Approximation algorithms and an integer program for multi-level graph spanners

Reyan Ahmed[1], Keaton Hamm[1], Mohammad Javad Latifi Jebelli[1], Stephen Kobourov[1], Faryad Darabi Sahneh[1], and Richard Spence[1]

University of Arizona, USA

Abstract. Given a weighted graph $G(V, E)$ and $t \geq 1$, a subgraph H is a *t–spanner* of G if the lengths of shortest paths in G are preserved in H up to a multiplicative factor of t. The *subsetwise spanner* problem aims to preserve distances in G for only a subset of the vertices. We generalize the minimum-cost subsetwise spanner problem to one where vertices appear on multiple levels, which we call the *multi-level graph spanner* (MLGS) problem, and describe two simple heuristics. Applications of this problem include road/network building and multi-level graph visualization, especially where vertices may require different grades of service.
We formulate a 0–1 integer linear program (ILP) of size $O(|E||V|^2)$ for the more general minimum *pairwise spanner problem*, which resolves an open question by Sigurd and Zachariasen on whether this problem admits a useful polynomial-size ILP. We extend this ILP formulation to the MLGS problem, and evaluate the heuristic and ILP performance on random graphs of up to 100 vertices and 500 edges.

Keywords: Graph spanners · Integer programming · Multi-level graph representation

1 Introduction

Given an undirected edge-weighted graph $G(V, E)$ and a real number $t \geq 1$, a subgraph $H(V, E')$ is a (multiplicative) *t–spanner* of G if the lengths of shortest paths in G are preserved in H up to a multiplicative factor of t; that is, $d_H(u, v) \leq t \cdot d_G(u, v)$ for all $(u, v) \in V \times V$, where $d_G(u, v)$ is the length of the shortest path from u to v in G. We refer to t as the *stretch factor* of H. Peleg et al. [12] show that determining if there exists a t–spanner of G with m or fewer edges is NP–complete. Further, it is NP–hard to approximate the (unweighted) t–spanner problem to within a factor of $O(\log |V|)$, even when restricted to bipartite graphs [15].

In the *pairwise spanner* problem [11], distances only need 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 where $\mathcal{P} = V \times V$. The *subsetwise spanner* problem is a special case of the pairwise spanner problem where $\mathcal{P} = S \times S$ for some $S \subset V$; that is, distances need only be preserved between vertices in S [11]. The case $t = 1$ is known as the *pairwise distance preserver* or *sourcewise distance preserver* problem, respectively [10]. The subsetwise spanner problem where t is arbitrarily large is known as the *Steiner tree* problem on graphs.

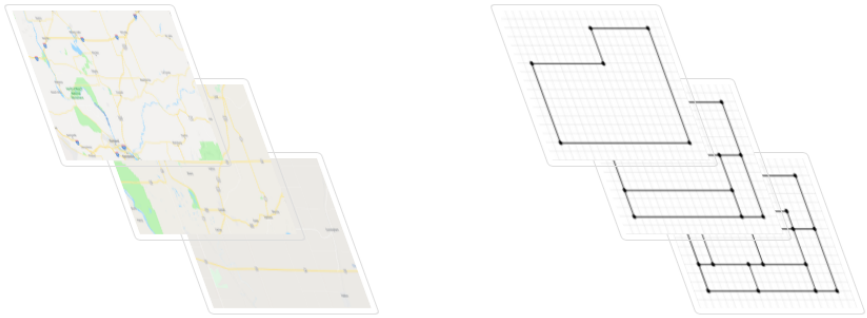


Fig. 1: An interactive road map serves as a good analogy for the MLGS problem, where the top level graph G_ℓ represents the network of major highways, and zooming in to $G_{\ell-1}$ shows a denser network of smaller roads.

1.1 Multi-level graph spanners

In many network design problems, vertices or edges come with a natural notion of priority, grade of service, or level; see Fig. 1. For example, consider the case of rebuilding a transportation infrastructure network after a natural disaster. Following such an event, the rebuilding process may wish to prioritize connections between important buildings such as hospitals or distribution centers, making these higher level terminals, while ensuring that no person must travel an excessive distance to reach their destination. Such problems have been referred to by names such as hierarchical network design, grade of service problems, multi-level, multi-tier, and have applications in network routing and visualization.

Similar to other graph problems which generalize to multiple levels or grades of service [8], we extend the subsetwise spanner problem to the *multi-level graph spanner (MLGS)* problem:

Definition 1. *[Multi-level graph spanner (MLGS) problem] Given a graph $G(V, E)$ with positive edge weights $c : E \to \mathbb{R}_+$, a nested sequence of terminals, $T_\ell \subseteq T_{\ell-1} \subseteq \ldots \subseteq T_1 \subseteq V$, and a real number $t \geq 1$, compute a minimum-cost 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 for G with stretch factor t for $i = 1, \ldots, \ell$. The cost of a solution is defined as the sum of the edge weights on each graph G_i, i.e., $\sum_{i=1}^{\ell} \sum_{e \in E(G_i)} c_e$.*

We refer to T_i and G_i as the terminals and the graph on level i. A more general version of the MLGS problem can involve different stretch factors on each level or a more general definition of cost, but for now we use the same stretch factor t for each level.

An equivalent formulation of the MLGS problem which we use interchangeably involves *grades of service*: given $G = (V, E)$ with edge weights, and required grades of service $R : V \to \{0, 1, \ldots, \ell\}$, compute a single subgraph $H \subseteq G$ with varying grades of service on the edges, with the property that for all $u, v \in V$, if u and v each have required a grade of service greater than or equal to i, then there

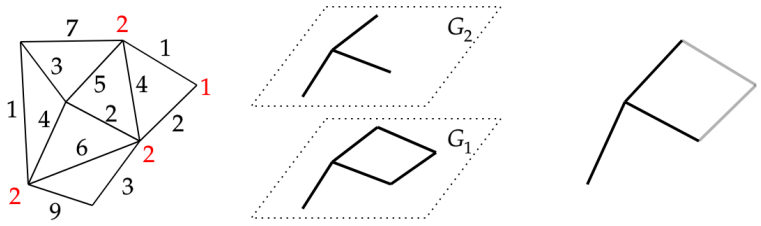


Fig. 2: *Left:* Input graph G with edge weights, $\ell = 2$, $|T_2| = 4$, $|T_1| = 3$, and $t = 3$. Required grades of service $R(v)$ are shown in red. *Center:* A valid MLGS $G_2 \subseteq G_1 \subseteq G$ is shown. *Right:* The equivalent solution, where dark edges e have $y_e = 2$ and light edges have $y_e = 1$. The cost of this solution is $2 \times (4 + 2 + 5) + 1 \times (1 + 2) = 25$.

exists a path in H from u to v using edges with a grade of service greater than or equal to i, and whose length is at most $t \cdot d_G(u, v)$. Thus, $T_\ell = \{v \in V \mid R(v) = \ell\}$, $T_{\ell-1} = \{v \in V \mid R(v) \geq \ell - 1\}$, and so on. If y_e denotes the grade of edge e (or the number of levels e appears in), then the cost of a solution is equivalently $\sum_{e \in H} c_e y_e$, that is, edges with a higher grade of service incur a greater cost. This interpretation makes it clear that more important vertices (e.g., hubs) are connected with higher quality edges; see example instance and solution in Fig. 2.

If t is arbitrarily large, the MLGS problem reduces to the *multi-level Steiner tree* (MLST) problem [1]. However it is worth noting that the problem of computing or approximating spanners is significantly harder than that of computing Steiner trees, and that a Steiner tree of G may be an arbitrarily poor spanner; a cycle on $|V|$ vertices with one edge removed is a possible Steiner tree of G, but is only a $(|V| - 1)$-spanner of G. The techniques used here have similarities to those used in the MLST problem, but more sophisticated methods are needed as well, including the use of approximate distance preservers and a new ILP formulation for the pairwise spanner problem.

1.2 Related work

Spanners and variants thereof have been studied for at least three decades, so we focus on results relating to pairwise or subsetwise spanners. Althöfer et al. [2] provide a simple greedy algorithm that constructs a multiplicative r–spanner given a graph G and a real number $r > 0$. The greedy algorithm sorts edges in E by nondecreasing weight, then for each $e = \{u, v\} \in E$, computes the shortest path $P(u, v)$ from u to v in the current spanner, and adds the edge to the spanner if the weight of $P(u, v)$ is greater than $r \cdot c_e$. The resulting subgraph H is a r–spanner for G. The main result of [2] is that, given a weighted graph G and $t \geq 1$, there is a greedy $(2t + 1)$–spanner H containing at most $n\lceil n^{1/t} \rceil$ edges, and whose weight is at most $w(MST(G))(1 + \frac{n}{2t})$ where $w(MST(G))$ denotes the weight of a minimum spanning tree of G.

Sigurd and Zachariasen [17] present an ILP formulation for the minimum-weight pairwise spanner problem (see Section 3), and show that the greedy

algorithm [2] performs well on sparse graphs of up to 64 vertices. Álvarez-Miranda and Sinnl [3] present a mixed ILP formulation for the tree t^*–spanner problem, which asks for a spanning tree of a graph G with the smallest stretch factor t^*.

Dinitz et al. [13] provide a flow-based linear programming (LP) relaxation to approximate the directed spanner problem. Their LP formulation is similar to that in [17]; however, they provide an approximation algorithm which relaxes their ILP, whereas the previous formulation was used to compute spanners to optimality. Additionally, the LP formulation applies to graphs of unit edge cost; they later take care of it in their rounding algorithm by solving a shortest path arborescence problem. They provide a $\tilde{O}(n^{\frac{2}{3}})$–approximation algorithm for the directed k–spanner problem for $k \geq 1$, which is the first sublinear approximation algorithm for arbitrary edge lengths. Bhattacharyya et al. [6] provide a slightly different formulation to approximate t–spanners as well as other variations of this problem. They provide a polynomial time $O((n \log n)^{1-\frac{1}{k}})$–approximation algorithm for the directed k–spanner problem. Berman et al. [5] provide an alternative randomized LP rounding schemes that lead to better approximation ratios. They improved the approximation ratio to $O(\sqrt{n} \log n)$ where the approximation ratio of the algorithm provided by Dinitz et al. [13] 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.

There are several results on multi-level or grade-of-service Steiner trees, e.g., [1, 4, 8, 9, 16], while multi-level spanner problems have not been studied yet.

2 Approximation algorithms for MLGS

Here, we assume an oracle subroutine that computes an optimal $(S \times S)$–spanner, given a graph G, subset $S \subseteq V$, and t. The intent is to determine if approximating MLGS is significantly harder than the subsetwise spanner problem. We formulate simple bottom-up and top-down approaches for the MLGS problem.

2.1 Oracle bottom-up approach

The approach is as follows: compute a minimum subsetwise $(T_1 \times T_1)$–spanner of G with stretch factor t. This immediately induces a feasible solution to the MLGS problem, as one can simply copy each edge from the spanner to every level (or, in terms of grades of service, assign grade ℓ to each spanner edge). We then prune edges that are not needed on higher levels. It is easy to show that the solution returned has cost no worse than ℓ times the cost of the optimal solution. Let OPT denote the cost of the optimal MLGS $G^*_\ell \subseteq G^*_{\ell-1} \subseteq \ldots \subseteq G^*_1$ for a graph G. Let MIN_i denote the cost of a minimum subsetwise $(T_i \times T_i)$–spanner for level i with stretch t, and let BOT denote the cost computed by the bottom-up approach. If no pruning is done, then $\text{BOT} = \ell\text{MIN}_1$.

Theorem 1. *The oracle bottom-up algorithm described above yields a solution that satisfies* $BOT \leq \ell \cdot OPT$.

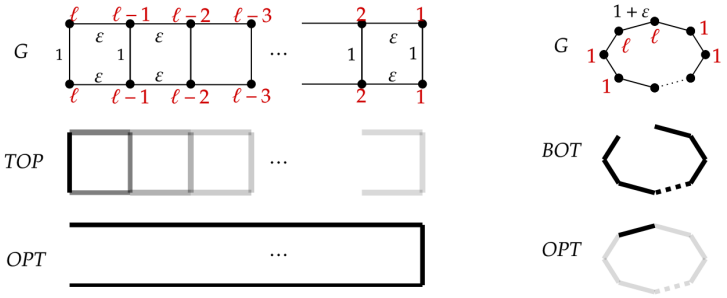


Fig. 3: *Left:* Tightness example of the top-down approach. Consider the lattice graph G with pairs of vertices of grade ℓ ($|T_\ell| = 2$), $\ell - 1$, and so on. The edge connecting the two vertices of grade i has weight 1, and all other edges have weight ε, where $0 < \varepsilon \ll 1$. Set $t = 2$. The top-down solution (middle) has cost $\text{TOP} \approx \ell + (\ell - 1) + \ldots + 1 = \frac{\ell(\ell+1)}{2}$, while the optimal solution (bottom) has cost $\text{OPT} \approx \ell$. *Right:* Tightness example of the bottom-up approach. Consider a cycle G containing two adjacent vertices of grade ℓ, and the remaining vertices of grade 1. The edge connecting the two vertices of grade ℓ is $1 + \varepsilon$, while the remaining edges have weight 1. Setting $t = |E|$ yields $\text{BOT} = \ell|E|$ while $\text{OPT} = (1 + \varepsilon)\ell + 1(|E| - 1) \approx |E| + \ell$.

Proof. We know $\text{MIN}_1 \leq \text{OPT}$, 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 asymptotically tight; an example can be constructed by letting G be a cycle containing t vertices and all edges of cost 1. Let two adjacent vertices in G appear in T_ℓ, while all vertices appear in T_1, as shown in Figure 3. As $t \to \infty$, the ratio $\frac{\text{BOT}}{\text{OPT}}$ approaches ℓ. 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 may not prune any edges, as a worst case example (Figure 3) 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 can compute a distance preserver of G_1 over terminals in T_2. One simple strategy is to use shortest paths as explained below.

Even more efficient pruning is possible through the *distant preserver* literature [7, 10]. A well-known result of distant preservers is due to the following theorem:

Theorem 2 ([10]). *Given $G = (V, E)$ with $|V| = n$, and $P \subset \binom{V}{2}$, there exists a subgraph G' with $O(n + \sqrt{n}|P|)$ edges such that for all $(u, v) \in P$ we have $d_{G'}(u, v) = d_G(u, v)$.*

The above theorem hints at a *sparse* construction of G_2 simply by letting $P = T_2 \times T_2$. 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.

2.2 Oracle top-down approach

A simple top-down heuristic that computes a solution is as follows: let G_ℓ be the minimum-cost $(T_\ell \times T_\ell)$–spanner over terminals T_ℓ with stretch factor t, and 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.

The solution returned by this approach, with cost denoted by TOP, is not worse than $\frac{\ell+1}{2}$ times the optimal. Define MIN_i and OPT as before. Define OPT_i to be the cost of edges on level i but not level $i+1$ in the optimal MLGS solution, so that $\mathrm{OPT} = \ell\mathrm{OPT}_\ell + (\ell-1)\mathrm{OPT}_{\ell-1} + \ldots + \mathrm{OPT}_1$. Define TOP_i analogously.

Theorem 3. *The oracle top-down algorithm described above yields an approximation that satisfies the following:*

(i) $TOP_\ell \leq OPT_\ell$,

(ii) $TOP_i \leq OPT_i + OPT_{i+1} + \ldots + OPT_\ell, \quad i = 1, \ldots, \ell - 1$,

(iii) $TOP \leq \frac{\ell+1}{2} OPT$.

Proof. Inequality (i) is true by definition, as we compute an optimal $(T_\ell \times T_\ell)$–spanner whose cost is TOP_ℓ, while OPT_ℓ is the cost of some $(T_\ell \times T_\ell)$-spanner. For (ii), note that $\mathrm{TOP}_i \leq \mathrm{MIN}_i$, with equality when the minimum-cost $(T_i \times T_i)$–spanner and G_{i+1} are disjoint. The spanner of cost $\mathrm{OPT}_i + \mathrm{OPT}_{i+1} + \ldots + \mathrm{OPT}_\ell$ is a feasible $(T_i \times T_i)$–spanner, so $\mathrm{MIN}_i \leq \mathrm{OPT}_i + \ldots + \mathrm{OPT}_\ell$, which shows (ii).

To show (iii), note that (i) and (ii) imply

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

as by definition $\mathrm{OPT} = \ell\mathrm{OPT}_\ell + (\ell-1)\mathrm{OPT}_{\ell-1} + \ldots + \mathrm{OPT}_1$. □

The ratio $\frac{\ell+1}{2}$ is tight as illustrated in Figure 3, left.

2.3 Combining top-down and bottom-up

Again, assume we have access to an oracle that computes a minimum weight $(S\times S)$–spanner of an input graph G with given stretch factor t. A simple combined method, similar to [1], is to run the top-down and bottom-up approaches for the MLGS problem, and take the solution with minimum cost. This has a slightly better approximation ratio than either of the two approaches.

Theorem 4. *The solution whose cost is* $\min(TOP, BOT)$ *is not worse than* $\frac{\ell+2}{3}$ *times the cost OPT of the optimal MLGS.*

The proof is given in Appendix A.

2.4 Heuristic subsetwise spanners

So far, we have assumed that we have access to an optimal subsetwise spanner given by an oracle. Here we propose a heuristic algorithm to compute subsetwise spanner. The key idea is to apply the greedy spanner to an auxiliary complete graph with terminals as its vertices and the shortest distance between terminals as edge weights. Then, we apply the distance preserver discussed in Theorem 2 to construct a subsetwise spanner.

Theorem 5. *Given graph* $G(V, E)$*, stretch factor* $t \geq 1$*, and subset* $T \subset V$*, there exists a* $(T \times T)$*–spanner for* G *with stretch factor* t *and* $O(n + \sqrt{n}|T|^{1+\frac{2}{t+1}})$ *edges.*

Proof. The spanner may be constructed as follows:

1. Construct the terminal complete graph $\bar{G}$ whose vertices are $\bar{V} := T$, such that the weight of each edge $\{u, v\}$ in $\bar{G}$ is the length of the shortest path connecting them in G, i.e., $w(u, v) = d_G(u, v)$.
2. Construct a greedy $t-$spanner $\bar{H}(\bar{V}, \bar{E}')$ of $\bar{G}$. According to [2], this graph has $|T|^{1+\frac{2}{t+1}}$ edges. Let $P = \bar{E}'$.
3. Apply Theorem 2 to obtain a subgraph H of G such that for all $(u, v) \in P$ we have $d_H(u, v) = d_G(u, v)$. Therefore, for arbitrary $u, v \in T$ we get $d_H(u, v) \leq t\, d_G(u, v)$.
4. Finally, let shortest-path(u, v) be the collection of edges in the shortest path from u to v in H, and

$$E = \bigcup_{(u,v)\in P} \{e \in E \mid e \in \text{shortest-path}(u, v).\}$$

According to Theorem 2, the number of edges in the constructed spanner $H(V, E)$ is $O(n + \sqrt{n}|P|) = O\left(n + \sqrt{n}|T|^{1+\frac{2}{t+1}}\right)$. □

Hence, we may replace the oracle in the top-down and bottom-up approaches (Sections 2.1-2.2) with the above heuristic; we call the resulting algorithms heuristic top-down and heuristic bottom-up. We analyze the performance of all algorithms on several types of graphs.

Incorporating the heuristic subsetwise spanner in our top-down and bottom up heuristics has two implications. First, the size of the final MLGS is dominated by the size of the spanner at the bottom level, i.e., $O(n + \sqrt{n}|P|) = O\left(n + \sqrt{n}|T_1|^{1+\frac{2}{t+1}}\right)$. Second, since the greedy spanner algorithm used in the above subsetwise spanner can produce spanners that are $O(n)$ more costly than the optimal solution, the same applies to the subsetwise spanner. Our experimental results, however, indicate that the heuristic approaches are very close to the optimal solutions obtained via our ILP.

3 Integer linear programming (ILP) formulations

We describe the original ILP formulation for the pairwise spanner problem [17]. Let $K = \{(u_i, v_i)\} \subset V \times V$ be the set of vertex pairs; recall that the t–spanner problem is a special case where $K = V \times V$. Here we will use unordered pairs of distinct vertices, so in the t–spanner problem we have $|K| = \binom{|V|}{2}$ instead of $|V|^2$. This ILP formulation uses *paths* as decision variables. Given $(u, v) \in K$, denote by P_{uv} the set of paths from u to v of cost no more than $t \cdot d_G(u, v)$, and denote by P the union of all such paths, i.e., $P = \bigcup_{(u,v)\in K} P_{uv}$. Given a path $p \in P$ and edge $e \in E$, let $\delta_p^e = 1$ if e is on path p, and 0 otherwise. Let $x_e = 1$ if e is an edge in the pairwise spanner H, and 0 otherwise. Given $p \in P$, let $y_p = 1$ if path p is in the spanner, and zero otherwise. An ILP formulation for the pairwise spanner problem is given below.

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

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

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

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

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

Constraint (3) ensures that for each pair $(u, v) \in K$, at least one t–spanner path is selected, and constraint (2) enforces that on the selected u-v path, every edge along the path appears in the spanner. The main drawback of this ILP is that the number of path variables is exponential in the size of the graph. The authors use delayed column generation by starting with a subset $P' \subset P$ of paths, with the starting condition that for each $(u, v) \in K$, at least one t–spanner path

in P_{uv} is in P'. The authors leave as an open question whether this problem admits a useful polynomial-size ILP.

We introduce a 0-1 ILP formulation for the pairwise t–spanner problem based on multicommodity flow, which uses $O(|E||K|)$ variables and constraints, where $|K| = O(|V|^2)$. Define t, c_e, $d_G(u,v)$, K, and x_e as before. Note that $d_G(u,v)$ can be computed in advance, using any all-pairs shortest path (APSP) method.

Direct the graph by replacing each edge $e = \{u,v\}$ with two edges (u,v) and (v,u) of weight c_e. Let E' be the set of all directed edges, i.e., $|E'| = 2|E|$. Given $(i,j) \in E'$, and an unordered pair of vertices $(u,v) \in K$, let $x^{uv}_{(i,j)} = 1$ if edge (i,j) is included in the selected u-v path in the spanner H, and 0 otherwise. This definition of path variables is similar to that by Álvarez-Miranda and Sinnl [3] for the tree t^*–spanner problem. We select a total order of all vertices so that the path constraints (8)–(9) are well-defined. This induces $2|E||K|$ binary variables, or $2|E|\binom{|V|}{2} = 2|E||V|(|V|-1)$ variables in the standard t–spanner problem. Note that if u and v are connected by multiple paths in H of length $\leq t \cdot d_G(u,v)$, we need only set $x^{uv}_{(i,j)} = 1$ for edges along some path. Given $v \in V$, let $In(v)$ and $Out(v)$ denote the set of incoming and outgoing edges for v in E'. In (7)–(11) we assume $u < v$ in the total order, so spanner paths are from u to v. An ILP formulation for the pairwise spanner problem is as follows.

$$\text{Minimize} \sum_{e \in E} c_e x_e \text{ subject to} \tag{6}$$

$$\sum_{(i,j) \in E'} x^{uv}_{(i,j)} c_e \leq t \cdot d_G(u,v) \qquad \forall (u,v) \in K; e = \{i,j\} \tag{7}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} - \sum_{(j,i) \in In(i)} x^{uv}_{(j,i)} = \begin{cases} 1 & i = u \\ -1 & i = v \\ 0 & \text{else} \end{cases} \qquad \forall (u,v) \in K; \forall i \in V \tag{8}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} \leq 1 \qquad \forall (u,v) \in K; \forall i \in V \tag{9}$$

$$x^{uv}_{(i,j)} + x^{uv}_{(j,i)} \leq x_e \qquad \forall (u,v) \in K; \forall e = \{i,j\} \in E \tag{10}$$

$$x_e, x^{uv}_{(i,j)} \in \{0,1\} \tag{11}$$

Constraint (7) requires that for all $(u,v) \in K$, the sum of the weights of the selected edges corresponding to the pair (u,v) is not more than $t \cdot d_G(u,v)$. Constraints (8)–(9) require that the selected edges corresponding to $(u,v) \in K$ form a simple path beginning at u and ending at v. Constraint (10) enforces that, if edge (i,j) or (j,i) is selected on some u-v path, then its corresponding undirected edge e is selected in the spanner; further, (i,j) and (j,i) cannot both be selected for some pair (u,v). Finally, (11) enforces that all variables are binary.

The number of variables is $|E| + 2|E||K|$ and the number of constraints is $O(|E||K|)$, where $|K| = O(|V|^2)$. Note that the variables $x^{uv}_{(i,j)}$ can be relaxed to be continuous in $[0,1]$.

3.1 ILP formulation for the MLGS problem

Recall that the MLGS problem generalizes the subsetwise spanner problem, which is a special case of the pairwise spanner problem for $K = S \times S$. Again, we use unordered pairs, i.e., $|K| = \binom{|S|}{2}$.

We generalize the ILP formulation in (6)–(11) to the MLGS problem as follows. Recall that we can encode the levels in terms of required grades of service $R : V \to \{0, 1, \ldots, \ell\}$. Instead of 0–1 indicators x_e, let y_e denote the grade of edge e in the multi-level spanner; that is, $y_e = i$ if e appears on level i but not level $i + 1$, and $y_e = 0$ if e is absent. The only difference is that for the MLGS problem, we assign grades of service to all u-v paths by assigning grades to edges along each u-v path. That is, for all $u, v \in T_1$ with $u < v$, the *selected* path from u to v has grade $\min(R(u), R(v))$, which we denote by m_{uv}. Note that we only need to require the existence of a path for terminals $u, v \in T_1$, where $u < v$. An ILP formulation for the MLGS problem is as follows.

$$\text{Minimize} \sum_{e \in E} c_e y_e \text{ subject to} \tag{12}$$

$$\sum_{(i,j) \in E'} x^{uv}_{(i,j)} c_e \leq t \cdot d_G(u, v) \qquad \forall u, v \in T_1; e = \{i, j\} \tag{13}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} - \sum_{(j,i) \in In(i)} x^{uv}_{(j,i)} = \begin{cases} 1 & i = u \\ -1 & i = v \\ 0 & \text{else} \end{cases} \qquad \forall u, v \in T_1; \forall i \in V \tag{14}$$

$$\sum_{(i,j) \in Out(i)} x^{uv}_{(i,j)} \leq 1 \qquad \forall u, v \in T_1; \forall i \in V \tag{15}$$

$$y_e \geq m_{uv} x^{uv}_{(i,j)} \qquad \forall u, v \in T_1; \forall\, e = \{i, j\} \tag{16}$$

$$y_e \geq m_{uv} x^{uv}_{(j,i)} \qquad \forall u, v \in T_1; \forall\, e = \{i, j\} \tag{17}$$

$$x^{uv}_{(i,j)} \in \{0, 1\} \tag{18}$$

Constraints (16)–(17) enforce that for each pair $u, v \in V$ such that $u < v$, the edges along the selected u-v path (not necessarily every u-v path) have a grade of service greater than or equal to the minimum grade of service needed to connect u and v, that is, m_{uv}. If multiple pairs (u_1, v_1), (u_2, v_2), $\ldots$, (u_k, v_k) use the same edge $e = \{i, j\}$ (possibly in opposite directions), then the grade of edge e should be $y_e = \max(m_{u_1 v_1}, m_{u_2 v_2}, \ldots, m_{u_k v_k})$. It is implied by (16)–(17) that $0 \leq y_e \leq \ell$ in an optimal solution.

Theorem 6. *An optimal solution to the ILP given in* (6)–(11) *yields an optimal pairwise spanner of G over a set $K \subset V \times V$.*

Theorem 7. *An optimal solution to the ILP given in* (12)–(18) *yields an optimal solution to the MLGS problem.*

We give the proofs in Appendices B and C.

3.2 Size reduction techniques

We can reduce the size of the ILP using the following shortest path tests, which works well in practice and also applies to the MLGS problem. Note that we are concerned with the total cost of a solution, not the number of edges.

If $d_G(i,j) < c(i,j)$, for some edge $\{i,j\} \in E$, then we can remove $\{i,j\}$ from the graph, as no min-weight spanner of G uses edge $\{i,j\}$. If H^* is a min-cost pairwise spanner that uses edge $\{i,j\}$, then we can replace $\{i,j\}$ with a shorter i-j path p_{ij} without violating the t–spanner requirement. In particular, if some u-v path uses both edge $\{i,j\}$ as well as some edge(s) along p_{ij}, then this path can be rerouted to use only edges in p_{ij} with smaller cost.

We reduce the number of variables needed in the single-level ILP formulation ((6)–(11)) with the following test: given $u, v \in K$ with $u < v$ and some directed edge $(i,j) \in E'$, if $d_G(u,i) + c(i,j) + d_G(j,v) > t \cdot d_G(u,v)$, then (i,j) cannot possibly be included in the selected u-v path, so set $x^{uv}_{(i,j)} = 0$. If (i,j) or (j,i) cannot be selected on any u-v path, we can safely remove $\{i,j\}$ from E.

Conversely, given some directed edge $(i,j) \in E'$, let G' be the directed graph obtained by removing (i,j) from E' (so that G' has $2|E| - 1$ edges). For each $u, v \in K$ with $u < v$, if $d_{G'}(u,v) > t \cdot d_G(u,v)$, then edge (i,j) must be in any u-v spanner path, so set $x^{uv}_{(i,j)} = 1$. For its corresponding undirected edge e, $x_e = 1$.

4 Experimental results

4.1 Setup

We use the Erdős–Rényi [14] and Watts–Strogatz [18] models to generate random graphs. Given a number of vertices, n, and probability p, the model $\mathrm{ER}(n,p)$ assigns an edge to any given pair of vertices with probability p. An instance of $\mathrm{ER}(n,p)$ with $p = (1+\varepsilon)\frac{\ln n}{n}$ is connected with high probability for $\varepsilon > 0$ [14]). For our experiments we use $n \in \{20, 40, 60, 80, 100\}$, and $\varepsilon = 1$.

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 $0 \le \beta \le 1$ while avoiding self-loops and duplicate edges. The Watts-Strogatz model generates small-world graphs with high clustering coefficients [18]. For our experiments we use $n \in \{20, 40, 60, 80, 100\}$, $K = 6$, and $\beta = 0.2$.

An instance of the MLGS problem is characterized by four parameters: graph generator, number of vertices $|V|$, number of levels ℓ, and stretch factor t. As there is randomness involved, we generated 3 instances for every choice of parameters (e.g., ER, $|V| = 80$, $\ell = 3$, $t = 2$).

We generated MLGS instances with 1, 2, or 3 levels ($\ell \in \{1, 2, 3\}$), where terminals are selected on each level by randomly sampling $\lfloor |V| \cdot (\ell - i + 1)/(\ell + 1) \rfloor$ vertices on level i so that the size of the terminal sets decreases linearly. As the terminal sets are nested, T_i can be selected by sampling from T_{i-1} (or from V if $i = 1$). We used four different stretch factors in our experiments, $t \in \{1.2, 1.4, 2, 4\}$. Edge weights are randomly selected from $\{1, 2, 3, \ldots, 10\}$.

Algorithms and outputs We implemented the bottom-up (BU) and top-down (TD) approaches from Section 2 in Python 3.5, as well as the combined approach that selects the better of the two (Section 2.3). To evaluate the approximation algorithms and the heuristics, we implemented the ILPs described in Section 3 using CPLEX 12.6.2. We used the same high-performance computer for all experiments (Lenovo NeXtScale nx360 M5 system with 400 nodes).

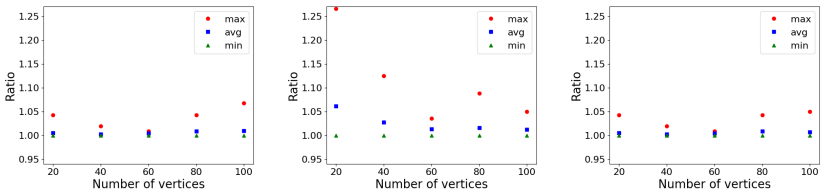
For each instance of the MLGS problem, we compute the costs of the MLGS returned using the bottom-up (BU), the top-down (TD), and the combined (min(BU, TD)) approaches, as well as the minimum cost MLGS using the ILP in Section 3.1. The three heuristics involve a (single-level) subroutine; we used both the heuristic described in Section 2.4, as well as the flow formulation described in Section 3 which computes subsetwise spanners to optimality. We compare the algorithms with and without the oracle to assess whether computing (single-level) spanners to optimality significantly improves the overall quality of the solution.

We show the performance ratio for each heuristic in the y-axis (defined as the heuristic cost divided by OPT), and how the ratio depends on the input parameters (number of vertices $|V|$, number of levels ℓ, and stretch factors t). Finally, we discuss the running time of the ILP. All box plots show the minimum, interquartile range and maximum, aggregated over all instances using the parameter being compared.

4.2 Results

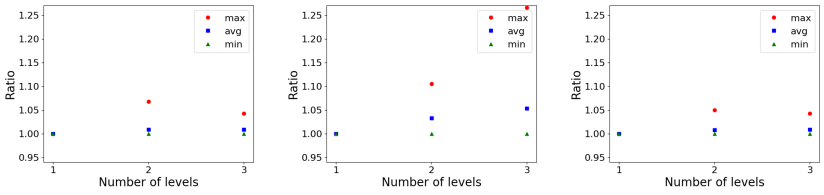
We first discuss the results for Erdős–Rényi graphs. Figures 4–7 show the results of the oracle top-down, bottom-up, and combined approaches. We show the impact of different parameters (number of vertices $|V|$, number of levels ℓ, and stretch factors t) using line plots for three heuristics separately in Figures 4-6. Figure 7 shows the performance of the three heuristics together in box plots. In Figure 4 we can see that the bottom-up heuristic performs slightly worse for increasing $|V|$, while the top-down heuristic performs slightly better. In Figure 5 we see that the heuristics perform worse when ℓ increases, consistent with the ratios discussed in Section 2. In Figure 6 we show the performance of the heuristics with respect to the stretch factor t. In general, the performance becomes worse as t increases.

The most time consuming part of the experiment is the execution time of the ILP for solving MLGS instances optimally. The running time of the heuristics is significantly smaller compared to that of the ILP. Hence, we first show the running times of the exact solution of the MLGS instances in Figure 8. We show the running time with respect to the number of vertices $|V|$, number of levels ℓ, and stretch factors t. For all parameters, the running time tends to increase as the size of the parameter increases. In particular, the running time with stretch factor 4 (Fig. 8, right) was much worse, as there are many more t-spanner paths to consider, and the size reduction techniques in Section 3.2 are less effective at reducing instance size. We show the running times of for computing oracle bottom-up, top-down and combined solutions in Figure 9.



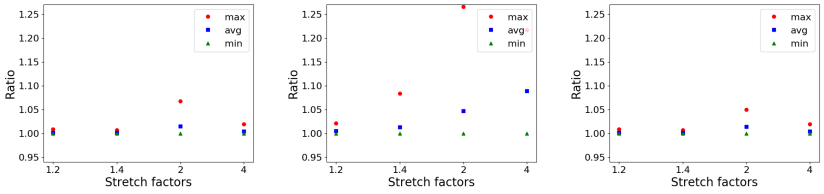
(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 4: Performance with oracle on Erdős–Rényi graphs w.r.t. $|V|$. Ratio is defined as the cost of the returned MLGS divided by OPT.



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 5: Performance with oracle on Erdős–Rényi graphs w.r.t. the number of levels



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 6: Performance with oracle on Erdős–Rényi graphs w.r.t. stretch factor

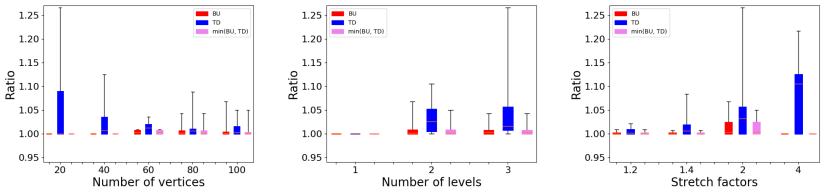


Fig. 7: Performance with oracle on Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

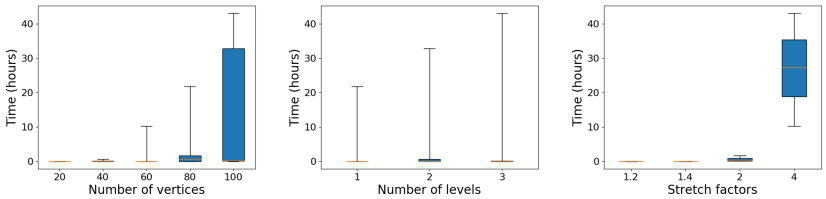


Fig. 8: Experimental running times for computing exact solutions on Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

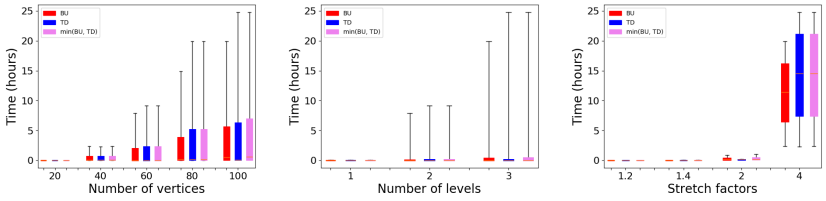
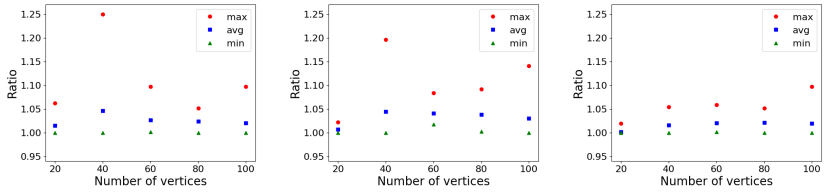


Fig. 9: Experimental running times for computing oracle bottom-up, top-down and combined solutions on Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

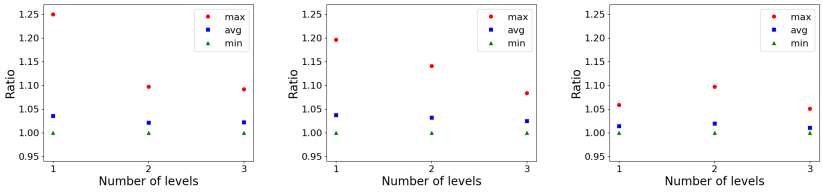
The ILP is too computationally expensive for larger input sizes and this is where the heurstic can be particularly useful. We now consider a similar experiment using the heuristic to compute subsetwise spanners, as described in Section 2.4. We show the impact of different parameters (number of vertices $|V|$, number of levels ℓ, and stretch factors t) using scatter plots for three heuristics separately in Figures 10–12. Figure 13 shows the performance of the three heuristics together in box plots. We can see that the heuristics perform very well in practice. Notably when the heuristic is used in place of the ILP (Fig 24), the running times decrease for larger stretch factors.



(a) Bottom up (b) Top down (c) min(BU, TD)

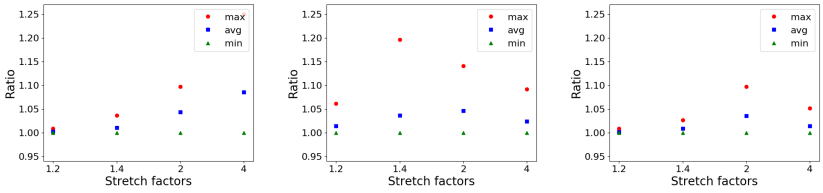
Fig. 10: Performance without oracle on Erdős–Rényi graphs w.r.t. $|V|$

We also analyzed graphs generated from the Watts–Strogatz model and the results are shown in Appendix D.



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 11: Performance without oracle on Erdős–Rényi graphs w.r.t. the number of levels



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 12: Performance without oracle on Erdős–Rényi graphs w.r.t. the stretch factors

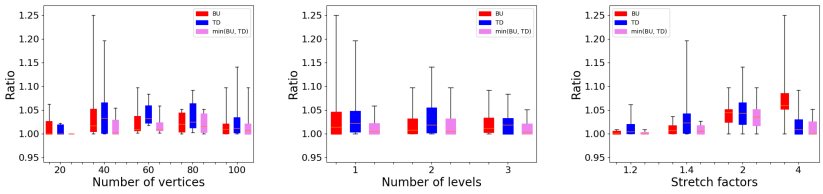


Fig. 13: Performance without oracle on Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

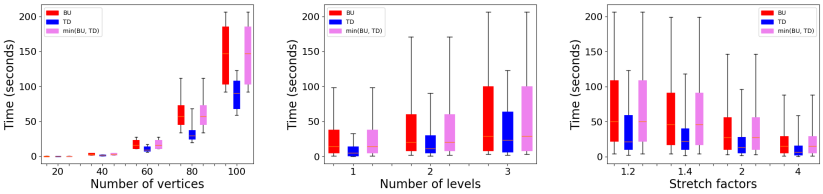


Fig. 14: Experimental running times for computing heuristic bottom-up, top-down and combined solutions on Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

Our final experiments test the heuristic performance on a set of larger graphs. We generated the graphs using the Erdős–Rényi model, with $|V| \in \{100, 200, 300, 400\}$. We evaluated more levels ($\ell \in \{2, 4, 6, 8, 10\}$) with stretch factors $t \in \{1.2, 1.4, 2, 4\}$. We show the performance of heuristic bottom-up and top-down in Appendix E. Here, the ratio is determined by dividing the BU or TD cost by $\min(BU, TD)$ (as computing the optimal MLGS would be too time-consuming). The results indicate that while running times increase with larger input graphs, the number of levels and the stretch factors seem to have little impact on performance.

5 Discussion and conclusion

We introduced a generalization of the subsetwise spanner problem to multiple levels or grades of service. Our proposed ILP formulation requires only a polynomial size of variables and constraints, which is an improvement over the previous formulation given by Sigurd and Zachariasen [17]. We also proposed improved formulations which work well for small values of the stretch factor t. It would be worthwhile to consider whether even better ILP formulations can be found for computing graph spanners and their multi-level variants. We showed that both the approximation algorithms and the heuristics work well in practice on several different types of graphs, with different number of levels and different stretch factors.

We only considered a stretch factor t that is the same for all levels in the multi-level spanner, as well as a fairly specific definition of cost. It would be interesting to investigate more general multi-level or grade-of-service spanner problems, including ones with varying stretch factors (e.g., in which more important terminals require a smaller or larger stretch factors), different definitions of cost, and spanners with other requirements, such as bounded diameters or degrees.

References

1. Ahmed, A.R., Angelini, P., Sahneh, F.D., Efrat, A., Glickenstein, D., Gronemann, M., Heinsohn, N., Kobourov, S., Spence, R., Watkins, J., Wolff, A.: Multi-level Steiner trees. In: 17th International Symposium on Experimental Algorithms, (SEA). pp. 15:1–15:14 (2018). https://doi.org/10.4230/LIPIcs.SEA.2018.15, `https://doi.org/10.4230/LIPIcs.SEA.2018.15`
2. Althöfer, I., Das, G., Dobkin, D., Joseph, D.: Generating sparse spanners for weighted graphs. In: Gilbert, J.R., Karlsson, R. (eds.) SWAT 90. pp. 26–37. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
3. Álvarez-Miranda, E., Sinnl, M.: Mixed-integer programming approaches for the tree t*-spanner problem. Optimization Letters (Oct 2018). https://doi.org/10.1007/s11590-018-1340-0, `https://doi.org/10.1007/s11590-018-1340-0`
4. Balakrishnan, A., Magnanti, T.L., Mirchandani, P.: Modeling and heuristic worst-case performance analysis of the two-level network design problem. Management Sci. **40**(7), 846–867 (1994). https://doi.org/10.1287/mnsc.40.7.846

5. Berman, P., Bhattacharyya, A., Makarychev, K., Raskhodnikova, S., Yaroslavtsev, G.: Approximation algorithms for spanner problems and directed steiner forest. Information and Computation **222**, 93 – 107 (2013). https://doi.org/https://doi.org/10.1016/j.ic.2012.10.007, http://www.sciencedirect.com/science/article/pii/S0890540112001484, 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)
6. Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Transitive-closure spanners. SIAM Journal on Computing **41**(6), 1380–1425 (2012). https://doi.org/10.1137/110826655, https://doi.org/10.1137/110826655
7. Bodwin, G.: Linear size distance preservers. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 600–615. Society for Industrial and Applied Mathematics (2017)
8. Charikar, M., Naor, J.S., Schieber, B.: Resource optimization in QoS multicast routing of real-time multimedia. IEEE/ACM Trans. Networking **12**(2), 340–348 (2004). https://doi.org/10.1109/TNET.2004.826288
9. Chuzhoy, J., Gupta, A., Naor, J.S., Sinha, A.: On the approximability of some network design problems. ACM Trans. Algorithms **4**(2), 23:1–23:17 (2008). https://doi.org/10.1145/1361192.1361200
10. Coppersmith, D., Elkin, M.: Sparse sourcewise and pairwise distance preservers. SIAM Journal on Discrete Mathematics **20**(2), 463–501 (2006)
11. Cygan, M., Grandoni, F., Kavitha, T.: On pairwise spanners. In: Portier, N., Wilke, T. (eds.) 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013). Leibniz International Proceedings in Informatics (LIPIcs), vol. 20, pp. 209–220. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013). https://doi.org/10.4230/LIPIcs.STACS.2013.209, http://drops.dagstuhl.de/opus/volltexte/2013/3935
12. David, P., A., S.A.: Graph spanners. Journal of Graph Theory **13**(1), 99–116 (1989). https://doi.org/10.1002/jgt.3190130114, https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.3190130114
13. Dinitz, M., Krauthgamer, R.: Directed spanners via flow-based linear programs. In: Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing. pp. 323–332. STOC '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/1993636.1993680, http://doi.acm.org/10.1145/1993636.1993680
14. Erdős, P., Rényi, A.: On random graphs, i. Publicationes Mathematicae (Debrecen) **6**, 290–297 (1959)
15. Kortsarz, G.: On the hardness of approximating spanners. Algorithmica **30**(3), 432–450 (Jan 2001). https://doi.org/10.1007/s00453-001-0021-y, https://doi.org/10.1007/s00453-001-0021-y
16. Mirchandani, P.: The multi-tier tree problem. INFORMS J. Comput. **8**(3), 202–218 (1996)
17. Sigurd, M., Zachariasen, M.: Construction of minimum-weight spanners. In: Albers, S., Radzik, T. (eds.) Algorithms – ESA 2004. pp. 797–808. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
18. Watts, D.J., Strogatz, S.H.: Collective dynamics of small-worldnetworks. Nature **393**(6684), 440 (1998)

A Proof of Theorem 4

Proof. We use the simple algebraic fact that $\min\{x, y\} \leq \alpha x + (1-\alpha)y$ for all $x, y \in \mathbb{R}$ and $\alpha \in [0, 1]$. Here, we can also use the fact that $\text{MIN}_1 \leq \text{OPT}_1 + \text{OPT}_2 + \ldots + \text{OPT}_\ell$, as the RHS equals the cost of G_1^*, which is some subsetwise $(T_1 \times T_1)$-spanner. Combining, we have

$$\begin{aligned}\min(\text{TOP}, \text{BOT}) &\leq \alpha \sum_{i=1}^{\ell} \frac{i(i+1)}{2} \text{OPT}_i + (1-\alpha)\ell \sum_{i=1}^{\ell} \text{OPT}_i \\ &= \sum_{i=1}^{\ell} \left[\left(\frac{i(i+1)}{2} - \ell \right) \alpha + \ell \right] \rho \text{OPT}_i\end{aligned}$$

Since we are comparing $\min\{\text{TOP}, \text{BOT}\}$ to $r \cdot \text{OPT}$ for some approximation ratio $r > 1$, we can compare coefficients and find the smallest $r \geq 1$ such that the system of inequalities

$$\begin{aligned}\left(\frac{\ell(\ell+1)}{2} - \ell \right) \alpha + \ell\rho &\leq \ell r \\ \left(\frac{(\ell-1)\ell}{2} - \ell \right) \alpha + \ell\rho &\leq (\ell-1) r \\ &\vdots \\ \left(\frac{2 \cdot 1}{2} - \ell \right) \alpha + \ell\rho &\leq r\end{aligned}$$

has a solution $\alpha \in [0, 1]$. Adding the first inequality to $\ell/2$ times the last inequality yields $\frac{\ell^2+2\ell}{2} \leq \frac{3\ell r}{2}$, or $r \geq \frac{\ell+2}{3}$. Also, it can be shown algebraically that $(r, \alpha) = (\frac{\ell+2}{3}, \frac{2}{3})$ simultaneously satisfies the above inequalities. This implies that $\min\{\text{TOP}, \text{BOT}\} \leq \frac{\ell+2}{3} \rho \cdot \text{OPT}$. □

B Proof of Theorem 6

Proof. Let H^* denote an optimal pairwise spanner of G with stretch factor t, and let OPT denote the cost of H^*. Let OPT_{ILP} denote the minimum cost of the objective in the ILP (6). First, given a minimum cost t–spanner $H^*(V, E^*)$, a solution to the ILP can be constructed as follows: for each edge $e \in E^*$, set $x_e = 1$. Then for each unordered pair $(u, v) \in K$ with $u < v$, compute a shortest path p_{uv} from u to v in H^*, and set $x_{(i,j)}^{uv} = 1$ for each edge along this path, and $x_{(i,j)}^{uv} = 0$ if (i, j) is not on p_{uv}.

As each shortest path p_{uv} necessarily has cost $\leq t \cdot d_G(u, v)$, constraint (7) is satisfied. Constraints (8)–(9) are satisfied as p_{uv} is a simple u-v path. Constraint (10) also holds, as p_{uv} should not traverse the same edge twice in opposite directions. In particular, every edge in H^* appears on some shortest

path; otherwise, removing such an edge yields a pairwise spanner of lower cost. Hence $\mathrm{OPT}_{ILP} \leq \mathrm{OPT}$.

Conversely, an optimal solution to the ILP induces a feasible t–spanner H. Consider an unordered pair $(u, v) \in K$ with $u < v$, and the set of decision variables satisfying $x^{uv}_{(i,j)} = 1$. By (8) and (9), these chosen edges form a simple path from u to v. The sum of the weights of these edges is at most $t \cdot d_G(u, v)$ by (7). Then by constraint (10), the chosen edges corresponding to (u, v) appear in the spanner, which is induced by the set of edges e with $x_e = 1$. Hence $\mathrm{OPT} \leq \mathrm{OPT}_{ILP}$.

Combining the above observations, we see that $\mathrm{OPT} = \mathrm{OPT}_{ILP}$. □

C Proof of Theorem 7

Proof. Given an optimal solution to the ILP with cost OPT_{ILP}, construct an MLGS by letting $G_i = (V, E_i)$ where $E_i = \{e \in E \mid y_e \geq i\}$. This clearly gives a nested sequence of subgraphs. Let u and v be terminals in T_i (not necessarily of required grade $R(\cdot) = i$), with $u < v$, and consider the set of all variables of the form $x^{uv}_{(i,j)}$ equal to 1. By (13)–(15), these selected edges form a path from u to v of length at most $t \cdot d_G(u, v)$, while constraints (16)–(17) imply that these selected edges have grade at least $m_{uv} \geq i$, so the selected path is contained in E_i. Hence G_i is a subsetwise $(T_i \times T_i)$–spanner for G with stretch factor t, and the optimal ILP solution gives a feasible MLGS.

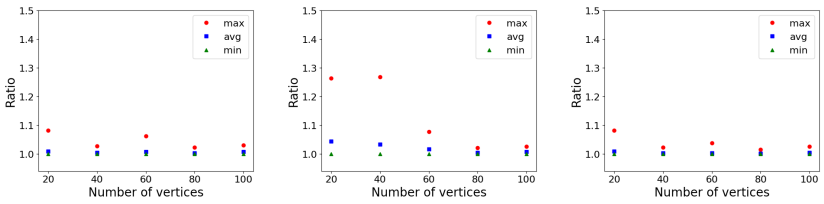
Given an optimal MLGS with cost OPT, we can construct a feasible ILP solution with the same cost in a way similar to the proof of Theorem 6. For each $u, v \in T_1$ with $u < v$, set $m_{uv} = \min(R(u), R(v))$. Compute a shortest path in $G_{m_{uv}}$ from u to v, and set $x^{uv}_{(i,j)} = 1$ for all edges along this path. Then for each $e \in E$, consider all pairs $(u_1, v_1), \ldots, (u_k, v_k)$ that use either (i, j) or (j, i), and set $y_e = \max(m_{u_1 v_1}, m_{u_2 v_2}, \ldots, m_{u_k v_k})$. In particular, y_e is not larger than the grade of e in the MLGS, otherwise this would imply e is on some u-v path at grade greater than its grade of service in the actual solution. □

D Experimental results on graphs generated using Watts-Strogatz

The results for graphs generated from the Watts–Strogatz model are shown in Figures 15–23, which are organized in the same way as for Erdős–Rényi.

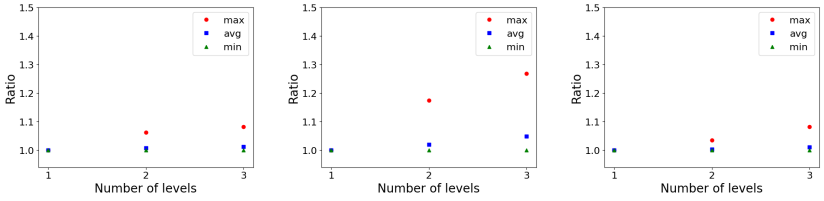
E Experimental results on large graphs using Erdős-Rényi

Figure 24 shows a rough measure of performance for the bottom-up and top-down heuristics on large graphs using the Erdős-Rényi model, where the ratio is defined as the BU or TD cost divided by min(BU, TD). Figure 25 shows the aggregated running times per instance, which significantly worsen as $|V|$ is large.



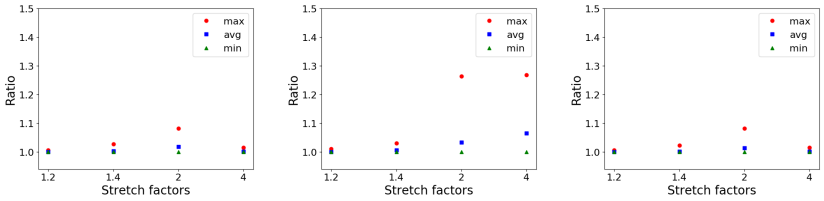
(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 15: Performance with oracle on Watts–Strogatz graphs w.r.t. the number of vertices



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 16: Performance with oracle on Watts–Strogatz graphs w.r.t. the number of levels



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 17: Performance with oracle on Watts–Strogatz graphs w.r.t. the stretch factors

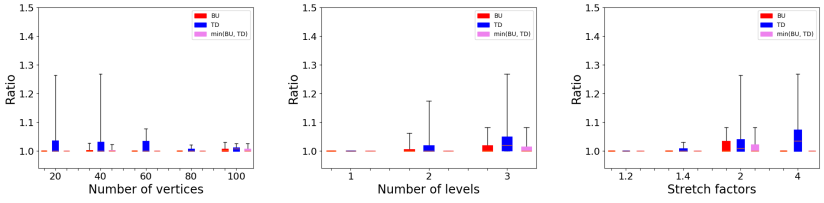


Fig. 18: Performance with oracle on Watts–Strogatz graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

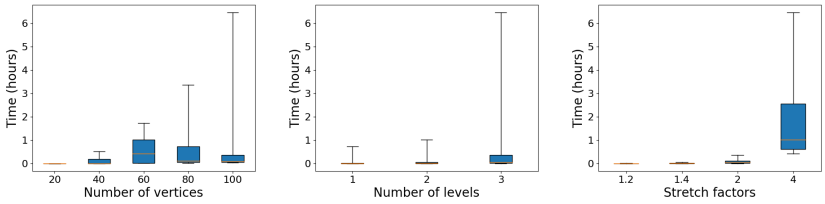
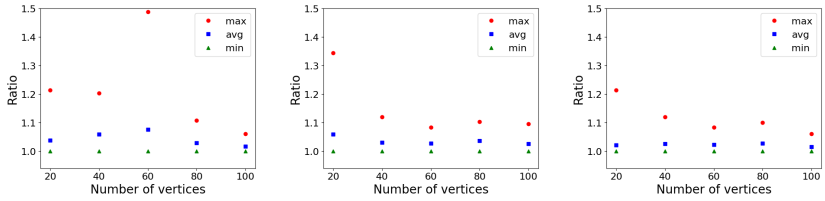
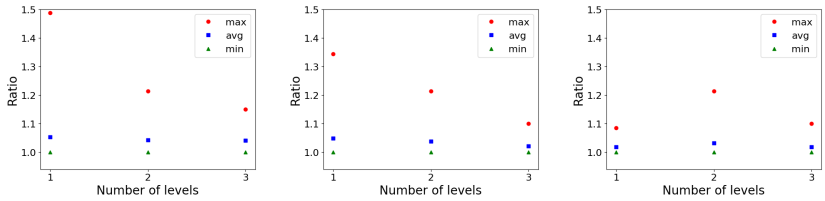


Fig. 19: Experimental running times for computing exact solutions on Watts–Strogatz graphs w.r.t. the number of vertices, the number of levels, and the stretch factors



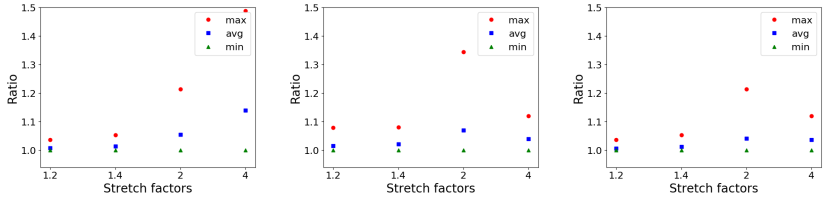
(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 20: Performance without oracle on Watts–Strogatz graphs w.r.t. the number of vertices



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 21: Performance without oracle on Watts–Strogatz graphs w.r.t. the number of levels



(a) Bottom up (b) Top down (c) min(BU, TD)

Fig. 22: Performance without oracle on Watts–Strogatz graphs w.r.t. the stretch factors

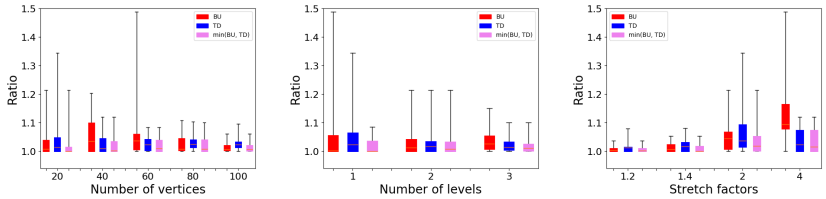


Fig. 23: Performance without oracle on Watts–Strogatz graphs w.r.t. the number of vertices, the number of levels, and the stretch factors

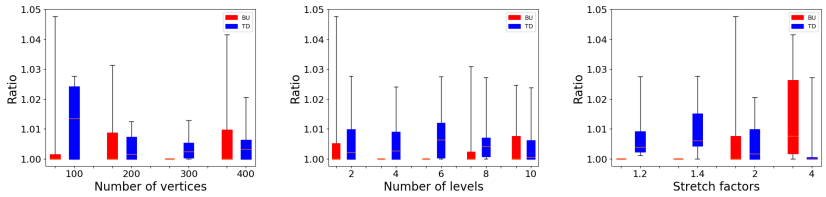


Fig. 24: Performance of heuristic bottom-up and top-down on large Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors. The ratio is determined by dividing the objective value of the combined (min(BU, TD)) heuristic.

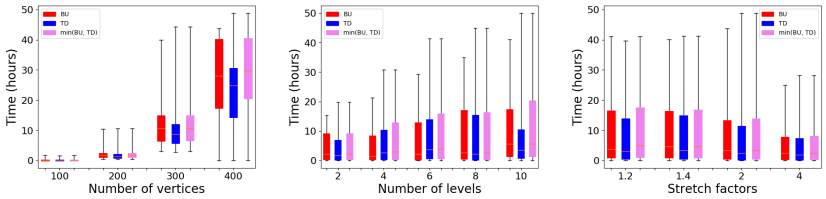


Fig. 25: Experimental running times for computing heuristic bottom-up, top-down and combined solutions on large Erdős–Rényi graphs w.r.t. the number of vertices, the number of levels, and the stretch factors.