Multi-Level Graph Spanners

January 9, 2019

Abstract

TBC. Design an algorithm to create a multi-level graph representation based on the corresponding single-level problem: Clearly define the multi-level problem (e.g., what is the input, what is being optimized, etc. Analyze the complexity of the problem. Deploy the multi-level algorithm as a subroutine in multi-level graph layout.

Contents

1 Introduction

1.1 Background

Given a graph $G(V, E)$, a subgraph $G'(V, E')$ is a (multiplicative) *t-spanner* of G, if for every $u, v \in V$, the distance from u to v in G' is at most t times longer than the distance in G. We refer to t as the *stretch factor* of G'. Peleg et al. [8] have shown that determining if there exists a t-spanner of G with m or fewer edges is NP-complete. Althöfer et al. [2] have provided an approximation algorithm to construct a t-spanner of an input graph. The following theorem provides the properties of a spanner generated by their algorithm:

FS: Add the greedy spanner algorithm

Theorem 1. Given a n-vertex graph G and a $t \geq 1$, there is a polynomially constructible $(2t + 1)$-spanner G' such that

- $Size(G') < n.\lceil n^{\frac{1}{t}} \rceil$,

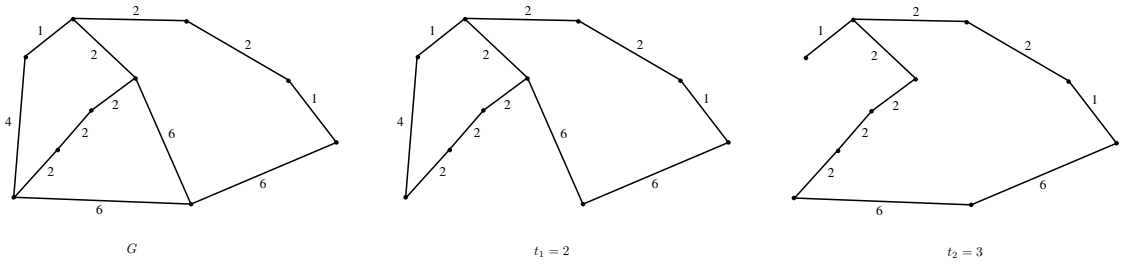


Figure 1: Independent inconsistent spanners

- $Weight(G') < Weight(MST(G)).(1 + \frac{n}{2t})$.

A more general version of the t-spanner problem is the *pairwise spanner* problem [7], where the stretch factor only needs to be preserved for a subset $\mathcal{P} \subseteq V \times V$ of pairs of vertices. Thus, the classical t-spanner problem is a special case of the pairwise spanner problem, with $\mathcal{P} = V \times V$. The *subsetwise spanner* problem aims to preserve the stretch factor between only a subset of vertices, i.e. $\mathcal{P} = S \times S$ for some $S \subseteq V$. add related work We will call the vertices in S *terminals*. The classical *Steiner tree* problem on graphs is a special case of the subsetwise spanner problem, where t is arbitrarily large.

1.2 Multi-level Graph Spanners

to simplify the problem, Reyan and I agreed to use a single stretch factor t for all levels - RCS

Similar to other graph problems which generalize to multiple levels or grades of service add citations, we extend the subsetwise spanner problem similarly, which we call the *multi-level graph spanner (MLGS)* problem: given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}^+$, a nested sequence $T_\ell \subset T_{\ell-1} \subset \ldots \subset T_1 \subseteq V$, and real number $t \geq 1$, compute a sequence of spanners $G_\ell \subseteq G_{\ell-1} \subseteq \ldots \subseteq G_1$, where G_i is a subsetwise $(T_i \times T_i)$-spanner with multiplicative stretch factor t.

This problem can be stated more naturally in terms of rates of service: given $G = (V, E)$ with edge weights, and rates of service (priorities) $R : V \to \{0, 1, \ldots, \ell\}$, compute a subgraph $H \subseteq G$ with edge priorities satisfying the following property: for all $u, v \in V$, if u and v each have rate of service greater than or equal to i, there exists a t-spanner path from u to v whose edges have priorities greater than or equal to i. If $priority(e)$ denotes the priority of edge e, we define the cost (weight) of a solution by $\sum_{e \in H} w(e) \cdot priority(e)$, that is, edges with higher priority incur a greater cost. This interpretation makes it clearer that more important vertices (hubs) are connected with higher quality.

it is a good idea to address: why are we solving this problem? what is the motivation behind the multi-level subsetwise spanner problem?

An immediate approach to compute a multi-level spanner of a graph G is to run the algorithm mentioned above l times independently with stretch factors $t_1, t_2, \ldots, t_l$ where every time the input graph of the algorithm is G. In this case the total number of edges is $n^{1+\frac{1}{t_1}} + n^{1+\frac{1}{t_2}} + \cdots + n^{1+\frac{1}{t_l}}$. If we assume $t_i = i$, then the total number of edges is $n^{1+\frac{1}{2}} + n^{1+\frac{1}{3}} + \cdots + n^{1+\frac{1}{l}} < n^{1+\frac{1}{2}}(1 + \frac{1}{n^{\frac{1}{6}}} + \frac{1}{n^{\frac{1}{4}}} + \frac{1}{n^{\frac{3}{10}}} + \cdots) \leq (1+\epsilon)n^{1+\frac{1}{2}}$, where $\epsilon \leq 2$. However the spanners generated by this approach may not have the property that t_{i+1}-spanner $\subseteq t_i$-spanner as shown by example in Figure 1

Hence instead of generating the spanners independently from G we go through an iterative approach. We first generate the spanner from G having stretch factor t_1. Now in order to generate a t_2-spanner, instead of giving G as input we input the t_1-spanner we have computed in the previous step. Now we have to analyze what is the appropriate input stretch factor to

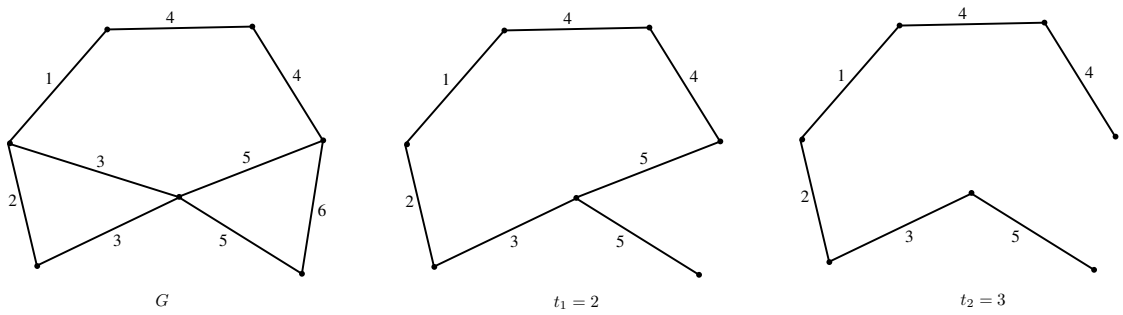


Figure 2: Spanners generated by iterative approach

generate a t_2-spanner of G from the t_1-spanner. Filtser et al. [12] have provided the following theorem that provides the relation between iterative spanners.

Lemma 1. Let $G = (V, E, w)$ be any weighted graph, let $t > 1$ be any stretch parameter, and let H be the greedy t-spanner of G. If H' is a t-spanner for H, then $H' = H$.

Assume that we have two stretch factors t_1 and t_2 for two levels. We first generate a t_1-spanner from the input graph G. Now we generate another spanner G' from the t_1-spanner. From the above lemma we know that if $t_1 = t_2$, then G' is a t_2-spanner of G. In most of the applications the stretch factor of two levels are different ($t_1 \neq t_2$). We want to find the relationship between the spanner of second level and the input graph G (Is it a t_2-spanner of G?). In the following lemma we show that G' may not be a t_2-spanner when $t_1 \neq t_2$.

Lemma 2. Let $G = (V, E, w)$ be any weighted graph, let $1 < t_1 < t_2$ be two stretch parameters, and let G_1 be the greedy t_1-spanner of G. If we compute a greedy spanner G_2 from G_1 with stretch parameter t_2, then G_2 may not be a t_2-spanner of G.

Proof. We provide a pictorial proof as shown in Figure 2. □

In the following lemma we provide a stronger result.

Lemma 3. Let $G = (V, E, w)$ be any weighted graph, let $1 < t_1 < t_2$ be two stretch parameters, and let G_1 be the greedy t_1-spanner of G. If we compute a greedy spanner G_2 from G_1 with stretch parameter t_2, then G_2 is a $(t_1 t_2)$-spanner of G.

Proof. We provide a pictorial proof as shown in Figure 3. □

FS: this is missing the part that no that the stretch factor is not larger than $t_1 t_2$.

In the problem defined previously we compute a spanner for every level. All spanners have the same vertex set V, the vertex set of the input graph G. In many practical scenarios instead of having same vertex set in every level we are interested to have a hierarchy of vertex sets. Hence we now define a variation of this problem called *multi-level spanner with nested terminal sets*. Given an input graph G, a sequence of positive integers $t_1, t_2, \ldots, t_l$ (Note that we have not mentioned the relationship among the stretch factors. We will consider either $t_1 < t_2 < \ldots < t_l$ or $t_1 > t_2 > \ldots > t_l$.) and l nested terminal sets $T_1 \subset T_2 \subset \ldots \subset T_l \subset V$, a $(t_1, t_2, \ldots, t_l)$-spanner with nested terminal sets is a series of spanners $G_1, G_2, \ldots, G_l$ such that

1. The stretch factor of G_i is equal to t_i
2. $G_{i-1} \subseteq G_i$
3. The vertex set of G_i contains T_i
4. The sum $\sum_{i \in \{1,2,\ldots,l\}} c(t_i\text{-spanner})$ is minimum.

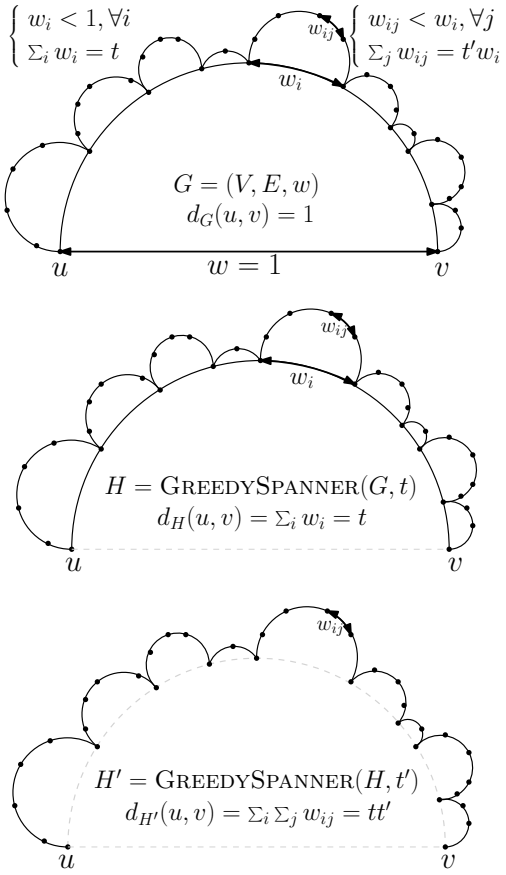


Figure 3: Spanners generated by iterative approach

So far, we have discussed multiplicative spanners. We now discuss additive spanners which are the class of spanners that has been got focused in more recent research works. We first present the definition of a single level additive spanner from literature. We then define multi-level additive spanner. Let $G(V, E)$ be an unweighted graph. We denote the shortest path distance between two vertices $u, v \in V$ in G by $d_G(u, v)$. A subgraph $G'(V, E')$ is an additive t-spanner of G, if for every $u, v \in V$, $d_{G'}(u, v) \leq d_G(u, v) + t$. We refer to t as the *additive stretch factor* of G'. We now define the *multi-level additive spanner problem*. Given an input graph G, an integer $t_1 < t_2 < \cdots < t_l$ and l nested terminal sets $T_1 \subset T_2 \subset \ldots \subset T_l \subset V$, a multi-level additive $(t_1, t_2, \cdots, t_l)$-spanner is a series of spanners $G_1, G_2, \ldots, G_l$ such that

1. The additive stretch factor of G_i is equal to t_i
2. $G_{i-1} \subseteq G_i$
3. The vertex set of G_i contains T_i
4. The sum $\sum_{i \in \{1,2,\ldots,l\}} c(t_i\text{-spanner})$ is minimum.

1.2.1 A Simple Multi-level Additive Spanner

We can consider a simple version of multi-level additive spanner problem where every level have same additive stretch: $t_1 = t_2 = \cdots = t_l = 2$. Also there is a pattern on the size of the terminal sets. The size of the terminal sets increase exponentially as we go from top to bottom: $|T_1| = 2, |T_2| = 4, \ldots, |T_l| = 2^l$. Cygan et al. [**?**] have provided an algorithm to compute subset-wise spanners with additive stretch 2 having size $O(n\sqrt{|S|})$ for unweighted graphs. We can run this algorithm independently on each level with subset equal to T_i. As the size of the terminal set is increasing exponentially, the size of this multi-level spanner is $O(n\sqrt{n})$. However, as we have discussed for multiplicative spanners, if we compute the spanners independently, then an edge present in a higher level may not present in all lower levels. We can deal this situation by putting that edge in all missing levels which will increase the size of the spanner. We can show that the size of the spanner is $O(n\sqrt{n}\log n)$.

1.3 Related Work

Sigurd et al. [19] have presented an exact algorithm based on an ILP formulation to compute spanners. They have constructed minimum weight spanners of arbitrary graphs using their ILP formulation. They have run an experiment to show the practical performance of greedy algorithm [2] compared to the exact algorithm. Kortsarz et al. have presented an approximation algorithm for the problem of finding the sparsest 2-spanner in a given graph with approximation ratio $\log(\frac{|E|}{|V|})$ [16]. The unweighted k-spanner problem admits $O(1)$ approximation once the stretch requirement becomes $k = \Omega(\log n)$ [10]. Dinitz et al. [9] have provided a flow-based linear programming relaxation to approximate the directed spanner problem. Their LP formulation is similar to the previous formulation [19]. However, they have provided an approximation algorithm relaxing their ILP whereas the previous formulation was used to get an exact algorithm. One major difference between two formulations is, in the formulation of Sigurd et al. [19] they consider the minimum weighted spanner problem whereas in the other formulation they solve the unweighted spanner problem. For example, in the first formulation the objective function is minimize $\sum_{e \in E} c_e x_e$ whereas in the other formulation the objective function is minimize $\sum_{e \in E} x_e$, here c_e is the weight of edge e and x_e is a variable that represents whether e is in the spanner or not. Hence, in the formulation of Dinitz et al. they do not care about the weight of the edges, they later take care of it in their rounding algorithm by solving shortest path arborescence problem. Also Dinitz et al. [9] have considered the directed version of the spanner problem. They have provided a $O(n^{\frac{2}{3}})$-approximation algorithm for the directed k-spanner problem for $k \geq 1$. This is the first sublinear approximation algorithm for arbitrary edge lengths. This algorithm has two major part: linear programming relaxation and sampling based on the shortest path arborescence problem. Bhattacharyya et al. [5] have provided a slightly different formulation

to approximate t-spanners as well as other variations of this problem. Their algorithm is very much similar to the previous work. They also have two major part as in the previous algorithm although theier linear programming conastraints are slightly different. However this work has in interesting aspect, they have later derandomized the sampling technique they used. They have provided a $O((n \log n)^{1-\frac{1}{k}})$-approximate deterministic polynomial time algorithm for the directed k-spanner problem. Berman et al. [4] have provided alternative randomized LP rounding schemes that lead to better approximation ratios. They have improved the approximation ratio to $O(\sqrt{n} \log n)$ where the approximation ratio of the algorithm provided by Dinitz et al. [9] was $O(n^{\frac{2}{3}})$. They have also improved the approximation ratio for the important special case of directed 3-spanners with unit edge lengths.

In the second formulation of this problem, where the vertex set varies from level to level, we are actually solving a different kind of spanner problem where we are only interested to maintain the shortest paths between some specific pairs of vertices. Note that in the traditional spanner problem we are interested in maintaining the shortest paths between all pair of veritces. Klein [15] has shown that for any edge-weighted planar graph G and a subset S of nodes of G, there is a subgraph H of G of weight a constant times that of the minimum Steiner tree for S such that distances in H between nodes in S are at most $1 + \epsilon$ times the corresponding distances in G, where $\epsilon > 0$. Cygan et al. [**?**] have provided algorithms to compute subset-wise and pair-wise additive spanners for unweighted graphs. They have shown that there exists an additive $\mathcal{P}$-spanner of size $O(n|\mathcal{P} \log n|^{\frac{1}{4}})$ with $f(d) = d + 4 \log n$. They have also shown how to construct $O(n\sqrt{|S|})$ size subset-wise additive spanner for stretch 2 where $|S|$ is the size of the subset. Kavitha [14] have provided an algorithm to compute an additive 2 $\mathcal{P}$-spanner of size $O(n|\mathcal{P}|^{\frac{1}{3}})$. Abboud et al. [1] have provided a tight lower bound on the sparsity of additive spanners. Kavitha [13] has studied pairwise spanners with additive stretch factors 4 and 6. Bodwin et al. [6] have provided an upper bound of the size of subset-wise spanners with polynomial additive stretch factor. They have provided the best sparsity and additive error trade off.

2 Approximate Algorithms for MLGS

As the MLGS problem generalizes the subsetwise spanner problem, which in turn is a generalization of the NP-hard t-spanner problem, MLGS is NP-complete. Here, we will assume a subroutine that computes an optimal subsetwise spanner, given a graph G, subset $S \subseteq V$, and t (for example, an ILP). The intent is to determine if approximating MLGS is significantly harder than the subsetwise spanner problem.

1. Compute the spanners in every level independently. Include an edge by force in all levels below it, if it is present in the current level.
2. Compute the spanners iteratively.

The second variant is relatively harder than the first variant. Here, we have a nested set of vertices $V_1 \subset V_2 \subset \cdots \subset V_l$ where V_i is the terminal set of level l. We formulate a top-down and bottom-up approach to the MLGS problem, in a way similar to the MLST problem.

A Bottom up approach: The approach is as follows: compute a subsetwise $(T_1 \times T_1)$-spanner of G with stretch factor t. This induces a feasible solution to the MLGS problem, as we can simply copy each edge from the spanner to every level (or, in terms of priorities, assign priority ℓ to each spanner edge). We can then try to prune edges afterwards to achieve a better solution. However, we first show that this simple approach is a trivial ℓ-approximation to the MLGS problem. Let OPT denote the cost of the optimal MLGS $G^*_\ell \subseteq G^*_{\ell-1} \subseteq \ldots \subseteq G^*_1$ on a graph G. Let MIN_i denote the minimum cost of a subsetwise $(T_i \times T_i)$-spanner for level i, and let BOT denote the cost computed by the bottom-up approach. If no pruning is done, then $\text{BOT} = \ell\text{MIN}_1$.

Lemma 4. $\text{BOT} \leq \ell \cdot \text{OPT}$.

Proof. We have $\text{MIN}_1 \leq \text{OPT}$ trivially, since the lowest-level graph G_1^* is a $(T_1 \times T_1)$-spanner whose cost is at least MIN_1. Further, we have $\text{BOT} = \ell\text{MIN}_1$ if no pruning is done. Then $\text{MIN}_1 \leq \text{OPT} \leq \text{BOT} = \ell \cdot \text{MIN}_1$, so $\text{BOT} \leq \ell \cdot \text{OPT}$. □

The ratio of ℓ is tight, as shown by example in Figure **??**. Note that in this example, no edges can be pruned without violating the t-spanner requirement.

We give a simple heuristic that attempts to "prune" unneeded edges without violating the t-spanner requirement. Note that any pruning strategy does not improve the approximation ratio, as a worst case example (Figure **??**) cannot be pruned. Let G_1 be the $(T_1 \times T_1)$-spanner computed by the bottom-up approach. To compute a $(T_2 \times T_2)$-spanner G_2 using the edges from G_1, we classify the edges in G_1 into three types: i) neither endpoint belongs to T_2, ii) one endpoint belongs to T_2, and iii) both endpoints belong to T_2.

For each type (i), ii), or iii)) in order, sort all edges of that type in descending order of weight. For each edge e in type i), remove e if it does not violate the t-spanner property for any pair of vertices in T_2; otherwise keep it. Proceed with type ii) and iii) edges similarly in order to compute G_2. In the same way, compute G_3 from G_2, and so forth.

below is a very simple pruning strategy that could be considered - is it better or worse than the previous one? Presumably it's much more efficient as APSP only needs to be computed once. In any case it should be mentioned if we mention any pruning. - RCS

An even simpler pruning strategy is as follows: given G_1, let G_i be a distance preserver of G_{i-1} over the terminals T_i, for all $i = 2, \ldots, \ell$. An example is to let G_2 be the union of all shortest paths (in G_1) over vertices $v, w \in G_2$. The result is clearly a feasible solution to the MLGS problem, as the shortest paths are preserved exactly from G_1, so each G_i is a $(T_i \times T_i)$ spanner of G with stretch factor t.

there is a lot of literature on distance preservers, should look into these:
1. Sparce Sourcewise and Pairwise Distance Preservers (Coppersmith, Elkin 2006)
2. Linear Size Distance Preservers (Bodwin 2016) - RCS

A top down approach: A simple top-down heuristic that computes a solution $(G_\ell, G_{\ell-1}, \ldots, G_1)$ is as follows: let G_ℓ be the minimum cost $(T_\ell \times T_\ell)$-spanner over terminals T_ℓ and stretch factor t, with cost MIN_ℓ. Then, compute a minimum cost $(T_{\ell-1} \times T_{\ell-1})$-spanner over $T_{\ell-1}$, and let $G_{\ell-1}$ be the union of this spanner and G_ℓ. Continue this process, where G_i is the union of the minimum cost $(T_i \times T_i)$-spanner and G_{i+1}. Clearly, this produces a feasible solution to the MLGS problem.

We show that this simple approach achieves an $\frac{\ell+1}{2}$-approximation, similar to the top-down approach for the MLST problem. Define MIN_i and OPT as before. Define OPT_i to be the total cost of edges on level i but not level $i+1$ in the optimal MLGS solution, so that $\text{OPT} = \ell\text{OPT}_\ell + (\ell-1)\text{OPT}_{\ell-1} + \ldots + \text{OPT}_1$. Define TOP_i analogously. We have the following:

Lemma 5.

$$\text{TOP}_\ell \leq \text{OPT}_\ell$$
$$\text{TOP}_i \leq \text{OPT}_i + \text{OPT}_{i+1} + \ldots + \text{OPT}_\ell$$

Proof. The first inequality $\text{TOP}_\ell \leq \text{OPT}_\ell$ is true by definition, as we compute an optimal $(T_\ell \times T_\ell)$ spanner whose cost is TOP_ℓ. The second inequality follows; note that $\text{TOP}_i \leq \text{MIN}_i$, with equality when the minimum cost $(T_i \times T_i)$ spanner and G_{i+1} are disjoint. As the $(T_i \times T_i)$-spanner of cost $\text{OPT}_i + \text{OPT}_{i+1} + \ldots + \text{OPT}_\ell$ is a feasible t-spanner, we have $\text{MIN}_i \leq \text{OPT}_i + \ldots + \text{OPT}_\ell$, concluding the lemma. □

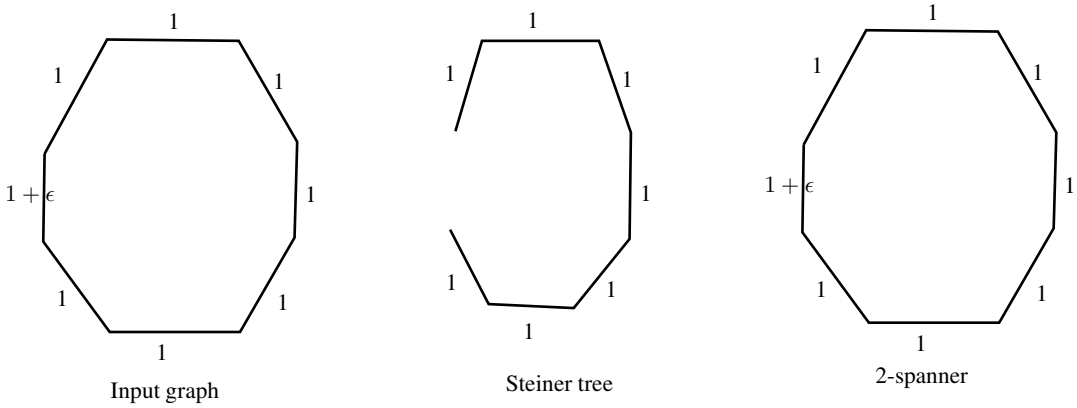


Figure 4: Here the Steiner tree is a 8-spanner but not a 2-spanner. And if we increase the number of vertices in the cycle, then we can show that the Steiner tree is not a valid t-spanner for any value of t. Note that here the edges represent a path, and the vertices represent terminals.

By Lemma 5, we have

$$\begin{aligned}
\text{TOP} &= \ell\text{TOP}_\ell + (\ell-1)\text{OPT}_{\ell-1} + \ldots + \text{TOP}_1 \\
&\leq \ell\text{OPT}_\ell + (\ell-1)(\text{OPT}_{\ell-1} + \text{OPT}_\ell) + \ldots + (\text{OPT}_1 + \text{OPT}_2 + \ldots + \text{OPT}_\ell) \\
&= \frac{\ell(\ell+1)}{2}\text{OPT}_\ell + \frac{(\ell-1)\ell}{2}\text{OPT}_{\ell-1} + \ldots + \frac{1\cdot 2}{2}\text{OPT}_1 \\
&\leq \frac{\ell+1}{2}\text{OPT}
\end{aligned}$$

using the definition $\text{OPT} = \ell\text{OPT}_\ell + (\ell-1)\text{OPT}_{\ell-1} + \ldots + \text{OPT}_1$.

For the MLST problem, the approximation ratio of $\frac{\ell+1}{2}$ is tight, and an example was given showing tightness. As the MLST problem is a special case of the MLGS problem where t is arbitrarily large (or infinite), the ratio $\frac{\ell+1}{2}$ here is also asymptotically tight.

insert picture of a worst case example? I also have a very simple example showing how TD is off by a factor of $O(\ell)$ for MLGS - RCS

3 Minimum Weight MLGS: an ILP Formulation

For the exact algorithms we first present the ILP formulation found in the literature [19]. We are given a connected undirected graph $G = (V, E)$ with edge weights c_e, $e \in E$, a set of node pairs $K = \{(u_i, v_i)\}, i = 1, \cdots, k$, and a stretch factor $t > 1$. We want to find a graph $G' = (V, E')$ such that the size of $E' \in E$ is minimum and for every pair $(u_i, v_i) \in K$ if we take the shortest path between (u_i, v_i) in G', then it is no more than t times than the shortest path in G. If $K = V \times V$, then it is exactly the same problem of finding a spanner of G. Before providing the ILP we first discuss some terminology that is used in the formulation. We denote the set of paths between u and v with length smaller than or equal to t times the shortest path in input graph by P_{uv} for all $(u, v) \in K$, and let $P = \cup_{(u,v)\in K} P_{uv}$. The indicator variable δ_p^e states whether the edge e is present in p or not (where $p \in P$). The decision variables x_e indicate whether the edge e is present in the spanner or not. The decision variables y_p indicate whether the path p is present in the spanner or not. Then the problem can be formulated as follows:

$$\text{minimize} \sum_{e\in E} c_e x_e \tag{1}$$

Subject to:

$$\sum_{p \in P_{uv}} y_p \delta_p^e \leq x_e, \forall e \in E, \forall (u,v) \in K \tag{2}$$

$$\sum_{p \in P_{uv}} y_p \geq 1, \forall (u,v) \in K \tag{3}$$

$$x_e = \{0,1\}, \forall e \in E \tag{4}$$

$$y_p = \{0,1\}, \forall p \in P \tag{5}$$

Since the number of variables may be exponential in the input size, we will not use the above formulation. Instead, we will solve the restricted master problem (RMP):

$$\text{minimize} \sum_{e \in E} c_e x_e \tag{6}$$

Subject to:

$$\sum_{p \in P_{uv}} y_p \delta_p^e \leq x_e, \forall e \in E, \forall (u,v) \in K \tag{7}$$

$$\sum_{p \in P_{uv}} y_p \geq 1, \forall (u,v) \in K \tag{8}$$

$$1 \geq x_e \geq 0, \forall e \in E \tag{9}$$

$$1 \geq y_p \geq 0, \forall p \in P' \tag{10}$$

where $P'_{uv} \subseteq P_{uv}, P'_{uv} \neq \emptyset, \forall (u,v) \in K$ and $P' = \cup_{(u,v) \in K} P'_{uv}$. Initially, there may be only one path in P'_{uv}. We will iteratively add paths to P'. This technique is known as delayed column generation. Now we consider the dual linear program with variables π_e^{uv} and $\sigma_{uv}, \forall e \in E, \forall (u,v) \in K$:

$$\text{maximize} \sum_{(u,v) \in K} \sigma_{uv} \tag{11}$$

Subject to:

$$\sum_{(u,v) \in K} \pi_e^{uv} \leq c_e, \forall e \in E \tag{12}$$

$$-\sum_{e \in E} \delta_p^e \pi_e^{uv} + \sigma_{uv} \leq 0, \forall p \in P_{uv}, \forall (u,v) \in K \tag{13}$$

$$\pi_e^{uv} \geq 0, \forall e \in E, \forall (u,v) \in K \tag{14}$$

$$\sigma_{uv} \geq 0, \forall (u,v) \in K \tag{15}$$

We first compute the primal linear program using Simplex method and check whether all the constraints of the dual program are satisfiable or not. If all constraints are satisfied, then according to the weak duality theorem, we have found the optimal solution. Otherwise we have to add a path in P' by solving the constrained shortest path problem.

In multi-level spanner, if an edge is selected in ith level, then it should be also selected in all lower levels. We denote the selection of an edge e at level l by x_e^l where $x_e^l \in \{0,1\}$. For the multi-level variation we add the following constraint:

$$x_e^l \geq x_e^{l-1}, l \in \{2,3,\ldots,k\} \tag{16}$$

The primal linear program remains the same for this formulation; however we introduce another dual variable m_l^e and the corresponding constraints become:

$$\sum_{(u,v) \in K} \pi_e^{uv} + m_l^e \leq c_e, \forall e \in E \tag{17}$$

Although this formulation is for multiplicative spanners, it also works for additive spanners. In this case, We denote the set of paths between u and v with length smaller than or equal to the addition of t and the shortest path in input graph by P_{uv} for all $(u, v) \in K$, and let $P = \cup_{(u,v)\in K} P_{uv}$. We have implemented this algorithm and provided some experimental result in Section 4.2.

4 Numerical Experiments

4.1 Set Up

For this experiment we implement the clustering algorithm provided by Cygan et al. [**?**] for additive stretch 2. We analyze how this algorithm performs in practice. We then use this algorithm to implement a simple version of multi-level spanner as described in Section 1.2.1.

4.1.1 Graph Data Synthesis.

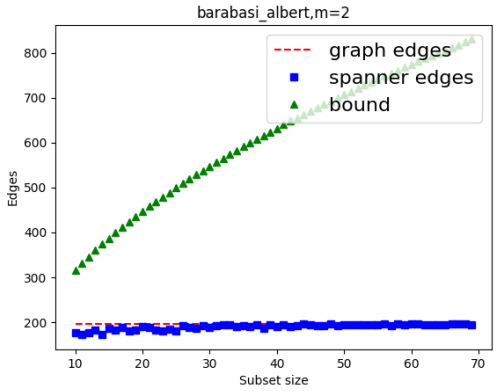
We use different graph generative models to get graph data for our experiment. We consider four random generative models: Barabási–Albert [3], Erdős–Renyi [11], random geometric [18], and Watts–Strogatz [20]. These are the most common graph generative models found in the literature [17]. We now briefly describe each of these models.

The Barabási–Albert model, $\mathrm{BA}(m_0, m)$, generates a growing scale-free network using a preferential attachment mechanism. Initially a graph has m_0 nodes in this model. Then, when a new node appears, it connects to $m \leq m_0$ existing nodes having probability proportional to their instantaneous degree. This BA model uses power-law degree distribution to generate networks, i.e. a small number of nodes become hubs with extremely large degree [3]. This is a network growth model. We let the network grow until a desired network size n is attained in our experiments. The value of m_0 varies from 10 to 100 in our experiments. We keep the value of m equal to 5 or 50. For the single level spanner we keep the value of m_0 equal to 100 and we change the size of the subset from 10 to 100. We keep the value of m equal to 50. For the multi-level spanner we vary the value of m_0 from 10 to 60 and the value of m equal to 5.

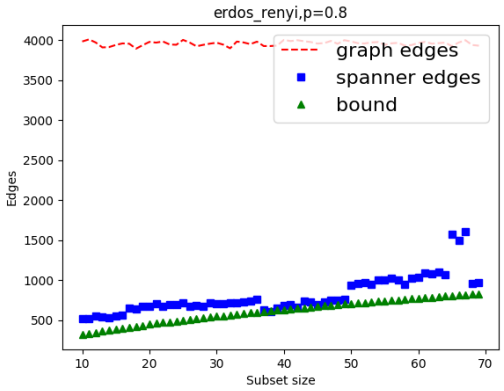
In the Erdős–Renyi model, $\mathrm{ER}(n, p)$, we assign an edge to every possible pair among n nodes with probability p. It is well-known that an instance of $\mathrm{ER}(n, p)$ with $p = (1 + \epsilon)\frac{\ln n}{n}$ is connected with a very high probability for $\epsilon > 0$ [11]. In our experiment, we vary n from 10 to 100 and p is equal to .8.

In the random geometric model, $\mathrm{RG}(n, r_c)$, n nodes are uniformly randomly distributed in the unit square $[0, 1]^2$. In the output graph, two nodes 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 [18]. The value of r_c in our experiment is 0.2.

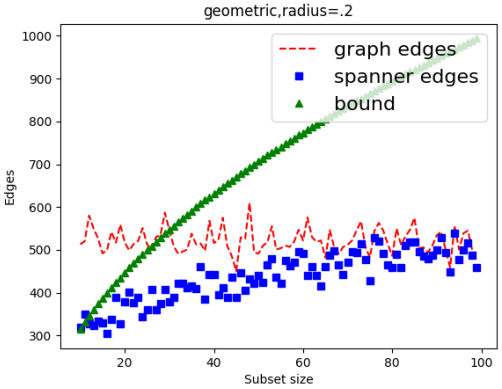
In the Watts-Strogatz model, $\mathrm{WS}(n, K, \beta)$, initially we create a ring lattice of constant degree K, and then rewire each edge with probability $0 \leq \beta \leq 1$ while avoiding self-loops or duplicate edges. The Watts-Strogatz model has an interesting property in that it generates graphs that have the small-world property while having high clustering coefficient [20]. For the single level spanner, we keep the number of nodes equal to 100 and we change the size of the subset from 10 to 100. We keep the value of K and β equal to 10 and 0.5 respectively. For the multi-level spanner we vary the number of nodes from 10 to 60. We keep the value of K and β equal to 6 and 0.2 respectively.



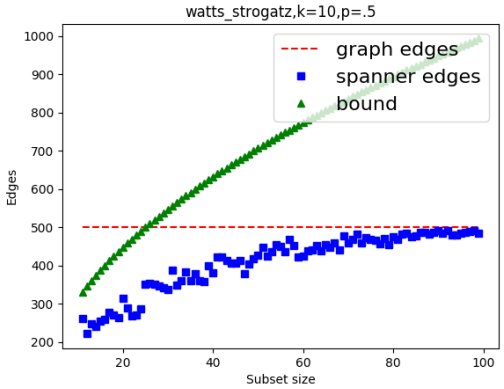
(a) Barabási–Albert



(b) Erdős–Renyi

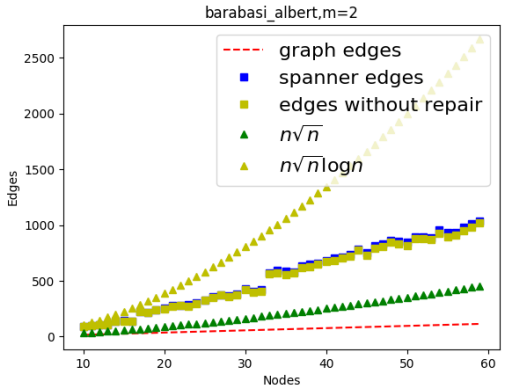


(c) Random Geometric

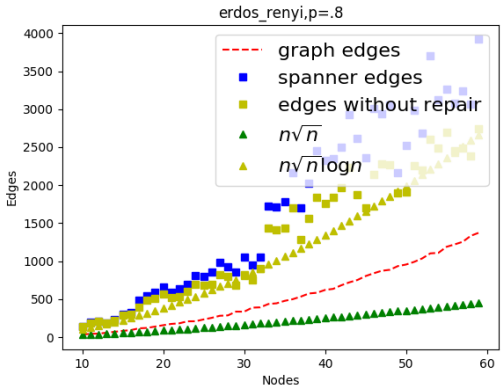


(d) Watts-Strogatz

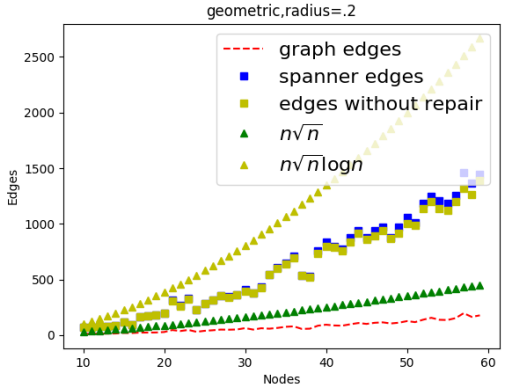
Figure 5: Results for the single level spanner with additive stretch 2



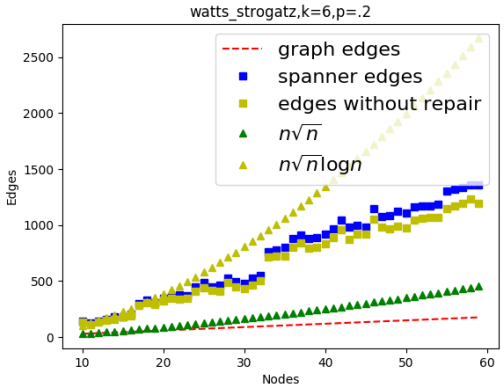
(a) Barabási–Albert



(b) Erdős–Renyi



(c) Random Geometric



(d) Watts-Strogatz

Figure 6: Results for the simple multi-level additive spanner

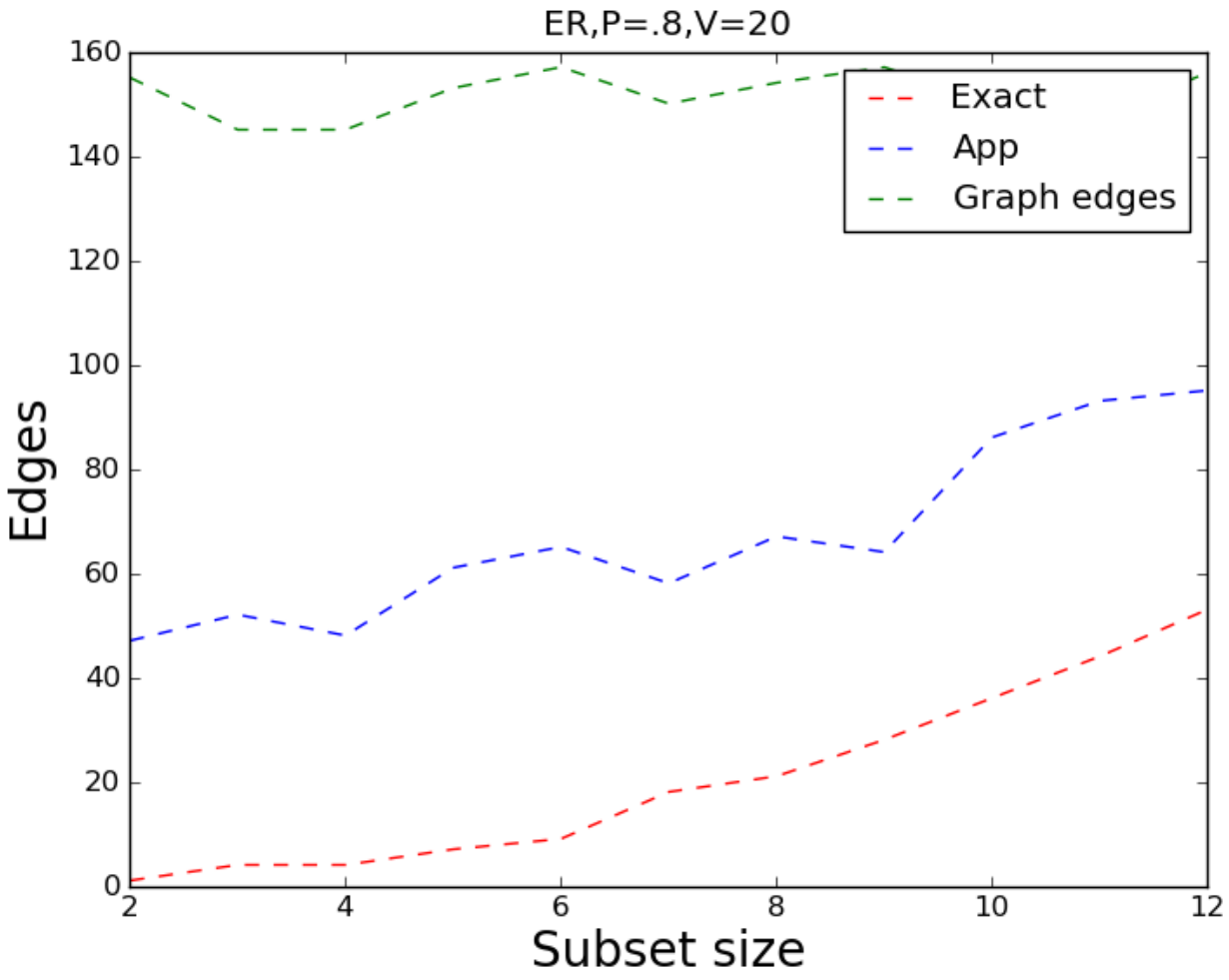


Figure 7: A comparison between the exact solution and the algorithm provided by Cygan et al. [**?**]

4.1.2 Selection of Levels and Terminal Nodes.

4.1.3 Algorithms and Outputs.

4.2 Results

5 Discussions and Conclusion

References

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

[2] Ingo Althöfer, Gautam Das, David Dobkin, and Deborah Joseph. Generating sparse spanners for weighted graphs. In John R. Gilbert and Rolf Karlsson, editors, *SWAT 90*, pages 26–37, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

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

[4] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Approximation algorithms for spanner problems and directed steiner forest. *Information and Computation*, 222:93 – 107, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[5] Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

[6] 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 '16, pages 855–872, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics.

[7] Marek Cygan, Fabrizio Grandoni, and Telikepalli Kavitha. On Pairwise Spanners. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 209–220, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[8] Peleg David and Schäffer Alejandro A. Graph spanners. *Journal of Graph Theory*, 13(1):99–116.

[9] Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 323–332, New York, NY, USA, 2011. ACM.

[10] Michael Elkin and David Peleg. The hardness of approximating spanner problems. *Theory of Computing Systems*, 41(4):691–729, Dec 2007.

[11] Paul Erdös and Alfréd Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[12] Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 9–17, New York, NY, USA, 2016. ACM.

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

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

[15] Philip N. Klein. A subset spanner for planar graphs, with application to subset tsp. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 749–756, New York, NY, USA, 2006. ACM.

[16] G. Kortsarz and D. Peleg. Generating sparse 2-spanners. *Journal of Algorithms*, 17(2):222 – 236, 1994.

[17] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[18] Mathew Penrose. *Random geometric graphs*. Number 5. Oxford university press, 2003.

[19] Mikkel Sigurd and Martin Zachariasen. Construction of minimum-weight spanners. In Susanne Albers and Tomasz Radzik, editors, *Algorithms – ESA 2004*, pages 797–808, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

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