abs/2207.11832, 2022. doi:10.48550/arXiv.2207.11832.

9 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 855–872. SIAM, 2016.

10 Gregory Bodwin and Virginia Vassilevska Williams. Very sparse additive spanners and emulators. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 377–382. ACM, 2015.

11 Dorit Dor, Shay Halperin, and Uri Zwick. All pairs almost shortest paths. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 452–461. IEEE Computer Society, 1996.

12 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Improved weighted additive spanners. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths with near-additive error in weighted graphs. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 23:1–23:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

14 Michael Elkin and Shaked Matar. Ultra-sparse near-additive emulators. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 235–246. ACM, 2021.

15 Michael Elkin and David Peleg. (1+epsilon, beta)-spanner constructions for general graphs. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 173–182. ACM, 2001.

16 Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *SIAM J. Discret. Math.*, 35(3):2129–2144, 2021. doi:10.1137/19M1306154.

17 Kevin Lu, Virginia Vassilevska Williams, Nicole Wein, and Zixuan Xu. Better lower bounds for shortcut sets and additive spanners via an improved alternation product. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 3311–3331. SIAM, 2022.

18 Seth Pettie. Low distortion spanners. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2007.

19 Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 802–809. ACM Press, 2006.

20 David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 389–398. IEEE Computer Society, 2006.