Multi-Level Weighted Additive Spanners

Reyan Ahmed
University of Arizona, Tucson, AZ, USA

Greg Bodwin
University of Michigan, Ann Arbor, MI, USA

Faryad Darabi Sahneh
University of Arizona, Tucson, AZ, USA

Keaton Hamm
University of Texas at Arlington, TX, USA

Stephen Kobourov
University of Arizona, Tucson, AZ, USA

Richard Spence
University of Arizona, Tucson, AZ, USA

Abstract

Given a graph $G = (V, E)$, a subgraph H is an *additive* $+\beta$ *spanner* if $\text{dist}_H(u, v) \leq \text{dist}_G(u, v) + \beta$ for all $u, v \in V$. A *pairwise spanner* is a spanner for which the above inequality is only required to hold for specific pairs $P \subseteq V \times V$ given on input; when the pairs have the structure $P = S \times S$ for some $S \subseteq V$, it is called a *subsetwise spanner*. Additive spanners in unweighted graphs have been studied extensively in the literature, but have only recently been generalized to weighted graphs.

In this paper, we consider a multi-level version of the subsetwise additive spanner in weighted graphs motivated by multi-level network design and visualization, where the vertices in S possess varying level, priority, or quality of service (QoS) requirements. The goal is to compute a nested sequence of spanners with the minimum total number of edges. We first generalize the $+2$ subsetwise spanner of [Pettie 2008, Cygan et al., 2013] to the weighted setting. We experimentally measure the performance of this and several existing algorithms by [Ahmed et al., 2020] for weighted additive spanners, both in terms of runtime and sparsity of the output spanner, when applied as a subroutine to multi-level problem.

We provide an experimental evaluation on graphs using several different random graph generators and show that these spanner algorithms typically achieve much better guarantees in terms of sparsity and additive error compared with the theoretical maximum. By analyzing our experimental results, we additionally developed a new technique of changing a certain initialization parameter which provides better spanners in practice at the expense of a small increase in running time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases multi-level, graph spanner, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SEA.2021.16

Related Version *Full Version*: `https://arxiv.org/abs/2102.05831`

Supplementary Material All algorithms, implementations, the ILP solver, experimental data and analysis are available on Github:
Software: `https://github.com/abureyanahmed/multi_level_weighted_additive_spanners`
archived at `swh:1:dir:95e49892297d01930d1de3c40e2bb2c39ccdd658`

Funding The research for this paper was partially supported by NSF grants CCF-1740858, CCF1712119, and DMS-1839274.

Acknowledgements The authors wish to thank the anonymous reviewers for their comments.

1 Introduction

Given an undirected graph, a *spanner* is a sparse subgraph with approximately the same distance metric as the original graph. Spanners are used as a primitive for many algorithmic tasks involving the analysis of distances or shortest paths in enormous input graphs; it is often advantageous to first replace the graph with a spanner, which can be analyzed much more quickly and stored in much smaller space, at the price of a small amount of error. See the recent survey [5] for more details on these applications.

Spanners were first studied with multiplicative error, where for an input graph G and an error ("stretch") parameter k, the spanner H must satisfy $\text{dist}_H(s,t) \leq k \cdot \text{dist}_G(s,t)$ for all vertices s,t, where $\text{dist}_G(s,t)$ denotes the distance in G between s and t. This setting was quickly resolved in a seminal paper by Althöfer, Das, Dobkin, Joseph, and Soares [9], where the authors proved that for all positive integers k, all n-vertex graphs have spanners on $O(n^{1+1/k})$ edges with stretch $2k-1$, and that this tradeoff is the best possible. Thus, as expected, one can trade off error for spanner sparsity, increasing the stretch k to pay more and more error for sparser and sparser spanners.

For very large graphs, additive error is arguably a much more appealing paradigm. Given $\beta > 0$, a $+\beta$ *spanner* of an n-vertex graph G is a subgraph H such that $\text{dist}_H(s,t) \leq \text{dist}_G(s,t) + \beta$ for all vertices s,t. Thus, for additive error the excess distance in H is independent of the graph size and of $\text{dist}_G(s,t)$, which can be large when n is large. Additive spanners were introduced by Liestman and Shermer [29], and followed by three landmark theoretical results on the sparsity of additive spanners in unweighted graphs: Aingworth, Chekuri, Indyk, and Motwani [8] showed that all graphs have $+2$ spanners on $O(n^{3/2})$ edges, Chechik [14, 18] showed that all graphs have $+4$ spanners on $O(n^{7/5})$ edges, and Baswana, Kavitha, Mehlhorn, and Pettie [12] showed that all graphs have $+6$ spanners on $O(n^{4/3})$ edges.

Despite the inherent appeal of additive error, spanners with multiplicative error remain much more commonly used in practice. There are two reasons for this.

1. First, while the multiplicative spanner of Althöfer et al [9] works without issue for weighted graphs, the previous additive spanner constructions hold only for unweighted graphs, whereas the metrics that arise in applications often require edge weights. Addressing this, recent work of the authors [3] and in two papers of Elkin, Gitlitz, and Neiman [23, 24] gave natural extensions of the classic additive spanner constructions to weighted graphs. For example, the $+2$ spanner bound becomes the following statement: for all n-vertex weighted graphs G, there is a subgraph H satisfying $\text{dist}_H(s,t) \leq \text{dist}_G(s,t) + 2W(s,t)$, where $W(s,t)$ denotes the maximum edge weight along an arbitrary $s \rightsquigarrow t$ shortest path in G. The $+4$ spanner generalizes similarly, and the $+6$ spanner does as well with the small exception that the error increases to $+(6+\varepsilon)W(s,t)$, for arbitrarily small $\varepsilon > 0$ which trades off with the implicit constant in the spanner size.
2. Second, $\text{poly}(n)$ factors in spanner size can be quite serious in large graphs, and so applications often require spanners of near-linear size, say $O(n^{1.01})$ edges for an n-vertex input graph. The worst case spanner sizes of $O(n^{4/3})$ or greater for additive spanner constructions are thus undesirable, and unfortunately, there is a theoretical barrier to improving them: Abboud and Bodwin [1] proved that one *cannot* generally trade off more additive error for sparser spanners, as one can in the multiplicative setting. Specifically, for any constant $c > 0$, there is no general construction of $+c$ spanners for n-vertex input graphs on $O(n^{4/3-0.001})$ edges. However, the lower bound construction is rather pathological, and it is not likely to arise in practice. It is known that for many practical

graph classes, e.g., those with good expansion, near-linear additive spanners always exist [12]. Thus, towards applications of additive error, it is currently an important open question whether modern additive spanner constructions on *practical* graphs of interest tend to exhibit performance closer to the worst-case bounds from [1], or bounds closer to the best ones available for the given input graphs We remark here that there are strong computational barriers to designing algorithms that achieve the sparsest possible $+c$ spanners directly, or which even closely approximate this quantity in general [19].

The goal of this work is to address the second point, by measuring the experimental performance of the state-of-the-art constructions for weighted additive spanners on graphs generated from various popular random models and measuring their performance. We consider both $+cW$ spanners (where $W = \max_{uv \in E} w(uv)$ is the maximum edge weight) and $+cW(\cdot,\cdot)$ spanners, whose additive error is $+cW(s,t)$ for each pair $s,t \in V$. We are interested both in runtime and in the ratio of output spanner size to the size of the sparsest possible spanner (which we obtain using an ILP, discussed in Section 4). We specifically consider generalizations of the three staple constructions for weighted additive spanners mentioned above, in which the spanner distance constraint only needs to be satisfied for given pairs of vertices.

In particular, the following extensions are considered. A *pairwise spanner* is a subgraph that must satisfy the spanner error inequality for a given set of vertex pairs P taken on input, and a *subsetwise spanner* is a pairwise spanner with the additional structure $P = S \times S$ for some vertex subset S. See e.g., [13, 15, 16, 21, 22, 27, 28, 32] for recent prior work on pairwise and subsetwise spanners, or the survey [5]. We then discuss a multi-level version of the subsetwise additive spanner where we have an edge-weighted graph $G = (V, E)$, a nested sequence of terminals $S_\ell \subseteq S_{\ell-1} \subseteq \cdots \subseteq S_1 \subseteq V$ and a real number $c \geq 0$ as input. We want to compute a nested sequence of subgraphs $H_\ell \subseteq H_{\ell-1} \subseteq \cdots \subseteq H_1$ such that H_i is a $+cW$ subsetwise spanner of G over S_i. The objective is to minimize the total number of edges in all subgraphs. Similar generalizations have been studied for the Steiner tree problem under various names including Multi-level Network Design [10], Quality of Service Multicast Tree (QoSMT) [17, 26], Priority Steiner Tree [20], Multi-Tier Tree [30], and Multi-level Steiner Tree [2, 7]. However, multi-level or QoS generalizations of spanner problems appear to have been much less studied in literature. These types of problems have applications in multi-level graph visualization and other network design problems where vertices may require different level or QoS requirements. Section 2 generalizes the clustering-based +2 subsetwise spanner [22] to weighted graphs, and Section 3 generalizes to the multi-level setting. Section 5 contains an experimental comparison between several different spanner algorithms given here and in [3] to infer that many of these spanner constructions typically achieve better error or sparsity bounds than the theoretical worst-case bounds. This comparison also suggests that spanners constructed by a certain light initialization technique given in [3] (Section 5.2.9) tend to outperform spanners constructed by clustering and path buying in terms of the sparsity; further, by varying an initialization parameter and selecting the sparsest spanner, we can compute much sparser additive spanners in random graphs at the expense of a logarithmic factor in running time.

2 Subsetwise spanners

All unweighted graphs have polynomially constructible +2 subsetwise spanners over $S \subseteq V$ on $O(n\sqrt{|S|})$ edges [22, 32]. For weighted graphs, Ahmed et al. [3] recently give a $+4W$ subsetwise spanner construction, also using $O(n\sqrt{|S|})$ edges. In this section we show how

to generalize the +2 subsetwise construction [22, 32] to the weighted setting by giving a construction which produces a subsetwise $+2W$ spanner of a weighted graph (with integer edge weights in $[1, W]$) on $O(nW\sqrt{|S|})$ edges.

A *clustering* $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ is a set of disjoint subsets of vertices. Initially, every vertex is unclustered. The subsetwise $+2W$ construction has two steps: the clustering phase and the path buying phase. The clustering phase is exactly the same as that of [22, 32] in which we construct a cluster subgraph $G_{\mathcal{C}}$ as follows: set $\beta = \log_n \sqrt{|S|W}$, and while there is a vertex v with at least $\lceil n^\beta \rceil$ unclustered neighbors, we add a cluster C to $\mathcal{C}$ containing exactly $\lceil n^\beta \rceil$ unclustered neighbors of v (note that $v \notin C$). We add to $G_{\mathcal{C}}$ all edges vx ($x \in C$) and xy ($x, y \in C$). When there are no more vertices with at least $\lceil n^\beta \rceil$ unclustered neighbors, we add all the unclustered vertices and their incident edges to $G_{\mathcal{C}}$.

In the second (path-buying) phase, we start with a clustering $\mathcal{C}$ and a cluster subgraph $G_0 := G_{\mathcal{C}}$. There are $z := \binom{|S|}{2}$ unordered pairs of vertices in S; let π_1, π_2, ..., π_z denote the shortest paths between these vertex pairs where $\pi_i = \pi(u_i, v_i)$ has endpoints $\{u_i, v_i\}$. As in [22], we iterate from $i = 1$ to $i = z$ and determine whether to add path π_i to the spanner. Define the cost and value of a path π_i as follows:

$$\begin{aligned}
\text{cost}(\pi_i) &:= \#\text{ edges of } \pi_i \text{ which are absent in } G_{i-1} \\
\text{value}(\pi_i) &:= \#\text{ pairs } (x, C) \text{ where } x \in \{u_i, v_i\}, C \in \mathcal{C}, \\
&\qquad C \text{ contains at least one vertex in } \pi_i, \\
&\qquad \text{and } \text{dist}_{\pi_i}(x, C) < \text{dist}_{G_{i-1}}(x, C)
\end{aligned}$$

If $\text{cost}(\pi_i) \leq (2W + 1)\text{value}(\pi_i)$, then we add ("buy") π_i to the spanner by letting $G_i = G_{i-1} \cup \pi_i$. Otherwise, we do not add π_i, and let $G_i = G_{i-1}$. The final spanner returned is $H = G_z$.

The unweighted +2 subsetwise spanner [22] and corresponding cluster subgraph $G_{\mathcal{C}}$ rely on the following properties:

- *Missing-edge property*: if an edge $uv \in E$ is absent in $G_{\mathcal{C}}$, then u and v belong to two different clusters
- *Cluster-diameter property*: the distance in $G_{\mathcal{C}}$ between two vertices in the same cluster is at most 2 ($2W$ for weighted graphs)

Using these properties, a lemma in [22] states that if a shortest u-v path $\pi(u, v)$ contains $t \geq 1$ edges which are absent in $G_{\mathcal{C}}$, then there are at least $t/2$ clusters in $\mathcal{C}$ which contain at least one vertex on $\pi(u, v)$. This lemma does not quite hold in weighted graphs since a shortest path can pass through the same cluster many times; we instead prove the following generalization:

▶ **Lemma 1.** *Let G be a weighted graph with edge weights in $[1, W]$ and let $\pi(u, v)$ be a shortest path which contains t edges which are absent from $G_{\mathcal{C}}$. Then there are at least $t/(W + 1)$ clusters of $\mathcal{C}$ which contain at least one vertex on $\pi(u, v)$.*

Proof. Consider pairs (x, e) where e is an edge of $\pi(u, v)$ absent in $G_{\mathcal{C}}$ and x is one of the endpoints of e. There are t missing edges, so there are $2t$ such pairs. A cluster $C \in \mathcal{C}$ is said to *own* the pair (x, e) if $x \in C$. By the missing-edge property, every missing edge is incident to two different clusters, so each pair (x, e) is owned by some cluster.

Consider some cluster $C \in \mathcal{C}$ such that $\pi(u, v)$ passes through some vertex in C. By the cluster-diameter property and using the fact that all edges have weight at least 1, $\pi(u, v)$ cannot pass through more than $2W$ vertices in C. Using this we can show that C owns at most $2W + 2$ pairs (x, e). Since there are exactly $2t$ vertex-edge pairs and each cluster passing through some point in $\pi(u, v)$ owns at most $2W + 2$ pairs, we conclude that there are at least $\frac{2t}{2W+2} = \frac{t}{W+1}$ clusters which contain at least one vertex in $\pi(u, v)$. ◀

▶ **Lemma 2.** *For any $u_i, v_i \in S$, we have $\text{dist}_H(u_i, v_i) \leq \text{dist}_G(u_i, v_i) + 2W$.*

▶ **Lemma 3.** *For $\beta = \log_n \sqrt{|S|W}$, the $+2W$ subsetwise spanner H has $O(nW\sqrt{|S|})$ edges.*

The proofs of these lemmas are largely the same as in [22] except we incur an additional W in the size bound due to the cost vs. value when considering when to buy the path π_i to the spanner.

▶ **Corollary 4.** *Let G be a weighted graph with integer edge weights in $[1, W]$. Then G has a $+6W$ pairwise spanner on $O(Wn|P|^{1/4})$ edges.*

This follows from applying the $+8W$ construction of Ahmed et al. [3] (Appendix A, Algorithm 3), except we use the above $+2W$ subsetwise spanner instead of the $+4W$ subsetwise spanner construction given in [3] as a subroutine.

3 Multi-level spanners

Here we study a multi-level variant of graph spanners. We first define the problem:

▶ **Definition 5** (Multi-level weighted additive spanner)**.** *Given a weighted graph $G(V, E)$ with maximum weight W, a nested sequence of subsets of vertices $S_\ell \subseteq S_{\ell-1} \subseteq \ldots \subseteq S_1 \subseteq V$, and $c \geq 0$, a multi-level $+cW$ spanner is a nested sequence of subgraphs $H_\ell \subseteq H_{\ell-1} \subseteq \ldots \subseteq H_1 \subseteq G$, where H_i is a subsetwise $+cW$ spanner over S_i.*

Observe that Definition 5 generalizes the subsetwise spanner, which is a special case where $\ell = 1$. We define the *sparsity* of a multi-level spanner by $\text{sparsity}(\{H_i\}_{i=1}^{\ell}) := \sum_{i=1}^{\ell} |E(H_i)|$, where lower sparsity is more desirable. In the following sections, we also measure the quality of a multi-level spanner in terms of the ratio of its sparsity to the minimum possible sparsity over all candidate multi-level spanners (denoted by OPT).

The multi-level spanner can equivalently be phrased in terms of priorities and rates: each vertex $v \in S_1$ has a priority $P(v)$ between 1 and ℓ (namely, $P(v) = \max\{i : v \in S_i\}$), and we wish to compute a single subgraph containing edges of different rates such that for all $u, v \in S_1$, there is a $+cW$ spanner path consisting of edges of rate at least $\min\{P(u), P(v)\}$. With this, we will typically refer to the *priority* of v to denote the highest i such that $v \in S_i$, or 0 if $v \notin S_1$. In this section, we show that the multi-level version is not significantly harder than the ordinary "single-level" version: a subroutine which can compute an additive spanner can be used to compute a multi-level spanner whose sparsity is comparably good.

We first describe a simple rounding-up approach based on an algorithm by Charikar et al. [17] for the QoSMT problem, a similar generalization of the Steiner tree problem. For this approach, assume we have a subroutine which computes an exact or approximate single-level subsetwise spanner. Given $v \in S_1$, let $P(v) \in [1, \ell]$ denote the priority of v. The rounding-up approach is as follows: for each v, round $P(v)$ up to the nearest power of 2. This effectively constructs a "rounded-up" instance where all vertices in S_1 have priority 1, 2, 4, ..., or $2^{\lceil \log_2 \ell \rceil}$. The sparsity of the optimum solution in the rounded-up instance is at most $2\,\text{OPT}$; given the optimum solution to the original instance with sparsity OPT, a feasible solution to the rounded-up instance with sparsity at most $2\,\text{OPT}$ can be obtained by rounding up the rate of each edge to the nearest power of 2.

For each $i \in \{1, 2, 4, \ldots, 2^{\lceil \log_2 \ell \rceil}\}$, use the subroutine to compute a level-i subsetwise spanner over all vertices whose rounded-up priority is at least i. The final multi-level additive spanner is obtained by taking the union of these computed spanners, by keeping an edge at the highest level it appears in. This requires $O(\log \ell)$ calls to the single-level subroutine.

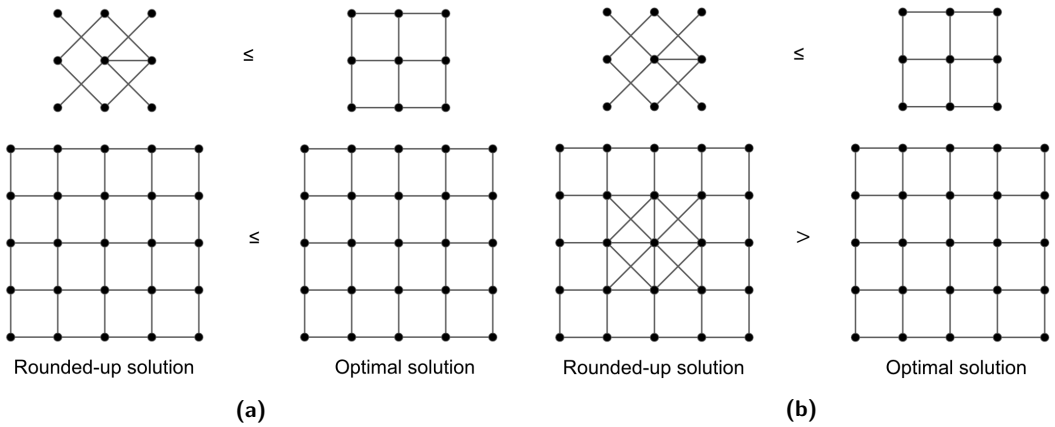

Figure 1 (a) The rounding-up approach computes an optimal spanner at each level (assuming an exact subroutine), so the sizes of the spanners on each level are at most that of the optimal solution ($9 + 40$ edges vs. $12 + 40$). (b) However, when an edge is present in a top-level solution, it must be present in lower-level solutions. The rounding-up approach takes the union of the spanners in the bottom level; in this case, the sparsity of the rounded-up solution ($9 + 48$ vs. $12 + 40$) is greater than that of the optimum.

▶ **Theorem 6.** *Assuming an exact subsetwise spanner subroutine, the solution computed by the rounding-up approach has sparsity at most* $4 \cdot \mathrm{OPT}$.

This is proved using the same ideas as the 4ρ-approximation for QoSMT [17]. As mentioned earlier, in practice we use an approximation algorithm to compute the subsetwise spanner instead of computing the minimum spanner.

▶ **Theorem 7.** *There exists a* $\tilde{O}(n/\sqrt{|S_1|})$*-approximation algorithm to compute multi-level* $+2W$ *spanners when* $W = O(\log n)$.

This follows from using the $+2W$ subsetwise construction in Section 2. The approximation ratio of this subsetwise spanner algorithm is $O(nW/\sqrt{|S|})$ as the construction produces a spanner of size $O(nW\sqrt{|S|})$, while the sparsest additive spanner trivially has at least $|S| - 1 = \Omega(|S|)$ edges.

We also show that, under certain conditions, if we have a subroutine which computes a subsetwise spanner of G, S of size $O(n^a|S|^b)$ where a and b are absolute constants, a very naïve algorithm can be used to obtain a multi-level spanner also with sparsity $O(n^a|S_1|^b)$.

▶ **Theorem 8.** *Suppose there is an absolute constant* $0 < \alpha < 1$ *such that* $|S_i| \leq \alpha|S_{i-1}|$ *for all* $i \in \{1, \ldots, \ell\}$. *Then we can compute a multi-level spanner with sparsity* $O(n^a|S_1|^b)$.

Proof. Consider the following simple construction: for each $i \in \{1, 2, 3, \ldots, \ell\}$, compute a level-$i$ subsetwise spanner of size $O(n^a|S_i|^b)$. Consider the union of these spanners, by keeping each edge at the highest level it appears. The sparsity of the returned multi-level spanner is at most

$$\begin{aligned}\text{sparsity}(\{H_i\}) &= O(n^a|S_1|^b + 2n^a|S_2|^b + 3n^a|S_3|^b + \ldots + \ell n^a|S_\ell|^b) \\ &\leq O(n^a|S_1|^b(1 + 2\alpha^b + 3\alpha^{2b} + \ldots + \ell\alpha^{(\ell-1)b})) \\ &= O(n^a|S_1|^b)\end{aligned}$$

where we used the arithmetico-geometric series $1 + 2(\alpha^b) + 3(\alpha^b)^2 + \ldots = \frac{1}{(1-\alpha^b)^2}$ which is constant for fixed α, b. Note that $0 < \alpha < 1$ and $b > 0$, which implies $0 < \alpha^b < 1$. ◀

The assumption that $|S_i| \leq \alpha |S_{i-1}|$ for some constant α is fairly natural, as many realistic networks tend to have significantly fewer hubs than non-hubs.

▶ **Corollary 9.** *Under the assumption $|S_i| \leq \alpha |S_{i-1}|$ for all $i \in \{2, \ldots, \ell\}$, there exists a poly-time algorithm which computes a multi-level $+2$ spanner of sparsity $O(n\sqrt{|S_1|})$.*

Proof. This follows by using the $+2$ construction by Cygan et al. [22] on $O(n\sqrt{|S|})$ edges as the subroutine. ◀

4 Integer programming formulation

To compute a minimum size $+cW$ spanner over vertex pairs P, we utilize a slight modification of the ILP in [5, Section 9], wherein we choose the specific distortion function $f(t) = t + cW$ and minimize the sparsity rather than total weight of the spanner. For completeness, we present the full ILP for computing a single-level additive subsetwise spanner below along with a brief description of the multi-level extension. Here E' represents the bidirected edge set, obtained by adding directed edges (u, v) and (v, u) for each edge $uv \in E$. The binary variable $x^{uv}_{(i,j)}$ is 1 if edge (i, j) is included on the selected u-v path and 0 otherwise, and $w(e)$ is the weight of edge e.

$$\text{Minimize} \sum_{e \in E} x_e \text{ subject to} \tag{1}$$

$$\sum_{(i,j) \in E'} x^{uv}_{(i,j)} w(e) \leq \text{dist}_G(u, v) + cW \qquad \forall (u, v) \in P;\ e = ij \tag{2}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} - \sum_{(j,i) \in In(i)} x^{uv}_{(j,i)} = \begin{cases} 1 & i = u \\ -1 & i = v \\ 0 & \text{else} \end{cases} \qquad \forall (u, v) \in P;\ \forall i \in V \tag{3}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} \leq 1 \qquad \forall (u, v) \in P;\ \forall i \in V \tag{4}$$

$$x^{uv}_{(i,j)} + x^{uv}_{(j,i)} \leq x_e \qquad \forall (u, v) \in P;\ \forall e = ij \in E \tag{5}$$

$$x_e, x^{uv}_{(i,j)} \in \{0, 1\} \tag{6}$$

Inequalities (3)–(4) enforce that for each $(u, v) \in P$, the selected edges corresponding to u, v form a path; inequality (2) enforces that the length of this path is at most $\text{dist}_G(u, v) + cW$ (note that W may be replaced with $W(u, v)$). Inequality (5) ensures that if $x^{uv}_{(i,j)} = 1$ or $x^{uv}_{(i,j)} = 1$, then edge ij is taken.

To generalize the ILP formulation to the multi-level problem, we take a similar set of variables for every level. The rest of the constraints are similar, except we define $x^k_e = 1$ if edge e is present on level k and the variables $x^{uv}_{(i,j)}$ are also indexed by level. We add the constraint $x^k_e \leq x^{k-1}_e$ for all $k \in \{2, \ldots, \ell\}$ which enforces that if edge e is present on level k, it is also present on all lower levels. Finally, the objective is to minimize the sparsity $\sum_{k=1}^{\ell} \sum_{e \in E} x^k_e$.

5 Experiments

In this section, we provide experimental results involving the rounding-up framework described in Section 3. This framework needs a single level subroutine; we use the $+2W$ subsetwise construction in Section 2 and the three pairwise $+2W(\cdot, \cdot)$, $+4W(\cdot, \cdot)$, $+6W$ constructions

provided in [3][1] (see Appendix A). We generate multi-level instances and solve the instances using the ILP and the four approximation algorithms. We consider natural questions about how the number of levels ℓ, number of vertices n, and decay rate of terminals with respect to levels affect the running times and (experimental) approximation ratios, defined as the sparsity of the returned multi-level spanner divided by OPT.

We used CPLEX 12.6.2 as an ILP solver in a high-performance computer for all experiments (Lenovo NeXtScale nx360 M5 system with 400 nodes). Each node has 192 GB of memory. We have used Python for implementing the algorithms and spanner constructions. Since we have run the experiment on several thousand instances, we run the solver for four hours per instance.

5.1 Experiment parameters

We run experiments first to test experimental approximation ratio vs. the parameters, and then to test running time vs. parameters. Each set of experiments has several parameters: the graph generator, the number of levels ℓ, the number of vertices n, and how the size of the terminal sets S_i (vertices requiring level or priority at least i) decrease as i decreases.

In what follows, we use the Erdős–Rényi (ER) [25], Watts–Strogatz (WS) [33], Barabási–Albert (BA) [11], and random geometric (GE) [31] models. Let p be the edge selection probability. If we set $p = (1+\varepsilon)\frac{\ln n}{n}$, then the generated Erdős–Rényi graph is connected with high probability for $\varepsilon > 0$ [25]. For our experiments we use $\varepsilon = 1$. In the Watts-Strogatz model, we initially create a ring lattice of constant degree K. For our experiments we use $K = 6$ and $p = 0.2$. In the Barabási–Albert model, a new vertex is connected to m existing vertices. For our experiments we use $m = 5$. In the random geometric graph model, two vertices are connected to each other if their Euclidean distance is not larger than a threshold r_c. For $r_c = \sqrt{\frac{(1+\epsilon)\ln n}{\pi n}}$ with $\epsilon > 0$, the synthesized graph is connected with a high probability [31]. We generate a set of small graphs ($10 \leq n \leq 40$) and a set of large graphs ($50 \leq n \leq 500$). We only compute the exact solutions for the small graphs since the ILP has an exponential running time. In this paper, we provide the results of Erdős–Rényi graphs since it is the most popular model. However, the radius[2] of Erdős–Rényi graphs is relatively small; in our dataset, the range of the radius is 2-4. Hence, we also provide the results of random geometric graphs which have larger radius (4-12). The remaining results and the radius distribution of different generators are available at the supplement Github link. We consider number of levels $\ell \in \{1, 2, 3\}$ for small graphs, $\ell \in \{1, \ldots, 10\}$ for large graphs, and adopt two methods for selecting terminal sets: *linear* and *exponential*. A terminal set S_1 with lowest priority of size $n(1 - \frac{1}{\ell+1})$ in the linear case and $\frac{n}{2}$ in the exponential case is chosen uniformly at random. For each subsequent level, $\frac{1}{\ell+1}$ vertices are deleted at random in the linear case, whereas half the remaining vertices are deleted in the exponential case. Levels/priorities and terminal sets are related via $S_i = \{v \in S_1 : P(v) \geq i\}$. We choose edge weights $w(e)$ independently uniformly at random from $\{1, 2, 3 \ldots, 10\}$.

An experimental instance of the multi-level problem here is thus characterized by four parameters: graph generator, number of vertices n, number of levels ℓ, and terminal selection method TSM $\in \{$Linear,Exponential$\}$. As there is randomness involved, we generated five instances for every choice of parameters (e.g., ER, $n = 30$, $\ell = 2$, Linear). For each

[1] Note that, one can show that the $+2W$, $+4W$, $+8W$ spanners in [3] are actually $+2W(.,.), +4W(.,.)$ and $+6W$ spanners respectively by using a tighter analysis [4].

[2] The minimum over all $v \in V$ of $\max_{w \in V} d_G(v, w)$ where $d_G(v, w)$ is the graph distance (by number of edges, not total weight) between v and w

instance of the small graphs, we compute the approximate solution using either the $+2W$, $+2W(\cdot,\cdot)$, $+4W$, or $+6W$ spanner subroutine, and the exact solution using the ILP described in Section 4. We compute the experimental approximation ratio ("Ratio") by dividing the sparsity of the approximate solution by the sparsity of the optimum solution (OPT). For large graphs, we only compute the approximate solution.

5.2 Results

We consider different spanner constructions as the single level subroutine in the rounding-up approach described in Section 3. We first consider the $+2W$ subsetwise construction (Section 2).

5.2.1 Multi-level $+2W$ spanner

We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method in Figure 2. The average experimental ratio increases as n increases approximately linearly in n, which is expected since the theoretical approximation ratio of $\tilde{O}(n/\sqrt{|S_1|})$ is proportional to n. The average and minimum experimental ratio does not change significantly as the number of levels increases; however, the maximum ratio increases. The experimental ratio of the linear terminal selection method is slightly better compared to that of the exponential method.

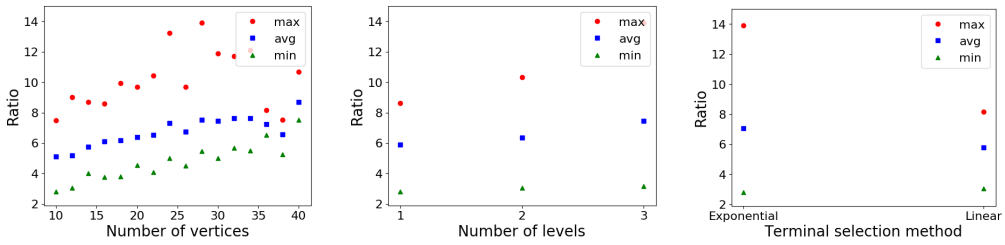

Figure 2 Performance of the algorithm that uses $+2W$ subsetwise spanner as the single level subroutine on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.2 Multi-level $+2W(\cdot,\cdot)$ spanner

We now consider the $+2W(\cdot,\cdot)$ pairwise construction [3] (Algorithm 1, Appendix A) as a subroutine, with $P = S \times S$. We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method in Figure 3. The average experimental ratio increases as n increases. This is again expected since the theoretical approximation ratio is proportional to n. The average and minimum experimental ratio do not change that much as the number of levels increases, however, the maximum ratio increases. The experimental ratio of the linear terminal selection method is also slightly better compared to that of the exponential method.

5.2.3 Comparison between global and local spanners

One major difference between the subsetwise and pairwise construction is the subsetwise construction considers the (global) maximum edge weight W of the graph in the error. On the other hand, the $+cW(\cdot,\cdot)$ spanners consider the (local) maximum edge weight in a shortest path for each pair of vertices s,t. We provide a comparison between the global and local settings.

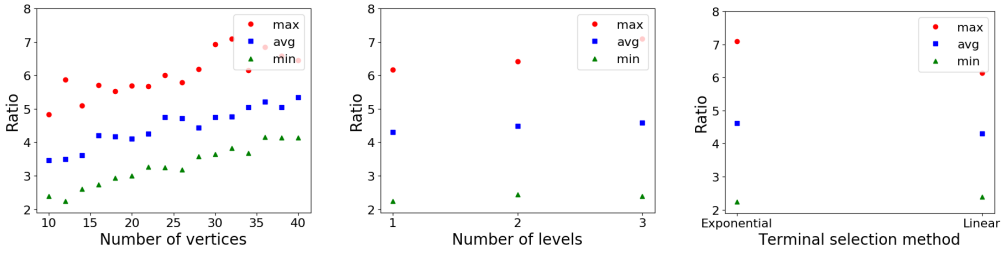

Figure 3 Performance of the algorithm that uses $+2W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

We describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 4. The average experimental ratio increases as n increases for both global and local settings. However, the ratio of the local setting is smaller compared to that of the global setting. One reason for this difference is the solution to the global exact algorithm is relatively smaller since the global setting considers larger errors. The ratio of the global setting increases as the number of levels increases and for the exponential terminal selection method. For the local setting, the ratio does not change significantly.

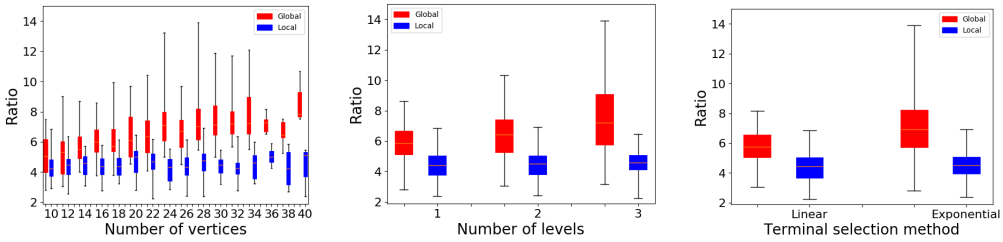

Figure 4 Performance of the global (subsetwise $+2W$) and local (pairwise $+2W(\cdot,\cdot)$) construction-based algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.4 Multi-level $+4W(\cdot,\cdot)$ spanner

We now consider the $+4W(\cdot,\cdot)$ pairwise construction [3] (Algorithm 2, Appendix A) as a single level subroutine. We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method in Figure 5. The average experimental ratio increases as n increases. This is expected since the theoretical approximation ratio is proportional to n. The average experimental ratio does not change significantly as the number of levels increases; however, the maximum ratio increases. The experimental ratio of the linear terminal selection method is also slightly better compared to that of the exponential method.

5.2.5 Comparison between $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ spanners

We now provide a comparison between the pairwise $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ construction-based approximation algorithms. We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 6. The average experimental ratio increases as n increases for both $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ settings. The $+4W(\cdot,\cdot)$ construction-based algorithm slightly outperforms the $+2W(\cdot,\cdot)$ algorithm for $\ell = 3$ and exponential selection method.

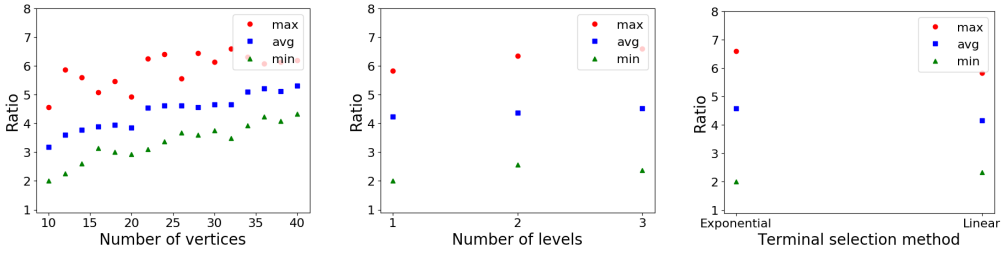

Figure 5 Performance of the algorithm that uses $+4W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

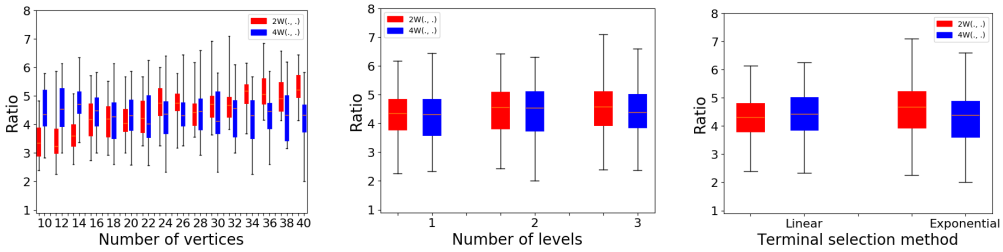

Figure 6 Performance of the pairwise $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ construction-based algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.6 Multi-level $+6W$ spanner

We now consider the $+6W$ pairwise construction [3] (Algorithm 3, Appendix A) as a single level subroutine. We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method in Figure 7. The average experimental ratio increases as n increases. This is expected since the theoretical approximation ratio is proportional to n. The average experimental ratio does not change significantly as the number of levels increases; however, the maximum ratio increases. The maximum and average experimental ratios of the linear terminal selection method are slightly better compared to that of the exponential method.

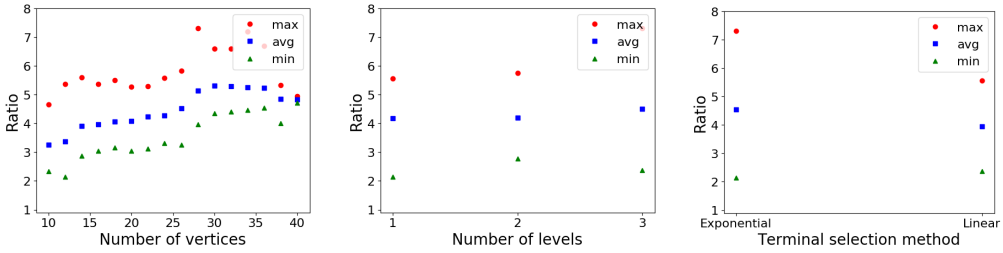

Figure 7 Performance of the algorithm that uses $+6W$ pairwise spanner as the single level subroutine on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.7 Comparison between $+2W$ and $+6W$ spanners

We now provide a comparison between the pairwise $+2W$ and $+6W$ construction-based approximation algorithms. We first describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 8. The average experimental

ratio increases as n increases for both $+2W$ and $+6W$ settings. As the number of vertices increases, the ratio of the $+6W$ construction-based algorithm gets smaller. This is expected since a larger error makes the problem easier to solve. Similarly, as ℓ increases, the $+6W$ construction-based algorithm outperforms the $+2W$ algorithm. The average experimental ratio of the $+6W$ construction based algorithm is smaller both in the linear and exponential terminal selection methods.

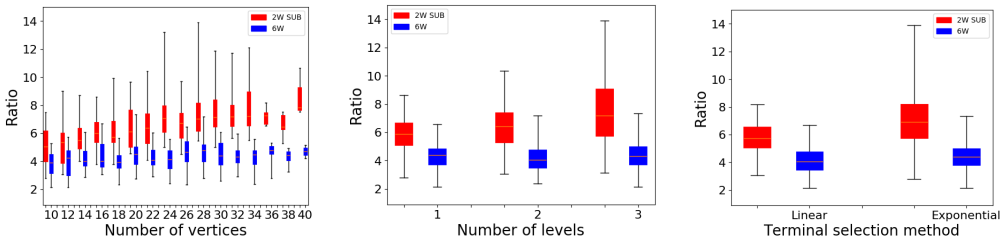

Figure 8 Performance of the pairwise $+2W$ and $+6W$ construction-based algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.8 Experiment on large graphs

We generate some large instances on up to 500 vertices and run different multi-level spanner algorithms on them. We use $n = \{50, 100, 150, \ldots, 500\}$ and $\ell = \{1, 2, 3, \ldots, 10\}$. We describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 9. We are comparing four multi-level algorithms, namely those using the $+2W$ subsetwise and $+2W(\cdot,\cdot)$, $+4W(\cdot,\cdot)$, $+6W$ pairwise constructions [3] as subroutines with $P = S \times S$. Since computing the optimal solution exactly via ILP is computationally expensive on large instances, we report the ratio in terms of relative sparsity, defined as the sparsity of the multi-level spanner returned by one algorithm divided by the minimum sparsity over the spanners returned by all four. The ratio of the $+6W$ construction based algorithm is lowest and the $+2W$ construction based algorithm is highest. This is expected since a higher additive error generally reduces the number of edges needed. Overall the ratio decreases as n increases. This is because the significance of small additive error reduces as the graph size and distances get larger. The relative ratio for the $+2W$ construction increases as ℓ increases, and for the exponential terminal selection method.

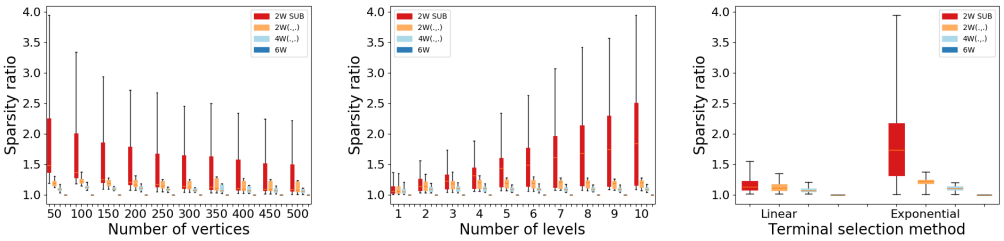

Figure 9 Performance of different approximation algorithms on large Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.2.9 Impact of the initialization parameters

It is worth mentioning that the $+2$ subsetwise spanner [22] and $+2W$ subsetwise spanner (Section 2) begin with a clustering phase, while the algorithms described in Appendix A begin with a d-light initialization. In d-light initialization, we add the d lightest edges incident to each vertex, where $d \geq 1$ is a parameter specific to the algorithm; these edges tend to be on shortest paths. In practice, there may be relatively few edges which appear on shortest paths and some of these edges might be redundant. Hence, we compute $+2W(\cdot,\cdot)$ spanners with different values of d. We describe the experimental results on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 10. We have computed the ratio as described in Section 5.2.8. From the figures, we see that as we reduce the value of d exponentially, the ratio decreases: in particular, the optimal choice of parameter d in practice might be significantly smaller than the optimal value of d in theory. Generally, it could make sense in practical implementations of spanner algorithms to try all values $\{d, d/2, d/4, d/8, \dots\}$, computing $\approx \log_2 d$ different spanners, and then use only the sparsest one. This costs only a $O(\log d)$ factor in the running time of the algorithm, which is typically reasonable.

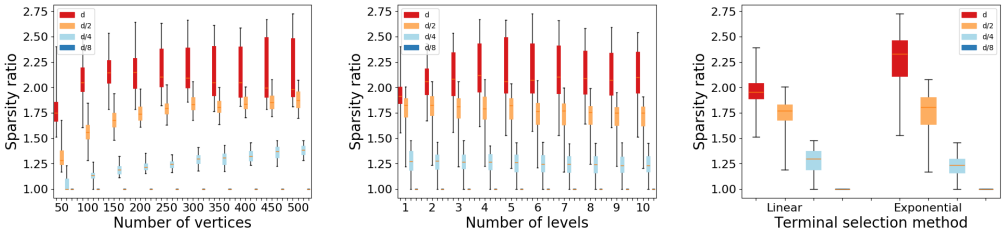

Figure 10 Impact of different values of d on large Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.3 Running time

We now provide the running times of the different algorithms. We show the running time of the ILP on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method in Figure 11. The running time of the ILP increases exponentially as n increases, as expected. The execution time of a single level instance with 45 vertices is more than 64 hours using a 28 core processor. Hence, we kept the number of vertices less than or equal to 40 for our small graphs. The experimental running time should increase as ℓ increases, but we do not see that pattern in these plots because some of the instances were not able to finish in four hours.

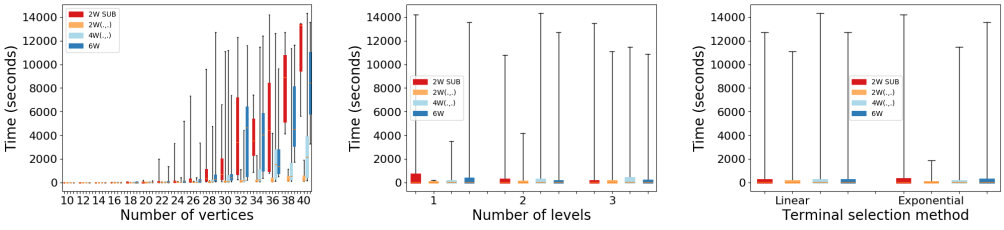

Figure 11 Running time of all exact algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

We provide the experimental running time of the approximation algorithm on Erdős–Rényi graphs in Figure 12. The running time of the $+2W$ construction-based algorithm is the largest. Overall, the running time increases as n increases. There is no straightforward relation between the running time and ℓ. Although the number of calls to the single level subroutine increases as ℓ increases, it also depends on the size of the subset in a single level. Hence, if the subset sizes are larger, then it may take longer for small ℓ. The running time of the linear method is larger.

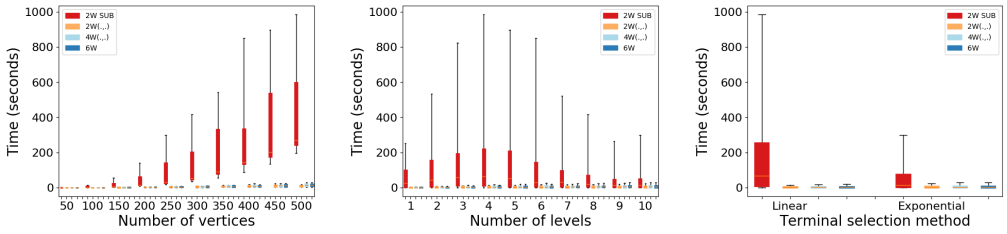

Figure 12 Running time of all approximation algorithms on large Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

The running times appear reasonable in other settings too; see the supplemental Github repository and the arXiv version [6] of this paper for details and experimental results.

5.4 Experimental additive error

Different spanner constructions provide theoretical guarantees on the maximum amount of additive error. For example, a $+2W$ subsetwise spanner over G, S ensures that any pair of vertices in S does not have an error of more than $+2W$. Similarly, the $+2W(\cdot,\cdot)$, $+4W(\cdot,\cdot)$, and $+6W$ pairwise construction ensures that the error in the shortest path distance in the spanner is no more than $+2W(\cdot,\cdot)$, $+4W(\cdot,\cdot)$, and $+6W$ respectively. These are theoretical upper bounds that directly appear from the construction. However, in our experiment, we have found that the theoretical upper bound is never achieved, and most vertex pairs contain a less additive error. We define the *error ratio* of an additive spanner H to be the sum of additive errors in H (over all vertex pairs) divided by the maximum possible sum of errors. For example, if H is a $+2W(\cdot,\cdot)$ spanner over vertex pairs $P \subseteq V \times V$, then

$$\text{error ratio} := \frac{\sum_{(u,v)\in P}(d_H(u,v) - d_G(u,v))}{\sum_{(u,v)\in P} 2W(u,v)}.$$

For a multi-level spanner, we define the error ratio similarly, except we additionally sum the numerator (and denominator) over all subgraphs H_i from $i = 1$ to ℓ. We consider the $+2W(\cdot,\cdot)$ pairwise construction [3] (Algorithm 1) as a single level subroutine and compute the error ratios using Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 13. Figure 13 suggests that the average error ratio is typically less than 0.05; in other words, the $+2W(\cdot,\cdot)$ spanner algorithm outputs a spanner whose average additive error is around 5% of the maximum allowable error. We provide the comparison among all algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 14.

5.5 Relative sparsity

In most of the previous figures, we provide the sparsity (number of edges) of the spanner, relative to the optimum spanner. Here we provide a comparison of the spanner sparsities of the four spanner algorithms ($+2W$ subsetwise, $+2W(\cdot,\cdot)$, $+4W(\cdot,\cdot)$, and $+6W$) relative to

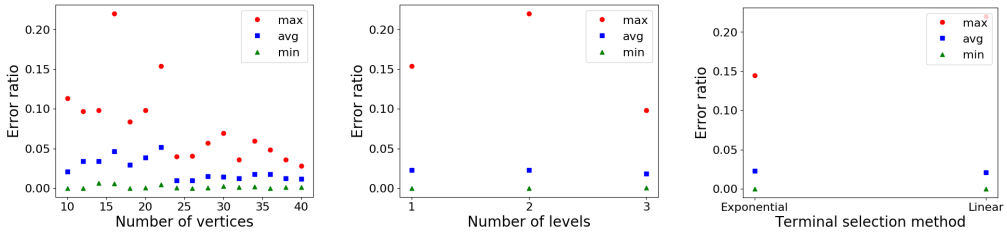

Figure 13 Average error ratios of the spanners computed using the algorithm that uses $+2W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

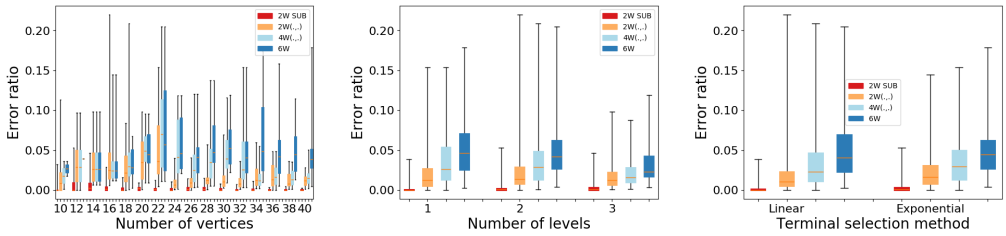

Figure 14 Average error ratios of the spanners computed using all algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

each other in Figure 15. According to Figure 15, the $+6W$ construction-based approximation algorithm uses the fewest number of edges, and the $+2W$ subsetwise spanner typically outputs a spanner with 50% to 75% more edges than the $+6W$ spanner; this is expected as more additive error generally leads to sparser spanners in practice.

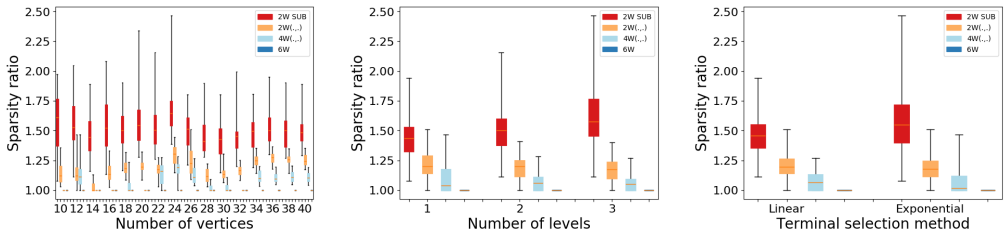

Figure 15 Sparsity ratio of the spanners computed using all algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

5.6 Amount of reduction

One of the major goals of graph spanners is to sparsify the input graph without losing significant information on distances in the original graph. A straightforward way to compute the amount of reduction is to take the ratio of the number of edges removed from the input graph in the spanner to the number of edges in the input graph. Since we are computing multi-level spanners, we define the reduction ratio as the number of edges removed from the input divided by $|E|\ell$ (in other words, $(|E|\ell - \text{sparsity}(\{H_i\}))/(|E|\ell)$). We describe these reduction ratios on Erdős–Rényi graphs w.r.t. n, ℓ, and the terminal selection method in Figure 16. The global $+2W$ construction-based approximation algorithm provides the smallest reduction ratio which is also consistent with the idea that sparser spanners generally

take on more additive error. Remember that the $+2W$ construction uses clustering and other construction uses d-initialization. This result again indicates that the clustering-based approach performs worse compared to the initialization-based approach.

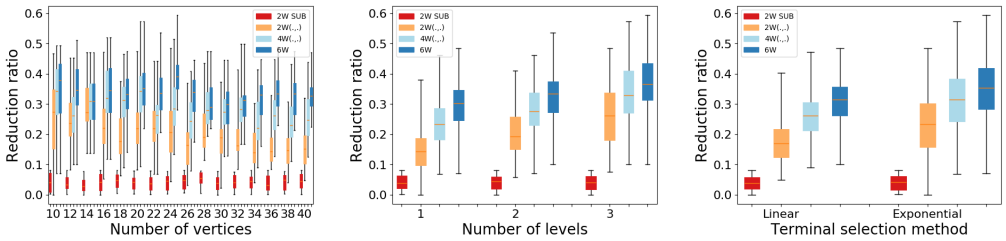

Figure 16 Reduction ratio of the spanners computed using all algorithms on Erdős–Rényi graphs w.r.t. n, ℓ, and terminal selection method.

6 Conclusion

We have provided a framework where we can use different spanner subroutines to compute multi-level spanners of varying additive error. We additionally introduced a generalization of the $+2$ subsetwise spanner [22] to integer edge weights, and illustrate that this can reduce the $+8W$ error in [3] to $+6W$. A natural question is to provide an approximation algorithm that can handle different additive error for different levels. We also provided an ILP to compute the optimum spanner; computing this optimally is very slow, so natural directions include using techniques such as graph reduction to sparsify the input graph before computing a spanner. The experimental results in Section 5 suggest that the $+2W$ clustering-based approach is slower and returns worse spanners than the initialization based approaches. We provided a method of changing the initialization parameter d which reduces the sparsity in practice.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *Journal of the ACM (JACM)*, 64(4):1–20, 2017.

2 Abu Reyan Ahmed, Patrizio Angelini, Faryad Darabi Sahneh, Alon Efrat, David Glickenstein, Martin Gronemann, Niklas Heinsohn, Stephen Kobourov, Richard Spence, Joseph Watkins, and Alexander Wolff. Multi-level Steiner trees. In *17th International Symposium on Experimental Algorithms, (SEA)*, pages 15:1–15:14, 2018. doi:10.4230/LIPIcs.SEA.2018.15.

3 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen Kobourov, and Richard Spence. Weighted additive spanners. In Isolde Adler and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science*, pages 401–413. Springer, 2020.

4 Reyan Ahmed, Greg Bodwin, Keaton Hamm, Stephen Kobourov, and Richard Spence. Weighted sparse and lightweight spanners with local additive error. *arXiv preprint arXiv:2103.09731*, 2021.

5 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020.

6 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Stephen Kobourov, and Richard Spence. Multi-level weighted additive spanners. *arXiv preprint arXiv:2102.05831*, 2021.

7 Reyan Ahmed, Faryad Darabi Sahneh, Keaton Hamm, Stephen Kobourov, and Richard Spence. Kruskal-based approximation algorithm for the multi-level Steiner tree problem. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms (ESA 2020)*, volume 173 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:21, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2020.4.

8 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28:1167–1181, 1999.

9 Ingo Althöfer, Gautam Das, David Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In *Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 26–37, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

10 Anantaram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Sci.*, 40(7):846–867, 1994. doi:10.1287/mnsc.40.7.846.

11 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

12 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):5, 2010.

13 Greg Bodwin. Linear size distance preservers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 600–615. Society for Industrial and Applied Mathematics, 2017.

14 Greg Bodwin. A note on distance-preserving graph sparsification. *arXiv preprint arXiv:2001.07741*, 2020.

15 Greg Bodwin and Virginia Vassilevska Williams. Better distance preservers and additive spanners. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 855–872. Society for Industrial and Applied Mathematics, 2016. URL: http://dl.acm.org/citation.cfm?id=2884435.2884496.

16 Hsien-Chih Chang, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Near-Optimal Distance Emulator for Planar Graphs. In *Proceedings of 26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, pages 16:1–16:17, 2018.

17 M. Charikar, J. Naor, and B. Schieber. Resource optimization in QoS multicast routing of real-time multimedia. *IEEE/ACM Transactions on Networking*, 12(2):340–348, April 2004. doi:10.1109/TNET.2004.826288.

18 Shiri Chechik. New additive spanners. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512. Society for Industrial and Applied Mathematics, 2013.

19 Eden Chlamtáč, Michael Dinitz, Guy Kortsarz, and Bundit Laekhanukit. Approximating spanners and directed Steiner forest: Upper and lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 534–553. SIAM, 2017.

20 Julia Chuzhoy, Anupam Gupta, Joseph (Seffi) Naor, and Amitabh Sinha. On the approximability of some network design problems. *ACM Trans. Algorithms*, 4(2):23:1–23:17, 2008. doi:10.1145/1361192.1361200.

21 Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

22 Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On pairwise spanners. In *Proceedings of 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20, pages 209–220, 2013.

23 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Almost shortest paths and PRAM distance oracles in weighted graphs. *arXiv preprint arXiv:1907.11422*, 2019.

24 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Improved weighted additive spanners. *arXiv preprint arXiv:2008.09877*, 2020.

25 Paul Erdős and Alfréd Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

26 Marek Karpinski, Ion I. Mandoiu, Alexander Olshevsky, and Alexander Zelikovsky. Improved approximation algorithms for the quality of service multicast tree problem. *Algorithmica*, 42(2):109–120, 2005. doi:10.1007/s00453-004-1133-y.

27 Telikepalli Kavitha. New pairwise spanners. *Theory of Computing Systems*, 61(4):1011–1036, 2017.

28 Telikepalli Kavitha and Nithin M. Varma. Small stretch pairwise spanners and approximate d-preservers. *SIAM Journal on Discrete Mathematics*, 29(4):2239–2254, 2015.

29 Arthur Liestman and Thomas Shermer. Additive graph spanners. *Networks*, 23:343–363, July 1993. doi:10.1002/net.3230230417.

30 Prakash Mirchandani. The multi-tier tree problem. *INFORMS J. Comput.*, 8(3):202–218, 1996.

31 Mathew Penrose. *Random geometric graphs*. Number 5. Oxford University Press, 2003.

32 Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6(1):7, 2009.

33 Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440, 1998.

A Pairwise spanner constructions [3]

Here, we provide pseudocode (Algorithms 1–3) describing the $+2W$, $+4W$, and $+8W$ pairwise spanner constructions[3] by Ahmed et al. [3]. These spanner constructions have a similar theme: first, construct a *d-light initialization*, which is a subgraph H obtained by adding the d lightest edges incident to each vertex (or all edges if the degree is at most d). Then for each pair $(s,t) \in P$, consider the number of edges in $\pi(s,t)$ which are absent in the current subgraph H. Add $\pi(s,t)$ to H if the number of missing edges is at most some threshold ℓ, or otherwise randomly sample vertices and either add a shortest path tree rooted at these vertices, or construct a subsetwise spanner among them.

Algorithm 1 $+2W$ pairwise spanner [3].

$d = |P|^{1/3}$, $\ell = n/|P|^{2/3}$
$H = d$-light initialization
let m' be the number of missing edges needed for a valid construction
while $m' > nd$ **do**
    for $(s,t) \in P$ **do**
        $x = |E(\pi(s,t)) \setminus E(H)|$
        if $x \leq \ell$ **then**
            add $\pi(s,t)$ to H
    $R =$ random sample of vertices, each with probability $1/(\ell d)$
    for $r \in R$ **do**
        add a shortest path tree rooted at r to each vertex
add the m' missing edges
return H

[3] Using a tighter analysis or the above $+2W$ subsetwise construction in place of the $+4W$ construction in Algorithm 3, the additive error can be improved to $+2W(\cdot,\cdot)$, $+4W(\cdot,\cdot)$, and $+6W$ for integer edge weights.

Algorithm 2 $+4W$ pairwise spanner [3].

$d = |P|^{2/7}$, $\ell = n/|P|^{5/7}$
$H = d$-light initialization
let m' be the number of missing edges needed for a valid construction
while $m' > nd$ **do**
    for $(s,t) \in P$ **do**
        $x = |E(\pi(s,t)) \setminus E(H)|$
        if $x \leq \ell$ **then**
            add $\pi(s,t)$ to H
        else if $x \geq n/d^2$ **then**
            R_1 = random sample of vertices, each w.p. d^2/n
            add a shortest path tree rooted at each $r \in R_1$
        else
            add first ℓ and last ℓ missing edges of $\pi(s,t)$ to H
            R_2 = i.i.d. sample of vertices, w.p. $1/(\ell d)$
            for each $r, r' \in R_2$ **do**
                if exists $r \to r'$ path missing $\leq n/d^2$ edges **then**
                    add to H a shortest $r \to r'$ path among paths missing $\leq n/d^2$ edges
add the m' missing edges
return H

Algorithm 3 $+8W$ pairwise spanner [3].

$d = |P|^{1/4}$, $\ell = n/|P|^{3/4}$
$H = d$-light initialization
let m' be the number of missing edges needed for a valid construction
while $m' > nd$ **do**
    for $(s,t) \in P$ **do**
        $x = |E(\pi(s,t)) \setminus E(H)|$
        if $x \leq \ell$ **then**
            add $\pi(s,t)$ to H
        else
            add first ℓ and last ℓ missing edges of $\pi(s,t)$ to H
            R = random sample of vertices, each w.p. $1/(\ell d)$
            $H' = +4W$ subsetwise $(R \times R)$-spanner [3]
            add H' to H
add the m' missing edges
return H

B Experiments

In the main paper, we mostly discussed the experimental results of Erdős–Rényi graphs. In this section, we provide the results of random geometric graphs. The plots of Watts–Strogatz and Barabási–Albert graphs are available in the Github repository.

B.1 Multi-level $+2W$ spanner

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and terminal selection methods in Figure 17. In both cases the average ratio increases as n and ℓ increases. The average ratio is relatively lower for the linear terminal selection method.

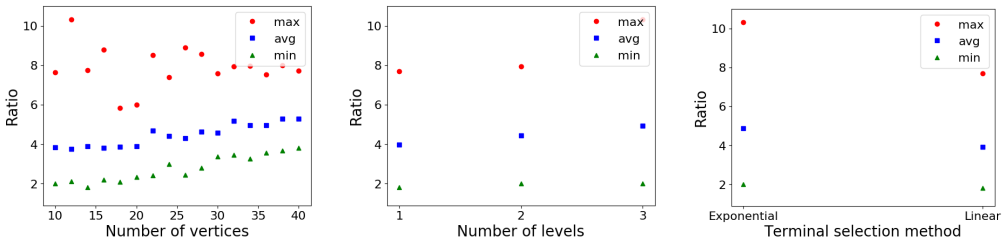

Figure 17 Performance of the algorithm that uses $+2W$ subsetwise spanner as the single level subroutine on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.2 Multi-level $+2W(\cdot,\cdot)$ spanner

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and terminal selection method in Figure 18. The average experimental ratio increases as n increases. The maximum ratio increases as ℓ increases. Again, the experimental ratio of the linear terminal selection method is relatively smaller compared to the exponential method.

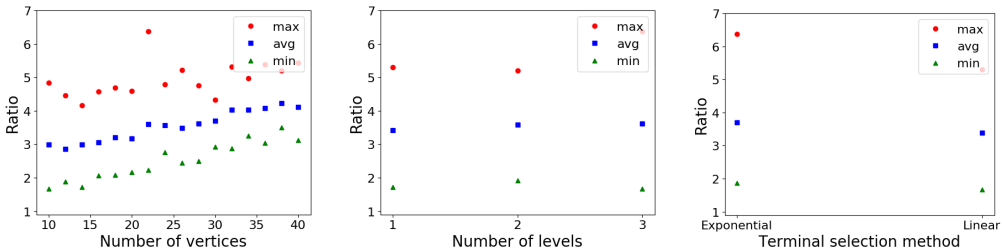

Figure 18 Performance of the algorithm that uses $+2W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.3 Comparison between global and local error

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and the terminal selection method in Figure 19. The ratio of the local setting is smaller compared to the global setting.

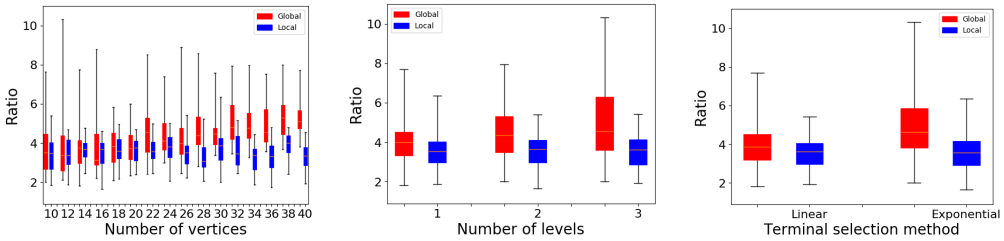

Figure 19 Performance of the global and local construction-based algorithms on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.4 Multi-level $+4W(\cdot,\cdot)$ spanner

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and terminal selection method in Figure 20. The experimental ratio increases as the number of vertices increases. The maximum ratio increases as the number of levels increases. Again, the experimental ratio of the linear terminal selection method is relatively smaller compared to the exponential method.

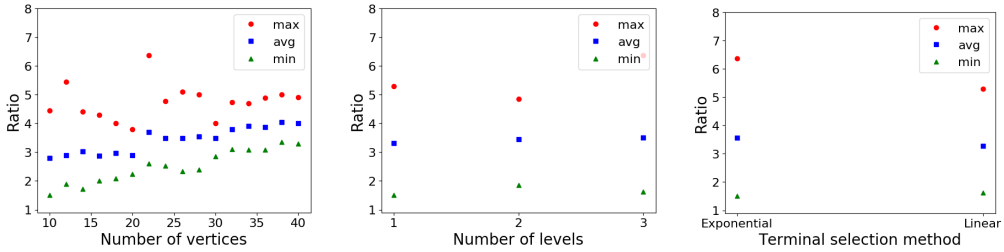

Figure 20 Performance of the algorithm that uses $+4W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.5 Comparison between $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ setups

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and the terminal selection method in Figure 21. As n increases the average ratio of $+4W(\cdot,\cdot)$-based approximation algorithm becomes smaller compared to the $+2W(\cdot,\cdot)$-based algorithm. The average ratio of $+4W(\cdot,\cdot)$ is relatively smaller for the exponential terminal selection method.

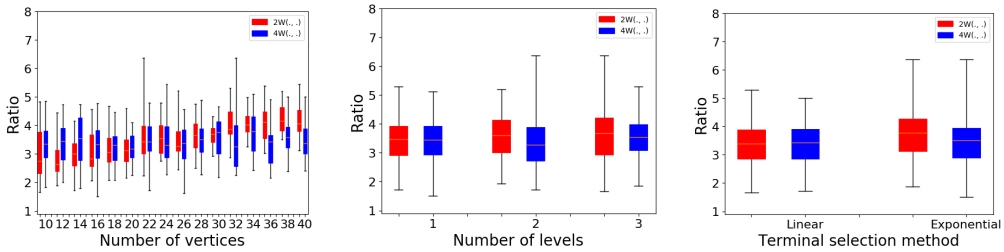

Figure 21 Performance of the pairwise $+2W(\cdot,\cdot)$ and $+4W(\cdot,\cdot)$ construction-based algorithms on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.6 Multi-level $+6W$ spanner

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and terminal selection method in Figure 22. The experimental ratio increases as the number of vertices increases. The maximum ratio increases as the number of levels increases. Again, the experimental ratio of the linear terminal selection method is relatively smaller compared to the exponential method.

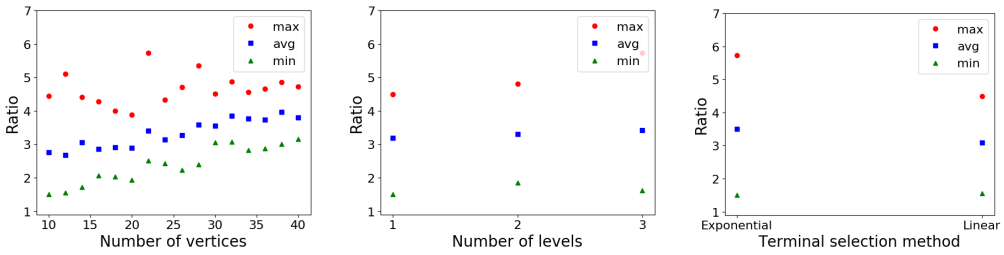

Figure 22 Performance of the algorithm that uses $+6W(\cdot,\cdot)$ pairwise spanner as the single level subroutine on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.7 Comparison between $+2W$ and $+6W$ setups

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and the terminal selection method in Figure 23. We can see that as n gets larger the ratio of $+6W$ gets smaller. The situation is similar when ℓ increases.

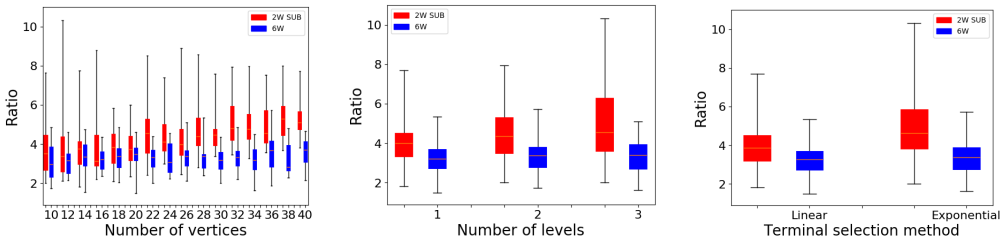

Figure 23 Performance of the pairwise $+2W$ and $+6W$ construction-based algorithms on random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.8 Experiment on large graphs

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and the terminal selection method in Figure 24.

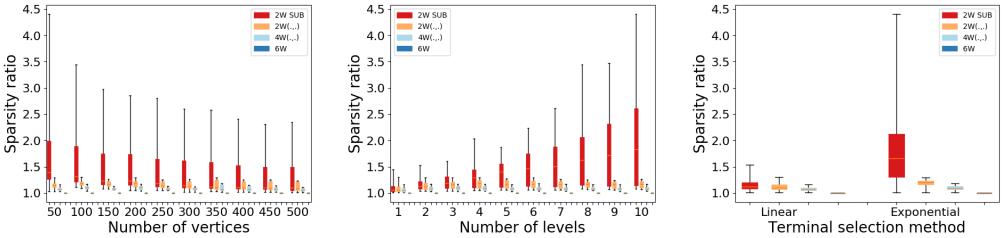

Figure 24 Performance of different approximation algorithms on large random geometric graphs w.r.t. n, ℓ, and terminal selection method.

B.9 Impact of the initialization parameters

We describe the experimental results on random geometric graphs w.r.t. n, ℓ, and the terminal selection method in Figure 25. Again, the experiment suggests that we can exponentially reduce the value of d and take the solution that has a minimum number of edges, with an additional cost of $O(\log d)$ running time.

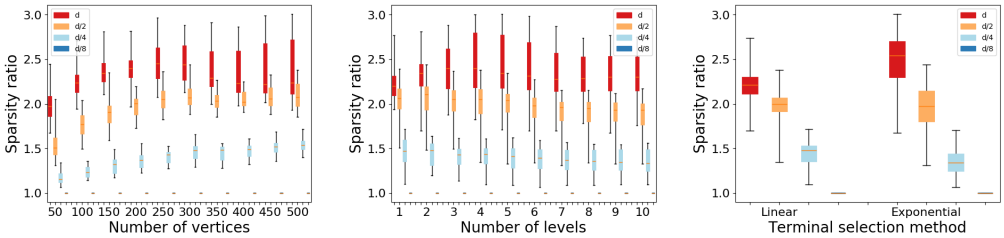

Figure 25 Impact of different values of d on large random geometric graphs w.r.t. n, ℓ, and terminal selection method.