Approximate Light Spanners in Planar Graphs

Hung Le* Shay Solomon† Cuong Than‡ Csaba D. Tóth§ Tianyi Zhang¶

Abstract

In their seminal paper, Althöfer et al. (DCG 1993) introduced the *greedy spanner* and showed that, for any weighted planar graph G, the weight of the greedy $(1+\epsilon)$-spanner is at most $(1+\frac{2}{\epsilon})\cdot \mathbf{w}(\mathsf{MST}(G))$, where $\mathbf{w}(\mathsf{MST}(G))$ is the weight of a minimum spanning tree $\mathsf{MST}(G)$ of G. This bound is optimal in an *existential sense*: there exist planar graphs G for which any $(1+\epsilon)$-spanner has a weight of at least $(1+\frac{2}{\epsilon})\cdot \mathbf{w}(\mathsf{MST}(G))$.

However, as an *approximation algorithm*, even for a *bicriteria* approximation, the weight approximation factor of the greedy spanner is essentially as large as the existential bound: There exist planar graphs G for which the greedy $(1+x\epsilon)$-spanner (for any $1 \le x = O(\epsilon^{-1/2})$) has a weight of $\Omega\left(\frac{1}{\epsilon\cdot x^2}\right)\cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$, where $G_{\mathsf{opt},\epsilon}$ is a $(1+\epsilon)$-spanner of G of minimum weight.

Despite the flurry of works over the past three decades on approximation algorithms for spanners as well as on light(-weight) spanners, there is still no (possibly bicriteria) approximation algorithm for light spanners in weighted planar graphs that outperforms the existential bound. As our main contribution, we present a polynomial time algorithm for constructing, in any weighted planar graph G, a $\left(1+\epsilon\cdot 2^{O(\log^* 1/\epsilon)}\right)$-spanner for G of total weight $O(1)\cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$.

To achieve this result, we develop a new technique, which we refer to as *iterative planar pruning*. It iteratively modifies a spanner; each iteration replaces a heavy set of edges by a light path, to substantially decrease the total weight of the spanner while only slightly increasing its stretch. We leverage planarity to prove a *laminar* structural property of the edge set to be removed, which enables us to optimize the path to be inserted via dynamic programming. Our technique applies dynamic programming *directly to the input planar graph*, which significantly deviates from previous techniques used for network design problems in planar graphs, and might be of independent interest.

*University of Massachusetts Amherst, hungle@cs.umass.edu
†Tel Aviv University, solo.shay@gmail.com
‡University of Massachusetts Amherst, cthan@umass.edu
§California State University Northridge and Tufts University, csaba.toth@csun.edu
¶Nanjing University, tianyiz25@nju.edu.cn

Contents

1 Introduction

A *t-spanner* of an edge-weighted undirected graph $G = (V, E, \mathbf{w})$ is a subgraph H of G such that $\mathsf{dist}_H(u, v) \leq t \cdot \mathsf{dist}_G(u, v)$ for every pair u, v of points in V, where dist_G denotes the shortest path distance in G w.r.t the weight function $\mathbf{w}$. The study of graph spanners was pioneered in the late 1980s by influential results on spanners for general graphs [PU89, PS89] and in low-dimensional spaces [Che86, Cla87, Kei88], and it has grown into a very active and vibrant research area; see the survey [ABS+20]. A central research direction within the area of spanners is the design of approximation algorithms for an *optimal* t-spanner of a given graph, where the optimality usually refers to the two most common "compactness" measures: the spanner *size* (number of edges) or *weight* (total edge weight).[1] The *greedy* algorithm by Althöfer et al. [ADD+93] gives a t-spanner of size $O(n^{1+2/(t+1)})$ for odd t and $O(n^{1+2/t})$ for even t. Therefore, the approximation ratio of the greedy algorithm (for minimizing the number of edges) is naively bounded by $O(n^{2/(t+1)})$ for odd t and by $O(n^{2/t})$ for even t. The recent line of work on light spanners [CDNS95, CW18, FN22, LS22a, LS23] implies almost the same approximation ratio for minimizing the weight of the spanners.

Beating the approximation ratio of the greedy algorithm has been extremely challenging despite years of effort. Better approximation factors for sparsity are known only for $t \in \{2, 3, 4\}$; specifically $O(\log n)$ [KP94] for $t = 2$ (matching a known lower bound [Kor01]) and $\tilde{O}(n^{1/3})$ for $t = 3$ [BBM+11] and $t = 4$ [DZ16]. The gap between the upper and lower bounds is fairly large: for any $t \geq 3$ and for any constant $\epsilon > 0$, there is no polynomial-time algorithm approximating a t-spanner with ratio better than $2^{\log n^{1-\epsilon}/t}$ assuming $NP \not\subseteq BPTIME(2^{\text{polylog}(n)})$ [DKR16]. (For directed graphs,[2] the current best approximation ratios are even worse [BBM+11, BGJ+12, BRR10, DK11].) Some of the aforementioned results extend to minimizing the weight of the spanner and produce the same (or sometimes worse) approximation ratio [BBM+11, GKL23]. Efforts have also been made to experimentally test LP-based algorithms to find the (exact) minimum weight spanner of general graphs [SZ04, AHJ+19, BCJW24].

For general graphs, perhaps the most interesting stretch regime is $t \geq 3$ since, in this regime, t-spanners have a subquadratic size for any input graph. Even in this regime, as mentioned above, poly-logarithmic approximation algorithms are impossible under a standard complexity assumption. In various real-life applications, even stretch $t = 3$ is too large. It is thus only natural to focus on structural classes of graphs, where we can hope to achieve (i) stretch $t = 1 + \epsilon$ for any given $\epsilon > 0$ and (ii) a constant-factor approximation (with a constant that does not depend on ϵ). Perhaps the two most basic and well-studied structural classes of graphs are *low-dimensional Euclidean spaces* and *planar graphs*.

In low-dimensional spaces, we are given a set P of n points in the Euclidean space $\mathbb{R}^d$ of any constant dimension d; the metric graph $G_P = (P, E, \mathbf{w})$ induced by P is the edge-weighted complete graph with vertex set P and edge weights $\mathbf{w}(u, v) = \|u - v\|_2$ for all $u, v \in P$. Euclidean spanners have been thoroughly investigated; the book by Smid and Narasimhan [NS07] covers dozens of techniques for constructing Euclidean $(1 + \epsilon)$-spanners, and a plethora of additional techniques have been devised since then [Smi25]. In particular, one can construct $(1 + \epsilon)$-spanners with $O(\epsilon^{1-d}n)$ size [RS91, LS22b] and $O(\epsilon^{-d}\mathbf{w}(\mathsf{MST}(G_P)))$ weight [DNS95, LS22b], where $\mathsf{MST}(G_P)$ is the weight of the Euclidean minimum spanning tree for P. These results imply an $O(\epsilon^{1-d})$-approximation for the minimum size and $O(\epsilon^{-d})$ for the minimum weight—of the optimal $(1 + \epsilon)$-spanner. Finding a minimum weight $(1 + \epsilon)$-spanner is still NP-hard in this setting [CC13]. A major open problem is to obtain an $O(1)$-approximation (w.r.t. the size or weight) for the optimal

[1] There are also other compactness measures, such as the *maximum degree*, but the most common and well-studied measures are the size and weight of spanners and for brevity we will restrict attention to them.

[2] In the literature survey that follows we restrict attention to undirected graphs, which is the focus of this work.

Euclidean $(1+\epsilon)$-spanner, where the *approximation factor does not depend on* ϵ. Despite a vast literature on Euclidean spanners, there has been no progress on this problem until the recent work of Le et al. [LST+24], which gives the first *bi-criteria approximation algorithm*. Specifically, they constructed $(1+\epsilon \cdot 2^{O(\log^*(d/\epsilon))})$-spanners whose size and weight are within $O(1)$ of those of the optimal $(1+\epsilon)$-spanners for any $\epsilon > 0$.

For planar graphs, the literature on spanners is much sparser. Basic geometric techniques and concepts do not apply and cannot even be adapted to planar graphs. One exception is with the aforementioned *greedy* algorithm by Althöfer et al. [ADD+93], which applies to any graph, as it is oblivious to the input structure. They showed that the greedy $(1+\epsilon)$-spanner of any edge-weighted planar graph G has weight at most $(1+\frac{2}{\epsilon})\mathbf{w}(\mathsf{MST}(G))$. (Since planar graphs have only $O(n)$ edges, the focus has been mainly on optimizing the weight.) Klein [Kle05] gave a more relaxed variant of the greedy algorithm, with the same asymptotic weight bound, which can be implemented in $O(n)$ time. Both the greedy algorithm and Klein's variant imply an $O(1/\epsilon)$-approximation for the minimum weight $(1+\epsilon)$-spanner for any input planar graph. The following fundamental question remains open.

Question 1.1. Can one get a polynomial-time algorithm that computes a $(1+\epsilon)$-spanner of weight $O(1) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$, where $G_{\mathsf{opt},\epsilon}$ is a minimum-weight $(1+\epsilon)$-spanner of any input planar graph G?

A wide variety of techniques have been designed for approximation algorithms in planar graphs over the years [Bak94, Kle05, BKM09, BHM11, BDHM16, BCE+11, FKS19, FL22], but none of them appears to be applicable to Question 1.1. Compared to many network design problems, such as *TSP* or *Steiner tree*, the key challenge in approximating the minimum weight $(1+\epsilon)$-spanner is that we have to achieve two guarantees: (i) stretch of $1+\epsilon$ (i.e., preserving all pairwise distances up to a factor of $1+\epsilon$) and (ii) minimizing the weight (approximating the minimum weight up to a given factor). Baker's technique [Bak94] is perhaps the most basic one: compute a BFS tree of G, divide the input graph into subgraphs consisting of $O(1)$ layers of the BFS tree, each subgraph then has treewidth $O(1)$, and solve the problem in each subgraph, say H, by applying dynamic programming on bounded treewidth graphs. If one applies Baker's technique to spanners, the issue is that the distance between vertices in each subgraph H (of bounded treewidth) is not the same as the distance in the input graph, and hence the optimal solution for H could be much heavier than $E(G_{\mathsf{opt},\epsilon}) \cap E(H)$. (When the graph is unweighted and the stretch is constant, one can extend H by $O(t)$ more layers to preserve the distances between vertices in H [DFG11]; when the graph is weighted, such a trick does not apply.) Other variants of Baker's technique, such as the contraction decomposition [Kle05], run into a similar issue.

The two different guarantees in approximating spanners are reminiscent of other problems with distance constraints in planar graphs. One such problem is the ρ-dominating set problem: given an input (non-constant) parameter ρ, find a minimum set of vertices such that other vertices must be within distance at most ρ from the set. For these problems, there is an effective technique for designing bicriteria approximation algorithms by embedding the graph into a small treewidth graph [FKS19, FL22, CCL+23]. Specifically, given a planar graph G with diameter Δ, one can embed G into a graph H with treewdith $O(1/\epsilon^4)$, such that $\mathsf{dist}_G(u,v) \le \mathsf{dist}_H(u,v) \le \mathsf{dist}_G(u,v) + \epsilon\Delta$ for any $\epsilon \in (0,1)$ [CCL+23]. However, the additive distortion $+\epsilon\Delta$ could be very large compared to $\mathsf{dist}_G(u,v)$; for example, in unweighted graph we have $\mathsf{dist}_G(u,v) = 1$ for every edge (u,v), while the additive distortion guarantee is $\epsilon\Delta$, which could be as large as $\Omega(\epsilon n)$. Therefore, such an embedding technique also does not provide a good bicriteria approximation algorithm for $(1+\epsilon)$-spanners.

Another natural attempt to obtain a bicriteria approximation algorithm is to apply the greedy algorithm with stretch $(1+x\cdot\epsilon)$, for some parameter x, and compare its weight to the minimum

weight of a $(1+\epsilon)$-spanner. In Section 6, we present a hard planar graph instance for which the greedy $(1+x\cdot\epsilon)$-spanner has weight as large as $\Omega\left(\frac{1}{x^2\epsilon}\right)\mathbf{w}(G_{\mathsf{opt},\epsilon})$; this instance is obtained by a careful adaptation of a hard Euclidean instance from [LST+24]. Therefore, the greedy algorithm does not yield an $O(1)$-approximation for the weight even when x is rather large, say $x\approx\epsilon^{-1/3}$.

1.1 Our Main Contribution: Approximate Light Spanners

In this paper, we make significant progress toward the resolution of Question 1.1 by designing a bicriteria approximation algorithm for $(1+\epsilon)$-spanners in planar graphs: The stretch is $1+\epsilon\cdot 2^{O(\log^*(1/\epsilon))}$ while the approximation ratio is $O(1)$; here $\log^*$ is iterated logarithm.

Theorem 1.2. *Given any edge-weighted planar graph $G=(V,E,\mathbf{w})$ with integral edge weights $\mathbf{w}: E\to\mathbb{N}_+$ as well as a parameter $\epsilon>0$, a $\mathrm{poly}(n,1/\epsilon)$-time algorithm can construct a $\left(1+\epsilon\cdot 2^{O(\log^* 1/\epsilon)}\right)$-spanner $H\subseteq G$ of total weight $O(1)\cdot\mathbf{w}(G_{\mathsf{opt},\epsilon})$, where $G_{\mathsf{opt},\epsilon}=(V,E_{\mathsf{opt},\epsilon})$ denotes a $(1+\epsilon)$-spanner of G of minimum total weight.*

To obtain our result, we develop a new technique, which we refer to as *iterative planar pruning*. While our technique is inspired by the recent biciteria approximation algorithm for Euclidean $(1+\epsilon)$-spanner [LST+24], it has to deviate significantly from it, since the algorithm of [LST+24] crucially relies on several basic properties of Euclidean geometry that do not hold in planar graphs. In Section 2 we provide a detailed overview of our technique, and its comparison with previous work and the technique of [LST+24] in particular. At a very high-level, our technique will heavily exploit the *planarity* of the input graph to establish a certain *laminar structural property* of carefully chosen paths in the current spanner. This laminar structure property is key for obtaining an efficient *dynamic programming* algorithm, which is, in turn, used for iteratively finding a light path to replace a set of much heavier edges.

Perhaps our most significant departure from existing techniques for network design problems in planar graphs, including TSP [Kle05], Steiner tree [BKM09], Steiner forest [BHM11], and their prize-collecting counterparts [BCE+11], to name a few, is the usage of dynamic programming. Existing techniques could be seen as providing a reduction to bounded treewidth graphs, where dynamic programming naturally arises[3]. In contrast to previous work, the dynamic programming in our work *applies directly to the input planar graph*, whose treewidth may be as large as $\Theta(\sqrt{n})$. Our technique could potentially be applicable to other network design problems where, in addition to minimizing the total weight, one seeks to impose distance constraints between nodes in the network. One class of such problems lies in the context of *hop-constrained network design*: find a network of minimum weight such that the hop-distance between two nodes is at most a given input parameter h. For several hop-constraint problems, strong inapproximability results [DKR16] rule out polylogarithmic *single-criteria* approximation algorithms. There is a long line of work on bicriteria-approximation algorithms for hop-constrained network design problems in general graphs, where one wishes to approximate both the weight of the network and the hop constraint. The state-of-the-art algorithms for general graphs achieve polylogarithmic approximations to both the weight and the hop-constraint; see [HHZ21] and references therein. Remarkably, none of the hop-constrained network design problems are known to admit a bicriteria constant approximation in planar graphs. We anticipate that exploring the applicability of our technique to these problems in planar graphs is a promising research avenue.

[3]These reductions basically apply Baker's technique to the dual planar graph. As we pointed out above, this technique can distort the distances significantly, and hence such a reduction does not seem applicable to our problem.

1.2 Hardness of Minimum Spanners

For *unweighted* planar graphs, the minimum t-spanner problem is fully understood. The regime of stretch $t < 2$ is equivalent to $t = 1$, for which the only 1-spanner is the entire graph. Dragan, Fomin and Golovach [DFG11] developed an efficient polynomial-time approximation scheme (EPTAS) using Baker's technique, constructing a t-spanner with at most $(1 + \delta)|E(G_{\mathsf{opt},t})|$ edges for any $t \geq 2$ and $\delta > 0$. For exact algorithms, Brandes and Handke [BH98] proved NP-hardness for the regime of stretch $t \geq 5$, which was later improved by Kobayashi [Kob18] to the entire nontrivial regime of stretch $t \geq 2$. Kobayashi [Kob18] also showed that the problem is NP-hard even for planar graphs of maximum degree at most $\Delta = 9$ and $t = 2$ (however, it can be solved in polynomial time for $\Delta \leq 4$ and $t = 2$ [CK94]); Gómez, Miyazawa, a Wakabayashi [GMW23] settled the dichotomy between NP-hardness and polynomial-time algorithms for almost all pairs (Δ, t) of maximum degree and stretch in unweighted planar graphs.

In *weighted* planar graphs, the goal is to find a *minimum weight* t-spanner (generalizing the measure of size in the unweighted setting). The problem remains NP-hard in this setting [BH98, Kob18], and Dragan et al. [DFG11] extended their result to planar graphs with positive integer weights bounded by a constant W, yielding an EPTAS with a running time of $\frac{1}{\delta} \cdot (t \cdot W)^{O((t \cdot W/\delta)^2)} \cdot n$, for any $t \geq 2$ and $\delta > 0$. Remarkably, no hardness result has been established in the stretch regime $1 < t < 2$, which is arguably the most interesting. In Section 5, we prove that the problem of computing an exact minimum-weight $(1+\epsilon)$-spanner in planar graphs is NP-hard, thus closing the only gap left open by previous work.

Theorem 1.3. *For every $\epsilon > 0$, the problem of computing a minimum-weight $(1+\epsilon)$-spanner for a given edge-weighted planar graph $G = (V, E, \mathbf{w})$ with polynomially bounded integral edge weights $\mathbf{w} : E \to \mathbb{N}_+$, is NP-hard.*

2 Technical Overview

In this section, we highlight the key ideas behind the proof of Theorem 1.2.

2.1 Hard instances

As a reminder of the greedy spanner algorithm, it goes over all edges in the input graph in the non-decreasing order in terms of edge weights, and in each iteration it adds the edge to the spanner if the distance stretch between the two endpoints is larger than $1 + \epsilon$.

We will first demonstrate that the *greedy* $(1+\epsilon)$-spanner algorithm incurs a weight approximation of $\Theta(1/\epsilon)$ for some hard instances. With a concrete hard instance at hand, we then demonstrate how a simple modification to the greedy spanner can yield a much better spanner construction for this particular instance. In Section 2.2 we highlight some key insights that are needed for generalizing this simple modification to obtain a *pruning procedure* for general planar instances.

Consider a planar graph $G' = (V', E', \mathbf{w}')$ on $2n+2$ vertices $V' = \{u_0, u_1, \ldots, u_n\} \cup \{v_0, v_1, \ldots, v_n\}$ that includes an edge (u_i, v_i) of weight 1 for every $0 \leq i \leq n$, and two edges $(u_0, u_j), (v_0, v_j)$ of weight $\epsilon/2$ each, for every $1 \leq j \leq n$. See Figure 1 for an illustration. The minimum weight $(1+\epsilon)$-spanner of G' consists of edges $\{(u_0, v_0)\} \cup \{(u_0, u_j), (v_0, v_j) : 1 \leq j \leq n\}$ and has total weight $1 + n\epsilon$. However, the greedy algorithm for stretch $1+\epsilon$, and in fact for any stretch less than $1+2\epsilon$, could possibly begin with a minimum spanning tree consisting of edges $\{(u_1, v_1)\} \cup \{(u_0, u_j), (v_0, v_j) : 1 \leq j \leq n\}$, and then greedily add all edges $(u_i, v_i), 2 \leq i \leq n$, incurring a total weight of $(1 + \epsilon)n$. Thus, the approximation ratio of the greedy algorithm is $(n + \epsilon n)/(1 + \epsilon n) \approx 1/\epsilon$ when n grows, which is asymptotically as large as the naive existential bound. In Section 6, we present a stronger hard

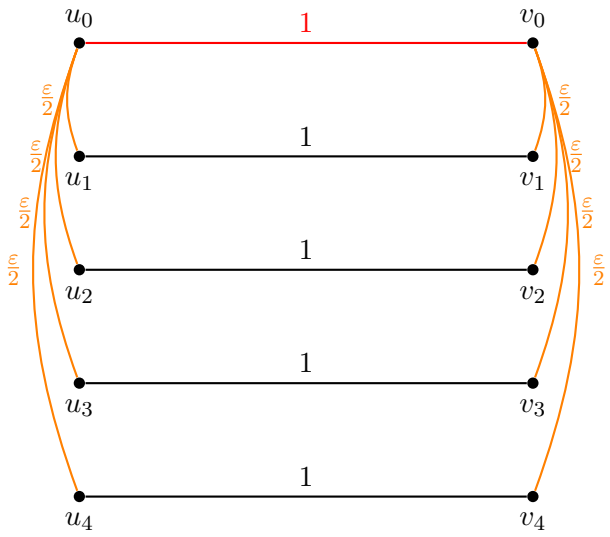


Figure 1: In this example, the optimal $(1+\epsilon)$-spanner contains all the red and orange edges, but a greedy $(1+\epsilon)$-spanner might contain all edges in the graph except for the red edge (u_0, v_0).

instance, for which the weight of the greedy $(1+x\epsilon)$-spanner, for any $1 \leq x = O(\epsilon^{-1/2})$, exceeds the minimum weight of any $(1+\epsilon)$-spanner by a factor of $\Omega\left(\frac{\epsilon^{-1}}{x^2}\right)$. In other words, the greedy algorithm performs poorly even as a *bicriteria approximation* algorithm.

Observe that the main reason that the greedy algorithm returns a heavy spanner is that it misses the *critical* edge (u_0, v_0), which, together with the $2n$ edges in $\{(u_0, u_j), (v_0, v_j) : 1 \leq j \leq n\}$, "serves" (i.e., provides $(1+\epsilon)$-spanner paths between) all pairs u_i, v_j of vertices in G. In general, the greedy $(1+\epsilon)$-spanner might miss some *critical paths* (instead of direct edges), each of which serves many pairs of vertices simultaneously in the unknown optimal $(1+\epsilon)$-spanner, while the greedy $(1+\epsilon)$-spanner has to connect each of these pairs of vertices separately, incurring a much higher total weight. Consequently, to outperform the $O(1/\epsilon)$-approximation of the greedy $(1+\epsilon)$-spanner algorithm, our high-level strategy is to identify critical paths that can replace as many existing edges in the greedy spanner as possible. For example, in the instance shown in Figure 1, edge (u_0, v_0) would be a critical edge, and adding (u_0, v_0) to the greedy spanner allows us to remove the edges (u_i, v_i), $1 \leq i \leq n$, thereby achieving a much lower, and in fact the optimal, weight.

2.2 A pruning framework in planar graphs

To reduce the approximation ratio to $O(1)$ for general instances, we develop a certain *pruning procedure*, which is inspired by the pruning framework in [LST+24], developed for Euclidean low-dimensional spaces, and is also reminiscent of the standard local search approach in the broader context of approximation algorithms.

At a high level, following [LST+24], we start with a (greedy) $(1+\epsilon)$-spanner H for the input graph $G = (V, E, \mathbf{w})$ that has a (high) approximation ratio $O(1/\epsilon)$ (in the Euclidean setting the initial approximation is $O(1/\epsilon^2)$), find two sets of edges $F^{\text{new}} \subseteq E$ and a $F^{\text{old}} \subseteq E(H)$, and exchange them: $H_1 \leftarrow F^{\text{new}} \cup (H \setminus F^{\text{old}})$. The key guarantees of our construction are: (i) H_1 has stretch $(1+O(\epsilon))$ and (ii) $\mathbf{w}(H_1) = O(\log 1/\epsilon) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$. Thus, we pruned a set of heavy edges F^{old} to reduce the approximation ratio for the weight of the spanner H *exponentially* at the expense of a slight increase in the stretch. We can then apply the same pruning procedure to H_1. By repeating

the pruning procedure $\log^*(1/\epsilon)$ times, we obtain a spanner H^* with stretch $(1+\epsilon \cdot 2^{O(\log^*(1/\epsilon))})$ and weight

$$\mathbf{w}(H^*) = O(1) \underbrace{\log \ldots \log(1/\epsilon)}_{\text{iterate } \log^*(1/\epsilon) \text{ times}} \mathbf{w}(G_{\mathsf{opt},\epsilon}) = O(1) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}),$$

as claimed in Theorem 1.2.

Our main technical contribution lies in efficiently finding the heavy(-weight) pruned set F^{old} and the light replacement set F^{new} in each iteration of the pruning procedure. We note that the *existence* of such sets is immediate: simply take $F^{\text{old}} = E(H)$ (the edges of the current spanner) while $F^{\text{new}} = E(G_{\mathsf{opt},\epsilon})$ (the edges of the optimal solution). However, $G_{\mathsf{opt},\epsilon}$ is not known, and the *algorithmic task of efficiently (in poly-time) computing* such sets F^{old} and F^{new}—on which our approximation algorithm crucially relies—is highly nontrivial, as discussed next.

In Euclidean spaces, Le et al. [LST+24] rely on Euclidean geometry in a crucial way to find such sets F^{old} and F^{new}. For each edge (u,v), let $\mathcal{E}_{uv}$ be the ellipsoid of width $O(\sqrt{\epsilon})$ that has u and v as the foci. A basic observation is that $\mathcal{E}_{uv}$ contains all points on any $(1+\epsilon)$-approximate path between u and v. One of the key ideas in [LST+24] is the following. Consider any two edges (s,t) and (s',t') of almost the same length, say $\Theta(\ell)$ for some $\ell > 0$, and suppose there are two points z, w such that: (i) both z and w lie in the intersection of the two ellipsoids $\mathcal{E}_{st}$ and $\mathcal{E}_{s't'}$, and (ii) $\|z-w\|_2 = \Theta(\ell)/c$ for a sufficiently large constant c. Item (i) guarantees that, by taking (z,w) to F^{new} (to be added to the spanner) and both edges $\{(s,t),(s',t')\}$ to F^{old} (to be removed from the spanner), the stretch of the new spanner may grow only slightly; item (ii) guarantees that F^{old} is heavy while F^{new} is light. Edge (z,w) is called a *helper* edge in [LST+24]. Clearly, Euclidean geometry is central to the approach in [LST+24]. Furthermore, [LST+24] also used the fact that in Euclidean spanners, the graph G is the complete graph on n input points in $\mathbb{R}^d$, and so any two vertices are connected by an edge, thus the helper edge can be added to F^{new}; in planar graphs, however, most pairs of vertices are non-adjacent.

A laminar structural property. In light of the above discussion, our primary objective is to efficiently compute critical paths in G that could replace a set of edges with large total weight in any given spanner $H \subseteq G$ such that $\mathbf{w}(H) \gg \mathbf{w}(G_{\mathsf{opt},\epsilon})$ (possibly slightly increasing the stretch of H). Before we can proceed to the *algorithmic task of efficiently computing* such critical paths, we must first prove their *existence*, which by itself is nontrivial. In particular, removable edges could form far more intricate structures than the basic ladder-like graph shown in Figure 1; for example, there could be multiple ladders hanging on a critical path (see Figure 2 for an illustration). The algorithmic task of computing such paths poses further technical challenges; it turns out that these challenges can be overcome by carefully employing dynamic programming, as discussed below.

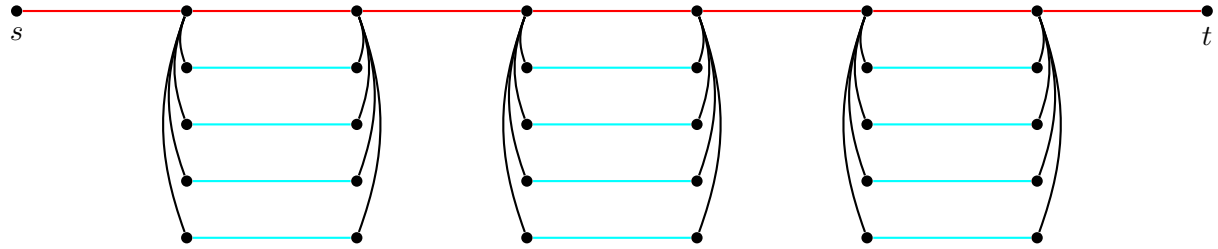


Figure 2: There could be multiple ladder structures that are hanging on a single critical path, colored red. The blue edges forming the ladders are removable edges of the current spanner H.

To be more precise, we shall consider critical paths with somewhat relaxed constraints. Specif-

ically, the distances from removable edges to the critical path will not necessarily be as small as an ϵ-fraction of the lengths of the removable edges, as in the hard instance shown in Figure 1. For general instances, we can only guarantee a much weaker upper bound of a $(1-\Omega(1))$-fraction instead of an ϵ-fraction; this is the main reason why our pruning procedure increases the stretch. With the above relaxation, we can ultimately prove the existence of a path ρ^* and an edge set $F^* = F^*(\rho^*) \subseteq E(H)$, such that $\frac{\mathbf{w}(F^*)}{\mathbf{w}(\rho^*)} \approx \frac{\mathbf{w}(H)}{\mathbf{w}(G_{\mathsf{opt},\epsilon})}$, while the stretch of $H \cup E(\rho^*) \setminus F^*$ only slightly exceeds that of H. Hence, adding ρ^* to H and pruning F^* from H produces a significantly lighter spanner with almost the same stretch guarantee. We refer to such a pair, ρ^* and F^*, as a *pruning pair*.

We note that proving the existence of a pruning pair using a naive averaging argument is doomed. Specifically, let us associate every edge $(s,t) \in E(H)$ with a shortest path $\gamma_{s,t}$ in $G_{\mathsf{opt},\epsilon}$ and map each such path $\gamma_{s,t}$ to a disjoint edge set $F_{s,t} \subseteq E(H)$, where the union of all paths $\gamma_{s,t}$ (respectively, edge sets $F_{s,t}$) is $G_{\mathsf{opt},\epsilon}$ (resp., H). Consequently, an average path $\gamma_{s,t}$ in $G_{\mathsf{opt},\epsilon}$ is mapped to a set $F_{s,t} \subseteq E(H)$ of total weight $\mathbf{w}(\gamma_{s,t}) \cdot \frac{\mathbf{w}(H)}{\mathbf{w}(G_{\mathsf{opt},\epsilon})}$; note that $\frac{\mathbf{w}(F_{s,t})}{\mathbf{w}(\gamma_{s,t})} \approx \frac{\mathbf{w}(H)}{\mathbf{w}(G_{\mathsf{opt},\epsilon})}$. It is now tempting to argue that $\gamma_{s,t}$ could replace this heavy edge set $F_{s,t} \subseteq E(H)$ (which would reduce the weight by the required amount) without blowing up the stretch. Alas, the main issue with such an averaging argument is that it completely ignores cases where $\gamma_{s,t}$ intersects many different paths $\gamma_{u,v}$, but any single path $\gamma_{u,v}$ shares only a small proportion with $\gamma_{s,t}$, and so the path $\gamma_{s,t}$ itself could not replace any other edge $(u,v) \in E(H)$ without significantly blowing up the stretch.

In our proof, to establish the existence of a pruning pair ρ^*, F^*, where the single path ρ^* in $G_{\mathsf{opt},\epsilon}$ can remove the much heavier set of edges F^* in H without blowing up the stretch, we need to drill much deeper, by *leveraging planarity* and exploring certain *laminar structures of all paths* $\{\gamma_{s,t} : (s,t) \in E(H)\}$. First, we introduce the following key definition of *κ-hanging*, which formalizes the notion of removable edges (a canonical setting of κ is $\kappa = 2/3$):

Definition 2.1 (κ-hang)**.** Consider any (not necessarily simple) path $\rho = \langle v_1, v_2, \ldots, v_k \rangle$ in G and any edge $(a,b) \in E(H)$. We say that edge (a,b) is *κ-hanging* on the path ρ if there exists a pair of vertices v_i, v_j with $i < j$ such that:

(1) $\mathbf{w}(\rho[v_i, v_j]) \geq \kappa \cdot \mathbf{w}(a,b)$, where $\rho[v_i, v_j]$ denotes the subpath of ρ between v_i and v_j;
intuitively, the lower bound on $\mathbf{w}(\rho[v_i, v_j])$ will be helpful when we add the path $\rho[v_i, v_j]$ and remove the edge (a,b) in spanner H while preserving the stretch;

(2) $\mathsf{dist}_G(a, v_i) + \mathbf{w}(\rho[v_i, v_j]) + \mathsf{dist}_G(v_j, b) \leq (1+\epsilon) \cdot \mathbf{w}(a,b)$.

We say that (a,b) is *κ-hanging* at (v_i, v_j) on ρ.

See Figure 3 for an illustration. We argue that, since $\kappa = 2/3$, and due to items (1) and (2) in Definition 2.1, the spanner obtained from the current $(1+\epsilon)$-spanner, by adding ρ to it and removing all the κ-hanging edges from it, has a stretch of at most $1 + O(\epsilon)$. The reason why the additive term ϵ has to increase by some constant factor to $O(\epsilon)$ is that our current spanner may not include the optimal sub-paths in G from a to v_i or from v_j to b. In fact, our current spanner may not even include $(1+\epsilon)$-spanner paths between these points, as we are iteratively pruning paths, hence our argument has to bypass several hurdles in order to prove the stretch bound.

Proving the stretch bound alone is insufficient—we also need to guarantee that the weight reduces significantly, and thus it is crucial that the κ-hanging edges are heavy with respect to the path on which they are hanging. Furthermore, while proving the existence of a path ρ^* with heavy κ-hanging edges is already nontrivial, the existence alone does not guarantee an efficient algorithm. Our key insight is that we can impose a *laminar structure* of κ-hanging paths in a way that can be

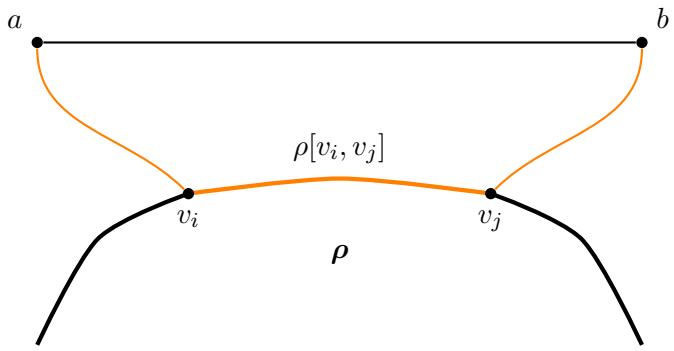


Figure 3: An example for an edge (a, b) that is κ-hanging at (v_i, v_j) on path ρ. The orange path between a and b has length at most $(1+\epsilon) \cdot \mathbf{w}(a, b)$; the orange sub-path of ρ between v_i and v_j, which is also a sub-path of the orange path between a and b, has length at least $\kappa \cdot \mathbf{w}(a, b)$.

exploited for an efficient algorithm, formalized by the following structural lemma; when applying the lemma, we may assume that the weight of the current $(1+\epsilon)$-spanner H exceeds that of the optimal solution $G_{\mathsf{opt},\epsilon}$ by a factor of α, for a large parameter $\alpha \gg 1$.

Lemma 2.2 (structural property, simplified)**.** *Define $\alpha = \frac{w(H)}{\mathbf{w}(G_{\mathsf{opt},\epsilon})}$. There exists a pruning pair that consists of a shortest path ρ^* in $G_{\mathsf{opt},\epsilon}$ and an edge set $F^* \subseteq E(H)$, with the following guarantees:*

(1) Each edge $e \in F^*$ is $\frac{2}{3}$-hanging on ρ^*;

(2) The total weight $\mathbf{w}(F^*)$ of F^* satisfies $\mathbf{w}(F^*) \geq \frac{\alpha}{6} \cdot \mathbf{w}(\rho^*)$;

(3) For each edge $e \in F^*$, let (a_e, b_e) be the hanging points of e on ρ^*; then all the sub-path intervals $\{\rho^*[a_e, b_e] : e \in F^*\}$ form a laminar family; recall that a laminar family is a family of sets where any two sets are either disjoint or related by containment.

By the first guarantee, the edges in F^* are 2/3-hanging; as discussed above, which means that we can remove them from H without incurring much stretch. The second guarantee implies that the set F^* of removable edges is heavy with respect to the weight of the edges in ρ^*, and hence removing the edges in F^* from the spanner H and adding the edges in ρ in their place results in a significant reduction to the weight of H. The third guarantee, namely the laminar family structure, is crucial for obtaining an efficient dynamic programming algorithm to compute an approximation of ρ^*. Here, we heavily exploit the *planarity* of the input graph in establishing the laminar property.

Dynamic programming. The structural property, which asserts the *existence* of pruning pairs as stated in Lemma 2.2, does not immediately lead to an efficient pruning procedure; indeed, in our existential proof, the structure of critical paths will depend on $G_{\mathsf{opt},\epsilon}$, which is of course unknown. So the algorithmic goal is to find an approximate shortest path ρ between some vertex pair s, t such that a large amount of edges in H could be hanging on ρ in some ladder-like manner, similarly to the illustration of Figure 2. For this task, we will adopt a dynamic programming approach which is fairly natural: we will maintain two tables (this is only an informal description, and some details in the main algorithm are omitted here)

$$\{\rho[s, t, L] : s, t \in V, L \leq (1+\epsilon)\,\mathsf{dist}_G(s, t)\}$$

$$\{P[s, t, L] : s, t \in V, L \leq (1+\epsilon)\,\mathsf{dist}_G(s, t)\},$$

where $\rho[s,t,L]$ will be an approximate shortest path (not necessarily a simple path) between s and t, and $P[s,t,L] \subseteq E(H)$ will be a set of edges which can $\frac{1}{3(1+\epsilon)}$-hang on $\rho[s,t,L]$; note that the hanging parameter now degrades by roughly a factor of 2, from $\frac{2}{3}$ to $\frac{1}{3(1+\epsilon)}$, which is needed for technical reasons. The transition rule of the dynamic programming table would be to find the best intermediate vertex z and $1 \leq L' < L$ so that the concatenated path $\rho[s,z,L'] \circ \rho[z,t,L-L']$ collects the heaviest possible edge set.

The main technical issue here is that the two removable edge sets, $P[s,z,L']$ and $P[z,t,L-L']$, are usually not disjoint, and so $\mathbf{w}(P[s,z,L'] \cup P[z,t,L-L'])$ could be much smaller than $\mathbf{w}(P[s,z,L']) + \mathbf{w}(P[z,t,L-L'])$. If we take the intermediate vertex z and L' that maximize the total weight of the union $P[s,z,L'] \cup P[z,t,L-L']$, then we could suffer a significant under-estimation of the total weight of edges removable by the best approximate shortest path between s and t, especially when $P[s,z,L']$ and $P[z,t,L-L']$ have large intersections. Hence, ultimately, the ratio $\frac{\mathbf{w}(P[s,t,L])}{\mathbf{w}(\rho[s,t,L])}$ may be significantly smaller than $\frac{\mathbf{w}(F^*)}{\mathbf{w}(\rho^*)}$.

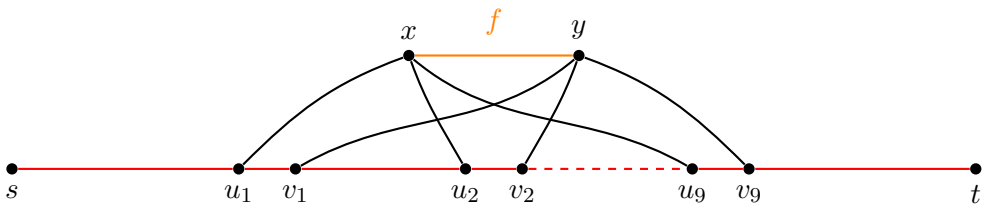


Figure 4: In this figure, the red path is $\rho[s,t,L]$, and $f = (x,y)$ appears many times in the multi-set $P[s,t,L]$. Then, we can show that f can hang on the path $\rho[s,t,L]$ at multiple positions, say $(u_1,v_1),(u_2,v_2),\ldots,(u_9,v_9)$. Then, we can replace the sub-path $\rho[u_1,v_9]$ with a shortcut $u_1 \to (x,y) \to v_9$, which reduces the total weight of ρ by at least $\mathbf{w}(f)$.

To resolve this issue, we will instead maximize the *union with multiplicities*, $P[s,z,L'] \uplus P[z,t,L-L']$, whose total weight is $\mathbf{w}(P[s,z,L']) + \mathbf{w}(P[z,t,L-L'])$. By using such an *over-estimation*, we will be able to *lower bound* $\frac{\mathbf{w}(P[s,t,L])}{\mathbf{w}(\rho[s,t,L])}$ with $\frac{\mathbf{w}(F^*)}{\mathbf{w}(\rho^*)}$. It turns out that we can handle multi-set union and the consequent over-estimation much more effectively than an ordinary set union and the consequent under-estimation. The issue with an over-estimation is that when we replace the edge set $P[s,t,L]$ with the path $\rho[s,t,L]$, the total weight of H usually does not decrease by the amount of $\mathbf{w}(P[s,t,L])$, since $P[s,t,L]$ is a multi-set. To lower bound the total weight of edges that we can actually prune, we will upper bound the multiplicities of most edges in the multi-set $P[s,t,L]$ by a large constant (say 20), crucially relying on the fact that $\rho[s,t,L]$ is an approximate shortest path. Roughly speaking, imagine that some edge $f \in P[s,t,L]$ has a high multiplicity, then we can show that f must hang on the path $\rho[s,t,L]$ at many different positions. In this case, there exists a shortcut on $\rho[s,t,L]$ through the edge f that can reduce the total weight of $\rho[s,t,L]$ by at least $\mathbf{w}(f)$. Since we know that $\mathbf{w}(\rho[s,t,L])$ is at most $(1+\epsilon)\mathsf{dist}_G(s,t)$ beforehand, we can upper bound the total weight improvements due to shortcuts by $\epsilon \cdot \mathsf{dist}_G(s,t)$, which, in turn, yields the required constant upper bound on the overall multiplicities of edges in $P[s,t,L]$. See Figure 4 for an illustration.

3 Preliminaries

For every integer $x \geq 1$, let $\lfloor x \rfloor_2$ be the largest integer power of 2 not exceeding x (that is, $\lfloor x \rfloor_2 = 2^{\lfloor \log_2 x \rfloor}$). Let $G = (V,E,\mathbf{w})$ be an undirected weighted planar graph, where $\mathbf{w} : E \to \{1,2,\ldots,W\}$

is an integral edge weight function. Let $\epsilon > 0$ be the input stretch parameter. We assume that shortest paths are unique in both G and $G_{\mathsf{opt},\epsilon}$ by breaking ties lexicographically.

For any $s, t \in V$, let $\mathsf{dist}_G(s,t)$ denote the length of the shortest path between s and t in G. For any (not necessarily simple) path $\rho = \langle v_1, v_2, \ldots, v_k \rangle$ in G and any pair of indices $1 \le i < j \le k$, let $P[v_i, v_j]$ be the sub-path of P between v_i and v_j. For any two (not necessarily simple) paths ρ_1 and ρ_2, let $\rho_1 \circ \rho_2$ be their concatenation if they share one endpoint. For any subgraph H of G, let $\mathbf{w}(H) = \sum_{e \in E(H)} \mathbf{w}(e)$ be the total weight of edges in H. For any multi-set of edges F, let $\mathbf{w}(F)$ be the total weight of edges in F. For two multi-sets F_1 and F_2, let $F_1 \uplus F_2$ denote the multi-set union of F_1 and F_2 by summing the multiplicities of each element.

4 Pruning Planar Light Spanners

Throughout this section, we assume that $W < n^2/\epsilon$; in the end we will show how to deal with general cases. In order to prove Theorem 1.2, our main technical contribution is a pruning algorithm, as summarized in the following statement.

Theorem 4.1. *Take two parameters $\epsilon, \delta > 0$ such that $\epsilon \le \min\{10^{-2}, \delta\}$. Let $G = (V, E, \mathbf{w})$ be an undirected planar graph with positive integral edge weights $\mathbf{w} : E \to \mathbb{N}_+$, and let H be a $(1+\delta)$-spanner of G such that $\theta := \mathbf{w}(H)/\mathbf{w}(G_{\mathsf{opt},\epsilon}) \ge \Omega(1)$. Then one can compute, in polynomial time, two sets of edges, $F^{\mathrm{new}} \subseteq E$ and $F^{\mathrm{old}} \subseteq E(H)$, such that the following holds:*

(1) the stretch of graph $H_1 = F^{\mathrm{new}} \cup (H \setminus F^{\mathrm{old}})$ is at most $1 + O(1) \cdot \delta$, and

(2) $\mathbf{w}(H_1) \le O(\log \theta) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$.

Our main theorem follows immediately by a successive application of Theorem 4.1.

Proof of Theorem 1.2. Starting with H being the greedy light $(1+\epsilon)$-spanner from [ADD+93] of weight $O(1/\epsilon) \cdot \mathbf{w}(G_{\mathsf{mst}}) \le O(1/\epsilon) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$, we successively apply Theorem 4.1 and update $H \leftarrow H_1$ for $O(\log^*(1/\epsilon))$ iterations. At the end, H is a $\left(1 + \epsilon \cdot 2^{O(\log^*(1/\epsilon))}\right)$-spanner of weight $O(1) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$. □

The rest of this section is dedicated to the proof of Theorem 4.1. In Section 4.1, we present a pruning algorithm that finds the edge sets $F^{\mathrm{new}} \subseteq E$ and $F^{\mathrm{old}} \subseteq E(H)$, followed by the weight and stretch analyses establishing properties (1) and (2) in Section 4.2.

4.1 Description of the Pruning Algorithm and Runtime Analysis

Definition 4.2. Consider any (not necessarily simple) path $\rho = \langle v_1, v_2, \ldots, v_k \rangle$ in G and any edge $(a,b) \in E(H)$. We say that edge (a,b) can *κ-hang* on the path ρ if there exists a pair of vertices v_i, v_j with $i < j$ such that

(1) $\mathbf{w}\left(\rho[v_i, v_j]\right) \ge \kappa \cdot \mathbf{w}(a,b)$;

(2) $\mathsf{dist}_G(a, v_i) + \mathbf{w}(\rho[v_i, v_j]) + \mathsf{dist}_G(v_j, b) \le (1+\epsilon) \cdot \mathbf{w}(a,b)$.

The vertex pair (v_i, v_j) will be called the *κ-hanging* points of (a,b) on ρ; or equivalently, we say that (a,b) is *κ-hanging* at (v_i, v_j) on ρ.

Initially, set $F^{\text{new}} \leftarrow \emptyset$ and $F^{\text{old}} \leftarrow \emptyset$. We will repeatedly find an approximate shortest path ρ on which a large number of edges in $P \subseteq E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ can $\frac{1}{3(1+\epsilon)}$-hang: In each round, we prune these $\frac{1}{3(1+\epsilon)}$-hanging edges in P by setting $F^{\text{old}} \leftarrow F^{\text{old}} \cup P$, and then add the path ρ to the spanner H_1 by setting $F^{\text{new}} \leftarrow F^{\text{new}} \cup E(\rho)$. We expect that adding $E(\rho)$ and removing P would significantly reduce the total weight while approximately preserving distance stretch.

To find such a good approximate shortest path ρ in each round, we will rely on a dynamic programming approach. The dynamic programming procedure maintains three tables, described in the following.

A dynamic programming table for triples (s, t, L), where $s, t \in V(G)$ and $L \leq (1+\epsilon) \cdot \mathsf{dist}_G(s, t)$ is a positive integer.

- A table of paths $\rho[s, t, L]$, for all $(s, t, L) \in V \times V \times [2nW]$ such that $L \leq (1+\epsilon) \cdot \mathsf{dist}_G(s, t)$., where $\rho[s, t, L]$ is a (not necessarily simple) path between s and t in G such that $\mathbf{w}(\rho[s, t, L]) = L$.
- A table of edge multi-sets $P[s, t, L] \subseteq E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$, for all $(s, t, L) \in V \times V \times [2nW]$ such that $L \leq (1+\epsilon) \cdot \mathsf{dist}_G(s, t)$, and each edge $e \in P[s, t, L]$ is $\frac{1}{3(1+\epsilon)}$-hanging on the path $\rho[s, t, L]$.
- A table of edge weight sums $\mathsf{DP}[s, t, L] \in \mathbb{N}^+$, for all $(s, t, L) \in V \times V \times [2nW]$ such that $L \leq (1+\epsilon) \cdot \mathsf{dist}_G(s, t)$ and $\mathsf{DP}[s, t, L] = \mathbf{w}(P[s, t, L])$.

Intuitively, our goal is to find an approximate shortest path $\rho[s, t, L]$ on which an edge set $P[s, t, L] \subseteq E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ of large total weight can be $\frac{1}{3(1+\epsilon)}$-hanging. Due to technical reasons, we approximate the charging sets by multi-sets (rather than ordinary sets).

We compute the entries of the three tables as follows. For every pair of vertices $s, t \in V$, let $\pi_{s,t}$ be the shortest path between s and t in G. First, compute the (non-multi) set $B[s, t]$ of all edges $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ that is $\frac{1}{3(1+\epsilon)}$-hanging at (s, t) on $\pi_{s,t}$; this computation can be done in $O(n^3)$ time.

Initialize $\rho[s, t, \mathsf{dist}_G(s, t)] \leftarrow \pi_{s,t}$ (i.e., a shortest st-path in G), $P[s, t, L] \leftarrow B[s, t]$, and $\mathsf{DP}[s, t, L] \leftarrow \mathbf{w}(B[s, t])$ for any $L \leq (1+\epsilon)\mathsf{dist}_G(s, t)$. Next, go over every length parameter $L = 1, 2, \ldots, nW$. For every pair (s, t) such that $L \leq (1+\epsilon) \cdot \mathsf{dist}_G(s, t)$, we choose a via point $z \in V$ and $0 \leq L' < L$ by setting

$$(z, L') = \arg\max_{(z,L')} \left\{ \mathsf{DP}[s, z, L'] + \mathsf{DP}[z, t, L - L'] + \mathbf{1}\left[\max\{L', L - L'\} < \lfloor L \rfloor_2\right] \cdot \mathbf{w}(B[s, t]) \right\},$$

and then assign

$$\begin{aligned}
\mathsf{DP}[s, t, L] &\leftarrow \mathsf{DP}[s, z, L'] + \mathsf{DP}[z, t, L - L'] + \mathbf{1}\left[\max\{L', L - L'\} < \lfloor L \rfloor_2\right] \cdot \mathbf{w}(B[s, t]), \\
P[s, t, L] &\leftarrow \begin{cases} P[s, z, L'] \uplus P[z, t, L - L'] & \text{if } \max\{L', L - L'\} \geq \lfloor L \rfloor_2 \\ P[s, z, L'] \uplus P[z, t, L - L'] \uplus B[s, t] & \text{if } \max\{L', L - L'\} < \lfloor L \rfloor_2, \end{cases} \\
\rho[s, t, L] &\leftarrow \rho[s, z, L'] \circ \rho[z, t, L - L'].
\end{aligned}$$

Intuitively, to find the best path between s, t, we identify the best via point z' along with a length parameter L' and build the path $\rho[s, t, L]$ as the concatenation of paths $\rho[s, z, L']$ and $\rho[z, t, L - L']$,

and entries $P[s,t,L], \mathsf{DP}[s,t,L]$ keep track of the pruned sets and weights associated with path $\rho[s,t,L]$.

After all entries in the dynamic programming table have been computed, which takes $O(n^5W^2) = n^{O(1)}$ time, find the triple (s^*,t^*,L^*) which maximizes the ratio

$$\beta = \frac{\mathsf{DP}[s^*,t^*,L^*]}{\mathbf{w}\left(\rho[s^*,t^*,L^*]\right)}.$$

If $\beta \geq 1$, then update $F^{\text{new}} \leftarrow F^{\text{new}} \cup E(\rho[s^*,t^*,L^*])$ and $F^{\text{old}} \leftarrow F^{\text{old}} \cup P[s^*,t^*,L^*]$, and start a new round (recomputing the dynamic programming tables); otherwise our pruning algorithm terminates and returns

$$H_1 = F^{\text{new}} \cup (H \setminus F^{\text{old}})$$

The whole algorithm is summarized below as Algorithm 1.

Algorithm 1: Prune a $(1+\delta)$-spanner $H \subseteq G$

initialize $F^{\text{new}}, F^{\text{old}} \leftarrow \emptyset$;
while *true* **do**
  compute $B[s,t] \subseteq E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ which are edges that is $\frac{1}{3(1+\epsilon)}$-hanging at (s,t) on $\pi_{s,t}$;
  initialize $\rho[s,t,\mathsf{dist}_G(s,t)] \leftarrow \pi_{s,t}, P[s,t,L] \leftarrow B[s,t], \mathsf{DP}[s,t,L] \leftarrow \mathbf{w}\left(B[s,t]\right), \forall L \leq (1+\epsilon)\mathsf{dist}_G(s,t)$;
  for $L = 1, 2, \ldots, nW$ **do**
    for (s,t) *such that* $L \leq (1+\epsilon)\cdot \mathsf{dist}_G(s,t)$ **do**
      choose a vertex $z \in V$ and $0 \leq L' < L$ such that: $(z,L') \leftarrow \arg\max_{(z,L')} \{\mathsf{DP}[s,z,L'] + \mathsf{DP}[z,t,L-L'] + \mathbf{1}\left[\max\{L', L-L'\} < \lfloor L \rfloor_2\right] \cdot \mathbf{w}\left(B[s,t]\right)\}$;
      $\mathsf{DP}[s,t,L] \leftarrow \mathsf{DP}[s,z,L'] + \mathsf{DP}[z,t,L-L'] + \mathbf{1}\left[\max\{L', L-L'\} < \lfloor L \rfloor_2\right] \cdot \mathbf{w}\left(B[s,t]\right)$;
      $P[s,t,L] \leftarrow \begin{cases} P[s,z,L'] \uplus P[z,t,L-L'] & \text{if } \max\{L', L-L'\} \geq \lfloor L \rfloor_2 \\ P[s,z,L'] \uplus P[z,t,L-L'] \uplus B[s,t] & \text{if } \max\{L', L-L'\} < \lfloor L \rfloor_2, \end{cases}$;
      $\rho[s,t,L] \leftarrow \rho[s,z,L'] \circ \rho[z,t,L-L']$;
  find the triple (s^*,t^*,L^*) which maximizes the ratio $\beta = \frac{\mathsf{DP}[s^*,t^*,L^*]}{\mathbf{w}(\rho[s^*,t^*,L^*])}$;
  if $\beta \geq 1$ **then**
    update $F^{\text{new}} \leftarrow F^{\text{new}} \cup E(\rho[s^*,t^*,L^*])$, $F^{\text{old}} \leftarrow F^{\text{old}} \cup P[s^*,t^*,L^*]$;
  else
    break;
return $H_1 \leftarrow F^{\text{new}} \cup (H \setminus F^{\text{old}})$;

4.2 Weight Analysis

Let us begin with some basic properties of the dynamic programming scheme.

Lemma 4.3. *Throughout the course of the dynamic programming algorithm, for any triple (s,t,L), $L \leq (1+\epsilon)\mathsf{dist}_G(s,t)$, the value of $\mathsf{DP}[s,t,L]$ is non-decreasing, and $\mathsf{DP}[s,t,L] \geq \mathbf{w}(B[s,t])$.*

Proof. At the beginning, we have $\mathsf{DP}[s,t,L] = \mathbf{w}(B[s,t])$. In each update, if we take $L' = 0$ and $z = s$, then the value of $\mathsf{DP}[s,z,L'] + \mathsf{DP}[z,t,L-L'] + \mathbf{1}\left[\max\{L', L-L'\} < \lfloor L \rfloor_2\right] \cdot \mathbf{w}\left(B[s,t]\right)$ is exactly the current value of $\mathsf{DP}[s,t,L]$. Since (z,L') is the maximizer, we know that $\mathsf{DP}[s,t,L]$ never decreases. □

4.2.1 Structural Properties

Let us begin with a high-level outline before focusing on technical details. When H is much heavier than the unknown optimal spanner $G_{\mathsf{opt},\epsilon}$, our goal is to add to H a small set of new edges to help remove a large set of old edges from H. The purpose of this subsection, roughly speaking, is to identify a key structural property that there always exists an approximate shortest path ρ^* in G as well as an edge subset $F^* \subseteq E(H)$, such that $\frac{\mathbf{w}(\rho^*)}{\mathbf{w}(F^*)} \approx \frac{\mathbf{w}(G_{\mathsf{opt},\epsilon})}{\mathbf{w}(H)}$, and $E(\rho^*) \cup (H \setminus F^*)$ has stretch $1+O(\delta)$. As we shall see shortly, finding the ideal choice of ρ^*, F^* requires knowledge of the optimal $(1+\epsilon)$-spanner $G_{\mathsf{opt},\epsilon}$, but we will discuss how to find ρ^*, F^* algorithmically by allowing approximations later on.

Fix a plane embedding of G (the embedding is used only in the analysis). For each edge $(a,b) \in E(H)$, let $\gamma_{a,b}$ be the shortest ab-path in $G_{\mathsf{opt},\epsilon}$. Since G is an embedded planar graph, we can define $R_{a,b} \subseteq \mathbb{R}^2$ as the bounded region enclosed by the edge (a,b) and the path $\gamma_{a,b}$.

Lemma 4.4. *For any two distinct edges $e_1, e_2 \in E(H)$, the regions R_{e_1} and R_{e_2} are either interior-disjoint, or one contains the other.*

Proof. The proof is evident because shortest paths and edges do not cross each other in any plane embedding of G. See Figure 5 for an illustration. □

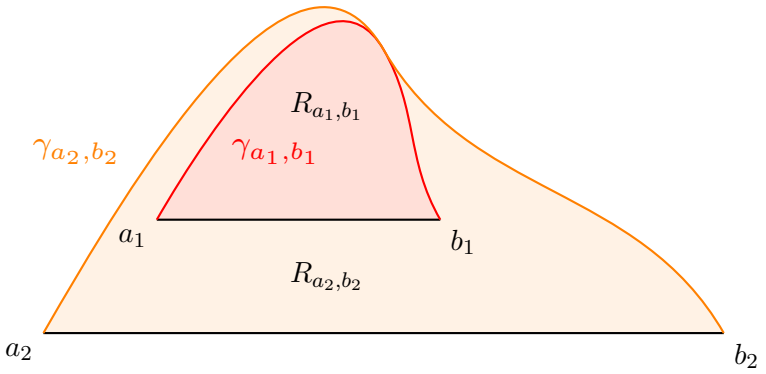


Figure 5: In this example, $e_1 = (a_1,b_1), e_2 = (a_2,b_2)$ and region R_{e_1} is contained within region R_{e_2}.

By Lemma 4.4, the regions R_e, $e \in E(H)$, naturally form a laminar family. Create a rooted tree $\mathcal{T}$ where each node corresponds to a region R_e, and a region R_{e_1} is an ancestor of another region R_{e_2} if and only if $R_{e_1} \supseteq R_{e_2}$.

Consider any iteration of the while-loop of Algorithm 1. Define a weight ratio

$$\alpha = \frac{\mathbf{w}\left(H \setminus (F^{\text{new}} \cup F^{\text{old}})\right)}{\mathbf{w}\left(G_{\mathsf{opt},\epsilon}\right)}. \tag{1}$$

The following structural property will be important to our analysis.

Lemma 4.5 (structural property)**.** *There exists a shortest path ρ^* in $G_{\mathsf{opt},\epsilon}$ together with an edge set $F^* \subseteq E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that:*

(1) Each edge $e \in F^*$ is $\frac{2}{3}$-hanging on ρ^*;

(2) The total weight $\mathbf{w}(F^*)$ of F^* satisfies $\mathbf{w}(F^*) \geq \frac{\alpha}{6} \cdot \mathbf{w}(\rho^*)$;

(3) For each edge $e \in F^*$, let (a_e, b_e) be the hanging points of e on ρ^*; then all the sub-path intervals $\{\rho^*[a_e, b_e] : e \in F^*\}$ form a laminar family.

Proof. The main part of this proof is to construct a set Γ of shortest paths of the form γ_e for some edges $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$, such that (1) each edge $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ is $\frac{2}{3}$-hanging on some path $\gamma \in \Gamma$, and (2) the total length $\mathbf{w}(\Gamma)$ of all paths in Γ is at most $6\mathbf{w}(G_{\mathsf{opt},\epsilon})$. Having constructed Γ, the proof will be concluded by applying the pigeon-hold principle.

Construction of Γ. Initialize $\Gamma \leftarrow \emptyset$. Traverse the nodes of the tree $\mathcal{T}$ in a depth-first-search order. For each node $R_e \in \mathcal{T}$ such that $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$, let $R_{e_0} \supseteq R_e$ be the lowest ancestor of R_e in $\mathcal{T}$ such that $\gamma_{e_0} \in \Gamma$ (if $\gamma_e \notin \Gamma$ for any ancestor R_e in $\mathcal{T}$, then $\gamma_{e_0} \leftarrow \emptyset$). Let $\eta = \gamma_e \cap \gamma_{e_0}$; note that $\gamma_e \cap \gamma_{e_0}$ is empty or a path because both γ_e and γ_{e_0} are unique shortest paths in $G_{\mathsf{opt},\epsilon}$. Then, we add γ_e to Γ iff $\mathbf{w}(\eta) < \frac{2}{3} \cdot \mathbf{w}(e)$. See Figure 6 for an illustration.

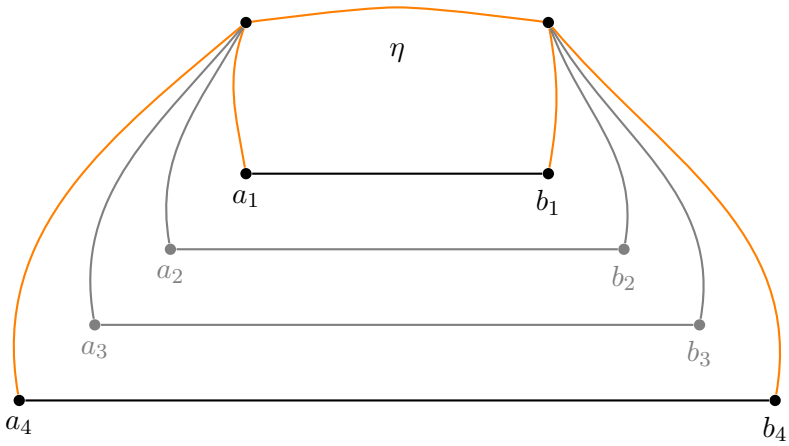


Figure 6: In this picture, $e = (a_1, b_1)$, and $e_0 = (a_4, b_4)$. If $\mathbf{w}(\eta) < \frac{2}{3}\mathbf{w}(e)$, we would add γ_e to Γ which decreases Φ. There are two intermediate regions $R_{(a_2,b_2)}$ and $R_{(a_3,b_3)}$ whose paths $\gamma_{(a_2,b_2)}, \gamma_{(a_3,b_3)}$ were not added to Γ.

Analysing the properties of Γ.

Claim 4.6. *Each edge $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ is $\frac{2}{3}$-hanging on a path in Γ.*

Proof. Consider the step in which node R_e is visited by the depth-first-search, and let R_{e_0} be its lowest ancestor, with $\gamma_{e_0} \in \Gamma$, as defined above. If $\gamma_e \in \Gamma$, then by definition, e is 1-hanging on $\gamma_e \in \Gamma$. Otherwise $\gamma_e \notin \Gamma$, and by the construction we have $\mathbf{w}(\eta) \geq \frac{2}{3} \cdot \mathbf{w}(e)$, where $\eta = \gamma_e \cap \gamma_{e_0}$, which means e is $\frac{2}{3}$-hanging on $\gamma_{e_0} \in \Gamma$. □

Claim 4.7. *The total weight $\mathbf{w}(\Gamma)$ is at most $6 \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$.*

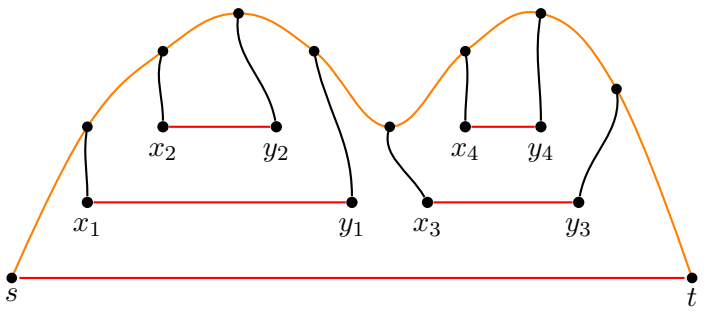


Figure 7: The path ρ^* is drawn as the orange curve, and $C(\rho^*)$ are the red edges which are $\frac{2}{3}$-hanging on ρ^*, and the black curves are shortest paths in $G_{\mathsf{opt},\epsilon}$.

Proof. For each edge $f \in E_{\mathsf{opt},\epsilon}$, let P_f^1 and P_f^2 be the two faces containing f in the planar embedding of G (possibly $P_f^1 = P_f^2$ if f is a cut edge). Define the following potential function $\Phi(f)$ with respect to Γ:

$$\Phi(f) = \begin{cases} 0 & \text{if both } P_f^1 \text{ and } P_f^2 \text{ are contained in some } R_e \text{ where } f \in \gamma_e \in \Gamma, \\ 3\mathbf{w}(f) & \text{if only one of } P_f^1 \text{ and } P_f^2 \text{ is contained in some } R_e \text{ where } f \in \gamma_e \in \Gamma, \\ 6\mathbf{w}(f) & \text{if neither } P_f^1 \text{ nor } P_f^2 \text{ is contained in any } R_e \text{ where } f \in \gamma_e \in \Gamma. \end{cases}$$

Define the following sum as the overall potential (in the sum below, we want to count each f at most once):

$$\Phi \;=\; \mathbf{w}\left(\Gamma\right) + \sum_{e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}}), f \in \gamma_e} \Phi(f).$$

Clearly, we have $\Phi \le 6\mathbf{w}\left(G_{\mathsf{opt},\epsilon}\right)$ at the beginning when $\Gamma = \emptyset$. We show that Φ never increases in the course of the algorithm. Consider any step when we add a certain path γ_e to Γ. Since R_e is enclosed by a simple curve, it contains exactly one face $P_f \in \{P_f^1, P_f^2\}$ for any edge $f \in \gamma_e \setminus \eta$. More importantly, since the algorithm visits all the nodes on $\mathcal{T}$ in a depth-first-search order, P_f was not included in any region $R_{e'}$ before where $f \in \gamma_{e'} \in \Gamma$.

Therefore, $\Phi(f)$ decreases by $3\mathbf{w}(f)$, and so the sum $\sum_{e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}}), f \in \gamma_e} \Phi(f)$ decreases by at least $3 \cdot (\mathbf{w}(\gamma_e) - \mathbf{w}(\eta)) > \mathbf{w}(\gamma_e)$. On the other hand, $\mathbf{w}(\Gamma)$ increases by at most $\mathbf{w}(\gamma_e)$, and so overall Φ deceases. □

Conclusion of the proof. By Claim 4.6, for every edge $e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})$, there exists a path in Γ path on which e can $\frac{2}{3}$-hang. For any path $\rho \in \Gamma$, let $C_\rho \subseteq E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})$ be the set of all edges e which can $\frac{2}{3}$-hang on ρ. By the averaging argument and Claim 4.7, there exists $\rho^* \in \Gamma$ such that

$$\frac{\mathbf{w}(C_{\rho^*})}{\mathbf{w}(\rho^*)} \ge \frac{\sum_{\rho \in \Gamma} \mathbf{w}(C_\rho)}{\sum_{\rho \in \Gamma} \mathbf{w}(\rho)} \ge \frac{\sum_{e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})} \mathbf{w}(e)}{\sum_{\rho \in \Gamma} \mathbf{w}(\rho)} \ge \frac{\sum_{e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})} \mathbf{w}(e)}{6 \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})} = \alpha/6.$$

Set $F^* = C(\rho^*)$. To verify the requirements, let s, t be the two endpoints of ρ^*. By construction, we know that for any $e \in F^*$, the region R_e is contained within region $R_{s,t}$. Also, since all edges in F^* are on the same side of ρ^*, all the sub-path intervals $\{\rho^*[a_e, b_e] : e \in F^*\}$ should form a laminar structure. Check Figure 7 for an illustration □.

Next, our main goal is to show that our dynamic programming finds a path $\rho[s^*, t^*, L^*]$ which would be a good approximation to the path ρ^* in $G_{\mathsf{opt},\epsilon}$, and the multiset $P[s^*, t^*, L^*]$ would also

be a good approximation to the set F^*. The analysis consists of two steps: first, we will show that the multi-set $P[s^*, t^*, L^*]$ contains a large number of edges in terms of edge weights; secondly, since $P[s^*, t^*, L^*]$ is a multi-set, we need to upper bound the amount of duplications so as to prove a true lower bound on $P[s^*, t^*, L^*]$.

4.2.2 Lower Bounding $\mathsf{DP}[s^*, t^*, L^*]$

In this subsection, we show that the multi-set $P[s^*, t^*, L^*]$ is relatively heavy (in terms of α, see Equation (1)) compared to $\rho[s^*, t^*, L^*]$.

Lemma 4.8. $\mathsf{DP}[s^*, t^*, L^*] \geq \frac{\alpha}{6} \cdot \mathbf{w}\left(\rho[s^*, t^*, L^*]\right)$.

Proof. Let ρ^*, F^* be the path and edge sets provided by Lemma 4.5. Let s and t be the two endpoints of the path ρ^* which is unknown to our algorithm, and set $L = \mathbf{w}(\rho^*)$. Since the ratio

$$\mathsf{DP}[s^*, t^*, L^*] / \mathbf{w}\left(\rho[s^*, t^*, L^*]\right)$$

is maximized by the triple (s^*, t^*, L^*), it suffices to show that:

$$\mathsf{DP}[s, t, L] \geq \frac{\alpha}{6} \cdot \mathbf{w}\left(\rho[s, t, L]\right).$$

For every edge $e \in F^*$, let (a_e, b_e) be the hanging points of e on ρ^*. According to the proof of Lemma 4.5, all the sub-path intervals $\{\rho^*[a_e, b_e] : e \in F^*\}$ form a laminar family $\mathcal{L}$. For notational convenience, we could naturally rewrite any sub-path $\rho^*[a, b]$ in an interval manner $[a, b]$. We can view $\mathcal{L}$ as a tree where each node corresponds to an interval $[a, b]$ together with a weight:

$$p_{[a,b]} = \sum_{e \in F^*, \{a_e, b_e\} = \{a, b\}} \mathbf{w}(e). \tag{2}$$

Note that multiple edges $e \in F^*$ might be hanging at the same vertex pair (a, b), which is why we define $p_{[a,b]}$ by a summation. According to Lemma 4.5, we have:

$$\sum_{[a,b] \in \mathcal{L}} p_{[a,b]} = \mathbf{w}(F^*) \geq \frac{\alpha}{6} \cdot \mathbf{w}(\rho^*).$$

Thus, our goal is to show that $\mathsf{DP}[s, t, L]$ is at least the total weight over all nodes in the tree $\mathcal{L}$.

For technical convenience, we will modify $\mathcal{L}$ to make it a binary tree. First, if the root node of $\mathcal{L}$ is not $[s, t]$, then add a root node corresponding to the interval $[s, t]$ with weight $p_{[s,t]} = 0$. In general, while there is a node $[a, b]$ with more than two children $[a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k]$, where $k > 2$, insert an intermediate node $[b_1, b]$ as a child of $[a, b]$ of zero weight $p_{[b_1,b]} = 0$; and move the children $[a_2, b_2], \ldots, [a_k, b_k]$ below $[b_1, b]$.

We will derive a sequence of lower bounds for $\mathsf{DP}[s, t, L]$ using the tree $\mathcal{L}$ and the dynamic programming rules. Intuitively, we will keep unpacking the term $\mathsf{DP}[s, t, L]$ recursively using the dynamic programming rule, and collect all the additive terms of the form $\mathbf{w}(B[\cdot, \cdot])$ that show up during the unpacking procedure, and argue that the total sum of these additive terms is lower bounded by $\mathbf{w}(F^*)$.

To do this, to each node $[a, b]$ of $\mathcal{L}$, we will associate with a value $q_{[a,b]} \geq 0$ such that:

$$\mathsf{DP}\left[a, b, \mathbf{w}\left(\rho^*[a, b]\right)\right] \geq \sum_{[c,d] \in \mathcal{L}[a,b]} q_{[c,d]}, \tag{3}$$

where $\mathcal{L}[a, b]$ denotes the subtree of $\mathcal{L}$ rooted at $[a, b]$. We will define the values in $\{q_{[a,b]} : [a, b] \in \mathcal{L}\}$ as following.

- $[a,b]$ is a leaf node of $\mathcal{L}$. In this case, assign $q_{[a,b]} = \mathbf{w}(B[a,b])$.
- $[a,b]$ has exactly one child $[a_1,b_1]$. In this case, we assign

$$\begin{aligned} q_{[a,b]} =& \mathbf{w}(B[a,a_1]) + \mathbf{w}(B[b_1,b]) \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b]\right) \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a_1,b_1]), \mathbf{w}(\rho^*[a,a_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b_1]\right). \end{aligned}$$

- $[a,b]$ has two children $[a_1,b_1]$ and $[a_2,b_2]$ (where a_1 lies between a and a_2). In this case, we assign

$$\begin{aligned} q_{[a,b]} =& \mathbf{w}(B[a,a_1]) + \mathbf{w}(B[b_1,a_2]) + \mathbf{w}(B[b_2,b]) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\right] \cdot \mathbf{w}\left(B[a,b]\right) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[a,a_1]), \mathbf{w}(\rho^*[a_1,b_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\right] \cdot \mathbf{w}\left(B[a,b_1]\right) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,b_2]), \mathbf{w}(\rho^*[b_2,b])\} < \lfloor \mathbf{w}(\rho^*[b_1,b])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b]\right) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,a_2]), \mathbf{w}(\rho^*[a_2,b_2])\} < \lfloor \mathbf{w}(\rho^*[b_1,b_2])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b_2]\right). \end{aligned}$$

Next, we verify Equation (3) based on definitions of $q_{[\cdot,\cdot]}$'s. This is done in a bottom up manner on the tree $\mathcal{L}$. The inequality holds trivially for leaf nodes by definition of $q_{[\cdot,\cdot]}$. For a non-leaf node $[a,b]$, if it has only one child $[a_1,b_1]$, then according our dynamic programming rule, we have

$$\begin{aligned} \mathsf{DP}[a,b,\mathbf{w}(\rho^*[a,b])] \geq& \mathsf{DP}[a,b_1,\mathbf{w}(\rho^*[a,b_1])] + \mathsf{DP}[b_1,b,\mathbf{w}(\rho^*[b_1,b])] \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b]\right) \\ \geq& \mathsf{DP}[a,a_1,\mathbf{w}(\rho^*[a,a_1])] + \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] + \mathsf{DP}[b_1,b,\mathbf{w}(\rho^*[b_1,b])] \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b]\right) \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a_1,b_1]), \mathbf{w}(\rho^*[a,a_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b_1]\right) \quad . \\ \geq& \mathbf{w}(B[a,a_1]) + \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] + \mathbf{w}(B[b,b_1]) \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b]\right) \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a_1,b_1]), \mathbf{w}(\rho^*[a,a_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b_1]\right) \\ \geq& \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] + q_{[a,b]} \end{aligned}$$

Then by induction, we can conclude Equation (3); here, the third inequality is due to Lemma 4.3, and recall the definition that $\lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2 = 2^k$ such that $\mathbf{w}(\rho^*[a,b]) \in [2^k, 2^{k+1})$.

Next, assume $[a,b]$ has two children $[a_1,b_1]$ and $[a_2,b_2]$ (where a_1 lies between a and a_2). According to our dynamic programming rule, we have

$$\begin{aligned} \mathsf{DP}[a,b,\mathbf{w}(\rho^*[a,b])] \geq& \mathsf{DP}[a,b_1,\mathbf{w}(\rho^*[a,b_1])] + \mathsf{DP}[b_1,b,\mathbf{w}(\rho^*[b_1,b])] \\ &+ \mathbf{1}\big[\max\{\mathbf{w}(\rho^*[a,b_1]), \mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\big] \cdot \mathbf{w}\left(B[a,b]\right). \end{aligned}$$

To further expand the two terms $\mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])], \mathsf{DP}[a_2,b_2,\mathbf{w}(\rho^*[a_2,b_2])]$, we apply again the dynamic programming rules together with Lemma 4.3, we have

$$\begin{aligned} \mathsf{DP}[a,b_1,\mathbf{w}(\rho^*[a,b_1])] \geq& \mathbf{w}(B[a,a_1]) + \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[a,a_1]), \mathbf{w}(\rho^*[a_1,b_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\right] \cdot \mathbf{w}\left(B[a,b_1]\right). \end{aligned}$$

$$\begin{aligned} \mathsf{DP}[b_1,b,\mathbf{w}(\rho^*[b)1,b])] \geq& \mathbf{w}(B[b_1,a_2]) + \mathsf{DP}[a_2,b_2,\mathbf{w}(\rho^*[a_2,b_2])] + \mathbf{w}(B[b_2,b]) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,b_2]), \mathbf{w}(\rho^*[b_2,b])\} < \lfloor \mathbf{w}(\rho^*[b_1,b])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b]\right) \\ &+ \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,a_2]), \mathbf{w}(\rho^*[a_2,b_2])\} < \lfloor \mathbf{w}(\rho^*[b_1,b_2])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b_2]\right). \end{aligned}$$

Summing up the above three inequalities, we have

$$\begin{aligned}
\mathsf{DP}[a,b,\mathbf{w}(\rho^*[a,b])] \geq & \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] + \mathsf{DP}[a_2,b_2,\mathbf{w}(\rho^*[a_2,b_2])] \\
& + \mathbf{w}(B[a,a_1]) + \mathbf{w}(B[b_1,a_2]) + \mathbf{w}(B[b_2,b]) \\
& + \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[a,b_1]),\mathbf{w}(\rho^*[b_1,b])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2\right] \cdot \mathbf{w}\left(B[a,b]\right) \\
& + \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[a,a_1]),\mathbf{w}(\rho^*[a_1,b_1])\} < \lfloor \mathbf{w}(\rho^*[a,b_1])\rfloor_2\right] \cdot \mathbf{w}\left(B[a,b_1]\right) \\
& + \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,b_2]),\mathbf{w}(\rho^*[b_2,b])\} < \lfloor \mathbf{w}(\rho^*[b_1,b])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b]\right) \\
& + \mathbf{1}\left[\max\{\mathbf{w}(\rho^*[b_1,a_2]),\mathbf{w}(\rho^*[a_2,b_2])\} < \lfloor \mathbf{w}(\rho^*[b_1,b_2])\rfloor_2\right] \cdot \mathbf{w}\left(B[b_1,b_2]\right) \\
\geq & \mathsf{DP}[a_1,b_1,\mathbf{w}(\rho^*[a_1,b_1])] + \mathsf{DP}[a_2,b_2,\mathbf{w}(\rho^*[a_2,b_2])] + q_{[a,b]}.
\end{aligned}$$

By induction, we can show that

$$\mathsf{DP}\left[s,t,L\right] \geq \sum_{[a,b]\in\mathcal{L}} q_{[a,b]}.$$

Finally, let us we prove that:

$$\sum_{[a,b]\in\mathcal{L}} q_{[a,b]} \geq \sum_{[a,b]\in\mathcal{L}} p_{[a,b]}.$$

By definition (2), the term $p_{[a,b]}$ is the total weight of edges $e \in F^*$ which are $\frac{2}{3}$-hanging on ρ^* at (a,b), and each $q_{[a,b]}$ is a sum of several terms of the form $\mathbf{w}(B[\cdot,\cdot])$. So the proof strategy is to assign each edge $e \in F^*$ to a set $B[\cdot,\cdot]$ which contains e and contributes to a value $q_{[a,b]}$.

Consider any edge $e \in F^*$ which is $\frac{2}{3}$-hanging at (a,b). Let $[c,d]$ be the lowest descendant of $[a,b]$ in $\mathcal{L}$ such that $\mathbf{w}(\rho^*[c,d]) \geq \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2$. There are several cases to consider.

- $[c,d]$ is a leaf node of $\mathcal{L}$.

  According to the assumption, we know that $\mathsf{dist}_G(c,d) \geq \frac{1}{1+\epsilon}\cdot\mathbf{w}(\rho^*[c,d]) > \frac{1}{2(1+\epsilon)}\cdot\mathbf{w}(\rho^*[a,b])$ and e is $\frac{2}{3}$-hanging at (a,b) on ρ^*. Since c,d both are lying on the sub-path $\rho^*[a,b]$, by Definition 4.2 we know that e is $\frac{1}{3(1+\epsilon)}$-hanging at (c,d) on the shortest path $\pi_{c,d}$, and so $e \in B[c,d]$. By definition, $q_{[c,d]} = \mathbf{w}(B[c,d])$, so we can assign $\mathbf{w}(e)$ to $q_{[c,d]}$.

- $[c,d]$ has one child $[c_1,d_1]$ in $\mathcal{L}$.

  If $\mathbf{w}(B[c,c_1])$ or $\mathbf{w}(B[d,d_1])$ is at least $\lfloor \mathbf{w}[\rho^*[c,d]]\rfloor_2$, then by the same calculation as above, we can argue that e is $\frac{1}{3(1+\epsilon)}$-hanging at one of (c,c_1) or (d_1,d) on shortest path π_{a,a_1} or $\pi_{d_1,d}$, respectively.

  So, let us assume that $\mathbf{w}(B[c,c_1]), \mathbf{w}(B[d_1,d]) < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$. There are two sub-cases to verify.

  - $\mathbf{w}(\rho^*[c,d_1]) < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$.
    In this case, by definition of $q_{[c,d]}$, it concludes the term $\mathbf{w}(B[c,d])$. As $\mathsf{dist}_G(c,d) \geq \frac{1}{1+\epsilon}\cdot\mathbf{w}(\rho^*[c,d]) > \frac{1}{2(1+\epsilon)}\cdot\mathbf{w}(\rho^*[a,b])$, we know that e is $\frac{1}{3(1+\epsilon)}$-hanging at (c,d) on shortest path $\pi_{s,t}$. So we can assign e to $q_{[c,d]}$.
  - $\mathbf{w}(\rho^*[c,d_1]) \geq \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$.
    In this case, since $[c,d]$ is the lowest descendant of $[a,b]$ such that $\mathbf{w}(\rho^*[c,d]) \geq \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2$, we have $\mathbf{w}(\rho^*[c_1,d_1]) < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2 \leq \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$. As we have already assumed $\mathbf{w}(\rho^*[c,c_1]) < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$, $q_{[c,d]}$ contains the term $\mathbf{w}(B[c,d_1])$. By the same calculation, we can show that e is $\frac{1}{3(1+\epsilon)}$-hanging at (c,d_1) and can be assigned to $q_{[c,d]}$.

- $[c,d]$ has two children, $[c_1,d_1]$ and $[c_2,d_2]$, in $\mathcal{L}$ and c_1 lies between c and c_2 on ρ^*.

  By the choice of descendant $[c,d]$, we know that

  $$\max\{\mathbf{w}(\rho^*[c_1,d_1]),\mathbf{w}(\rho^*[c_2,d_2])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor \le \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2.$$

  We can first assume that

  $$\max\{\mathbf{w}(\rho^*[c,c_1]),\mathbf{w}(\rho^*[d_1,c_2]),\mathbf{w}(\rho^*[d_2,d])\} < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2.$$

  since otherwise we could assign e to one of the three edge sets which contribute to $q_{[c,d]}$. Next, we need to discuss several cases.

  - $\max\{\mathbf{w}(\rho^*[c,d_1],\mathbf{w}(\rho^*[d_1,d]))\} < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$.
    In this case, the term $\mathbf{w}(B[c,d])$ would be included in the definition of $q_{[c,d]}$. As before, we can show that e is $\frac{1}{3(1+\epsilon)}$-hanging at (c,d), so we can assign $e \in B[c,d]$ to $q_{[c,d]}$.
  - $\mathbf{w}(\rho^*[c,d_1]) \ge \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$.
    In this case, since we already know

    $$\mathbf{w}(\rho^*[c,c_1]),\mathbf{w}(\rho^*[c_1,d_1]) < \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2 \le \lfloor \mathbf{w}(\rho^*[c,d_1])\rfloor_2,$$

    the term $\mathbf{w}(B[c,d_1])$ should be included in $q_{[c,d]}$. As $\mathbf{w}(\rho^*[c,d_1]) \ge \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$, for the same reason as before we can show $e \in B[c,d_1]$, and so we are able to assign e to $q_{[c,d]}$.
  - $\mathbf{w}(\rho^*[d_1,d]) \ge \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$.
    In this case, if $\mathbf{w}(\rho^*[d_1,d_2]) < \lfloor \mathbf{w}(\rho^*[d_1,d])\rfloor_2$, then the term $\mathbf{w}(B[d_1,d])$ would be included in the definition of $q_{[c,d]}$. As $\mathbf{w}(\rho^*[d_1,d]) \ge \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2 > \frac{1}{2}\mathbf{w}(\rho^*[a,b])$, we know that $e \in B[d_1,d]$, and so we can assign e to $q_{[c,d]}$.
    Otherwise, we have $\mathbf{w}(\rho^*[d_1,d_2]) \ge \lfloor \mathbf{w}(\rho^*[d_1,d])\rfloor_2 \ge \lfloor \mathbf{w}(\rho^*[c,d])\rfloor_2$. Hence, by the choice of $[c,d]$, we must have

    $$\max\{\mathbf{w}(\rho^*[d_1,c_2]),\mathbf{w}(\rho^*[c_2,d_2])\} < \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2 \le \lfloor \mathbf{w}(\rho^*[d_1,d_2])\rfloor_2.$$

    Consequently, $q_{[c,d]}$ contains the term $\mathbf{w}(B[d_1,d_2])$. As $\mathbf{w}(\rho^*[d_1,d_2]) \ge \lfloor \mathbf{w}(\rho^*[a,b])\rfloor_2$, we know that $e \in B[d_1,d_2]$, and so we can assign e to $q_{[c,d]}$.

In this way, we can show that any term $p_{[a.b]} > 0$ can be assigned to some term $q_{[c,d]}$, and so we have

$$\mathsf{DP}[s,t,L] \ge \sum_{[a,b]\in\mathcal{L}} q_{[a,b]} \ge \sum_{[a,b]\in\mathcal{L}} p_{[a,b]} \ge \frac{\alpha}{6}\cdot\mathbf{w}(\rho^*). \quad \square$$

4.2.3 Upper Bounding the Multiplicity of $P[s^*,t^*,L^*]$

Lemma 4.8 implies that the weight of the multi-set $P[s^*,t^*,L^*]$ of edges that are $\frac{1}{3(1+\epsilon)}$-hanging on $\rho[s^*,t^*,L^*]$ is at least $\frac{\alpha}{6}\cdot\mathbf{w}(\rho[s^*,t^*,L^*])$. However, it does not mean we are pruning a lot of weight by updating $F^{\text{old}} \leftarrow F^{\text{old}} \cup P[s^*,t^*,L^*]$ since the multi-set $P[s^*,t^*,L^*]$ might include many edges in $E(H)\setminus(F^{\text{new}}\cup F^{\text{old}})$ with high multiplicity. Next, our main goal is to upper bound the total amount of *over-counting* (that is, the difference between the weight of the multi-set $P[s^*,t^*,L^*]$ and the corresponding (non-multi) set).

Suppose $\rho[s^*,t^*,L^*]$ is a (not necessarily simple) path $\rho = \langle (s^*=)u_1,u_2,\ldots,u_m(=t^*)\rangle$. Build a tree $\mathcal{T}_{\text{dp}}$ according to the maximizers of the dynamic programming table as follows. The tree is built

in a top-down manner. Each node of $\mathcal{T}_{\rm dp}$ is associated with an interval $[i,j] \subseteq [1,m]$. The root node corresponds to interval $[1,m]$. For an arbitrary node $[i,j]$, consider the maximizer for computing the entry $\mathsf{DP}[u_i, u_j, \mathbf{w}(\rho[u_i,u_j])]$. Since $\rho[u_i,u_j]$ is the walk $\langle u_i, u_{i+1}, \ldots, u_j\rangle$, the maximizer for entry $\mathsf{DP}[u_i, u_j, \mathbf{w}(\rho[u_i,u_j])]$ is either a vertex u_k for $i<k<j$, or $\mathsf{DP}[u_i, u_j, \mathbf{w}(\rho[u_i,u_j])]$ is simply equal to $\mathbf{w}(B[u_i,u_j])$. In the former case, we leave $[i,j]$ as a leaf node on $\mathcal{T}_{\rm dp}$; and in the latter case, we create two children of $[i,j]$ associated with $[i,k]$ and $[k,j]$.

Definition 4.9. According to the dynamic programming rules, for any copy of edge e in the multi-set $P[s^*,t^*,L^*]$, it should appear at some tree node $[i,j]$ by set $B[u_i,u_j]$. For convenience, we say that e is *hanging* at the interval $[i,j]$ on ρ.

It is clear that any two different copies of the same edge e in $P[s^*,t^*,L^*]$ should be hanging at distinct intervals since none of the sets $B[x,y]$ is a multi-set. Therefore, any edge $e \in \big(E(H) \setminus (F^{\rm new} \cup F^{\rm old})\big) \cap P[s^*,t^*,L^*]$ is mapped to a set I_e of distinct intervals in $\mathcal{T}_{\rm dp}$.

Lemma 4.10. *For any edge $e \in E(H) \setminus (F^{\rm new} \cup F^{\rm old})$ and any interval $[i,j] \in I_e$, the interval $[i,j]$ has at most 3 ancestors in I_e on $\mathcal{T}_{\rm dp}$.*

Proof. Suppose, for the sake of contradiction, that $[i,j]$ has four different ancestors $[i_1,j_1] \subset [i_2,j_2] \subset [i_3,j_3] \subset [i_4,j_4]$ in the tree $\mathcal{T}_{\rm dp}$ which are all in the set I_e. On the one hand, e should belong to $B[i,j] \cap B[i_1,j_1] \cap B[i_2,j_2] \cap B[i_3,j_3] \cap B[i_4,j_4]$, which implies that e is $\frac{1}{3(1+\epsilon)}$-hanging at all of the pairs $(i,j),(i_1,i_1),(i_2,j_2),(i_3,j_3),(i_4,j_4)$, and therefore we have

$$\begin{aligned}\mathbf{w}(e) &\geq \mathsf{dist}_G(u_{i_4},u_{j_4}) \geq \mathsf{dist}_G(u_{i_3},u_{j_3}) \geq \mathsf{dist}_G(u_{i_2},u_{j_2})\\ &\geq \mathsf{dist}_G(u_{i_1},u_{j_1}) \geq \mathsf{dist}_G(u_i,u_j) \geq \frac{1}{3(1+\epsilon)}\mathbf{w}(e).\end{aligned}$$

However, by our rule of dynamic programming, we know that

$$\begin{aligned}(1+\epsilon)\mathbf{w}(e) &\geq \lfloor \mathbf{w}(\rho[u_{i_4},u_{j_4}])\rfloor_2 > \lfloor \mathbf{w}(\rho[u_{i_3},u_{j_3}])\rfloor_2 > \lfloor \mathbf{w}(\rho[u_{i_2},u_{j_2}])\rfloor_2\\ &> \lfloor \mathbf{w}(\rho[u_{i_1},u_{j_1}])\rfloor_2 > \lfloor \mathbf{w}(\rho[u_i,u_j])\rfloor_2 \geq \frac{1}{6(1+\epsilon)}\mathbf{w}(e).\end{aligned}$$

This is impossible as there cannot be 5 different integral powers of 2 in the interval $\left[\frac{\mathbf{w}(e)}{6(1+\epsilon)}, (1+\epsilon)\mathbf{w}(e)\right]$ when $\epsilon < 0.1$. □

For each $e \in E(H) \setminus (F^{\rm new} \cup F^{\rm old})$, let $J_e \subseteq I_e$ be the set of lowest nodes on $\mathcal{T}_{\rm dp}$. According to Lemma 4.10, we know that $|J_e| \geq \frac{1}{4}|I_e|$, and therefore

$$\sum_{e \in E(H)\setminus(F^{\rm new}\cup F^{\rm old})} |J_e| \cdot \mathbf{w}(e) \geq \frac{1}{4} \sum_{e \in E(H)\setminus(F^{\rm new}\cup F^{\rm old})} |I_e| \cdot \mathbf{w}(e) = \frac{\beta}{4} \cdot \mathbf{w}(\rho) \geq \frac{\alpha}{48} \cdot \mathbf{w}(\rho).$$

Recall that $\beta = \mathsf{DP}[s^*,t^*,L^*]/\mathbf{w}(\rho[s^*,t^*,L^*])$. To lower bound the total weight of edges in $P[s^*,t^*,L^*]$ without any double-counting, it suffices to lower bound the quantity $\sum_{e\in E(H)\setminus F^*} \mathbf{1}[J_e \neq \emptyset] \cdot \mathbf{w}(e)$.

Lower bounding $\sum_{e\in E(H)\setminus(F^{\rm old}\cup F^{\rm new})} \mathbf{1}[J_e \neq \emptyset] \cdot \mathbf{w}(e)$ via shortcuts. Intuitively speaking, if a set J_e is very large, say $|J_e| > 20$, then e is hanging at many different positions simultaneously $[c_1,d_1],[c_2,d_2],\ldots,[c_{20},d_{20}]$ on ρ. Then, $\mathbf{w}(\rho[u_{c_1},u_{d_{20}}])$ would be much larger than $\mathsf{dist}_G(u_{c_1},u_{d_{20}})$, and so we could shortcut the path ρ by replacing the sub-path $\rho[u_{c_1},u_{d_{20}}]$ with the shortest path

between $u_{c_1}, u_{d_{20}}$ and reduce $\mathbf{w}(\rho)$ significantly. Since we knew that ρ was already a $(1+\epsilon)$-approximate shortest path between s and t at the beginning, we could shortcut ρ by at most $\epsilon \cdot \mathsf{dist}_G(s,t)$ in total length, which would upper bound the total amount of multiplicities of J_e's.

Let us now formalize the above idea. We will design a procedure which repeatedly finds shortcuts along ρ and maintains some invariants.

A shortcut structure on ρ.

- A subset $K_e \subseteq J_e$ for all $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$.
- A sequence of disjoint intervals $\mathcal{I} = \{[a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k]\}$ such that $b_i \leq a_{i+1}$ for $1 \leq i < k$.
- A sequence of shortcut (not necessarily simple) paths $\mathcal{P} = \{\eta_1, \eta_2, \ldots, \eta_{k+1}\}$, where η_i is a path in G connecting $u_{b_{i-1}}$ and u_{a_i} such that $\mathbf{w}(\eta_i) < \mathbf{w}(\rho[u_{b_{i-1}}, u_{a_i}])$; conventionally, set $s = u_{b_0}$ and $t = u_{a_{k+1}}$. The paths η_i are the so-called shortcuts.

  Denote $\eta = \eta_1 \circ \rho[u_{a_1}, u_{b_1}] \circ \eta_2 \circ \rho[u_{a_2}, u_{b_2}] \circ \cdots \circ \rho[u_{a_k}, u_{b_k}] \circ \eta_{k+1}$ which is a (not necessarily simple) path between s and t.

We formulate two properties in terms of the above notation.

Invariant 4.11. Our shortcut algorithm will preserve the following properties.

(1) For any $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$, if $|K_e| > 20$, then there exists an interval $[a, b] \in \mathcal{I}$ that contains all $[c, d] \in K_e$.

(2) $\sum_{e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})} |J_e \setminus K_e| \cdot \mathbf{w}(e) \leq 10\beta \cdot (\mathbf{w}(\rho) - \mathbf{w}(\eta))$.

Let us now describe the shortcut algorithm which will only refine the intervals in $\mathcal{I}$. At the beginning, initialize $K_e \leftarrow J_e$ for all $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$, $\mathcal{I} \leftarrow \{[1, m]\}$, and $\eta \leftarrow \rho$. Note that Invariant 4.11 holds initially. Our strategy is to successively decrease the length of η as long as some nonempty set K_e has size larger than 20. We need to define the following notion of *span* for each edge $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that $|K_e| > 20$.

Definition 4.12 (span)**.** For each edge $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that $|K_e| > 20$, by Invariant 4.11(1), there exists an interval $[a, b] \in \mathcal{I}$ which contains all intervals in K_e. Assume $[c_1, d_1], [c_2, d_2], \ldots, [c_l, d_l]$ are all elements of K_e, then define the *span* of e (with respect to the current $\mathcal{P}, \mathcal{I}$) to be $\mathsf{span}(e) = \mathbf{w}(\rho[u_{c_1}, u_{d_l}])$.

Claim 4.13. *For any $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that $|K_e| > 20$, we have $\mathsf{span}(e) > \frac{20}{3(1+\epsilon)} \mathbf{w}(e)$.*

Proof. This is straightforward since e is $\frac{1}{3(1+\epsilon)}$-hanging on ρ at (u_{c_i}, u_{d_i}) and $l > 20$. □

In each iteration of the shortcut procedure, as long as there exists $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that $|K_e| > 20$, let $f \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ be the edge such that $|K_f| > 20$ and $\mathsf{span}(f)$ is *maximized*. Let $[c_1, d_1], [c_2, d_2], \ldots, [c_l, d_l]$ be all the elements in K_f, and we already know $l > 20$. Next, we show how to update $\mathcal{I}$, $\mathcal{P}$ and sets $\{K_e : e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})\}$.

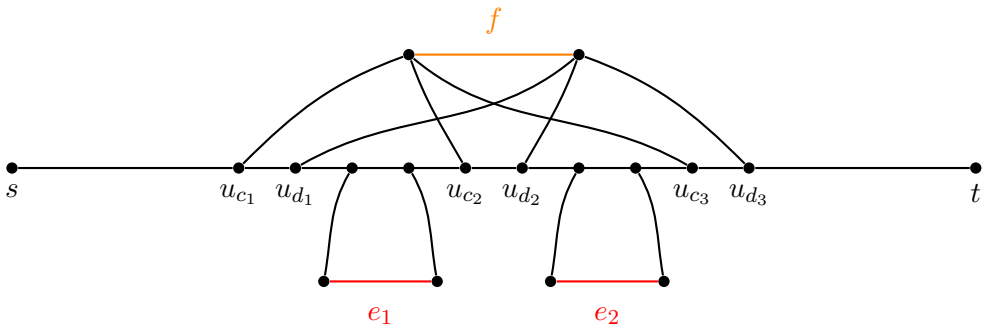


Figure 8: In this example, we find the best edge f maximizing $\mathsf{span}(f)$ such that $|K_f| > 20$. Then, we make a shortcut between u_{c_1} and u_{d_1}, and remove some intervals from K_{e_1} and K_{e_2}.

- Updating $\mathcal{P}, \mathcal{I}$.

  Suppose $f = (x, y)$. By definition of I_f, each f is $\frac{1}{3(1+\epsilon)}$-hanging at vertex pair (u_{c_1}, u_{d_1}) and (u_{c_l}, u_{d_l}). Therefore, we have

  $$\mathsf{dist}_G(x, u_{c_1}) \leq \left(1 - \frac{1}{3(1+\epsilon)}\right) \mathbf{w}(f) < \frac{3}{4}\mathbf{w}(f),$$

  $$\mathsf{dist}_G(y, u_{d_l}) < \left(1 - \frac{1}{3(1+\epsilon)}\right) \mathbf{w}(f) < \frac{3}{4}\mathbf{w}(f).$$

  Hence, by triangle inequality, we have

  $$\mathsf{dist}_G(u_{c_1}, u_{d_l}) < \mathsf{dist}_G(u_{c_1}, x) + \mathbf{w}(f) + \mathsf{dist}_G(y, u_{d_l}) < \frac{5}{2}\mathbf{w}(f).$$

  Let λ be the shortest path in G between u_{c_1} and u_{d_l}. Then, replace $[a, b]$ with $[a, c_1]$ and $[d_l, b]$ in $\mathcal{I}$, and add λ to $\mathcal{P}$ as a new shortcut between c_1 and d_l.

- Updating $\{K_e : e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})\}$.

  Let $[i_1, j_1]$ be the maximal set corresponding to a tree node in $\mathcal{T}_{\text{dp}}$ such that $i_1 < c_1 < j_1$, and there exists an edge $e_1, |K_{e_1}| > 20$ and $[i_1, j_1] \in K_{e_1}$; if no such edge e_1 exists, then simply set $i_1 = j_1 = c_1$. Symmetrically, define $[i_2, j_2]$ to be the maximal interval such that $i_2 < d_l < j_2$, and there exists e_2 such that $|K_{e_2}| > 20$ and $[i_2, j_2] \in K_{e_2}$. Note that since both $[i_1, j_1], [i_2, j_2]$ are tree nodes in $\mathcal{T}_{\text{dp}}$, these two intervals are disjoint internally.

  To update the sets $\{K_e : e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})\}$, for each K_e such that $|K_e| > 20$, if any $[i, j] \in K_e$ satisfies $[i, j] \subseteq [i_1, j_2]$, then remove $[i, j]$ from K_e.

See Figure 8 for an illustration. Next, we show that these updates preserve Invariant 4.11.

Lemma 4.14. *After each iteration of updating $\mathcal{P}$, $\mathcal{I}$, and $\{K_e : e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})\}$, Invariant 4.11 is preserved.*

Proof. Let us first verify Invariant 4.11(1). Consider any edge $e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$ such that $|K_e| > 20$ at the beginning of the iteration. Since Invariant 4.11(1) held before this iteration, there exists $[a', b'] \in \mathcal{I}$ such that $K_e \subseteq [a', b']$. If $[a', b'] \neq [a, b]$, then Invariant 4.11(1) continues to hold for e.

When $[a', b'] = [a, b]$, according to our update rules, for each interval $[i, j] \in K_e$, $|K_e| > 20$, which strictly contains c_1 or d_l, this interval must be contained entirely within $[i_1, j_2]$. Therefore, after the updates, all elements in K_e are contained either in $[a, c_1]$ or $[d_l, b]$. To complement the argument for Invariant 4.11(1), it suffices to show that there cannot be two different elements $[i, j], [i', j'] \in K_e$ such that $[i, j] \subseteq [a, c_1], [i', j'] \subseteq [d_l, b]$. This is because we chose f to be the maximizer of $\mathsf{span}(f)$.

Next, we mainly focus on Invariant 4.11(2). It suffices to upper bound the total amount of edge weight we remove from all the sets K_e by $10\beta\left(\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)\right)$.

To limit the total amount of edge weights that we remove when updating $\{K_e : e \in E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})\}$, we distinguish between three cases for an interval $[i, j]$ that belonged to K_e, $|K_e| > 20$, before the update.

- *Case 1:* $[i, j] \subseteq [i_1, j_1]$. Notice that each such interval $[i, j]$ corresponds to an appearance of e in the multi-set of edges $P\left[u_{i_1}, u_{j_1}, \mathbf{w}(\rho[u_{i_1}, u_{j_1}])\right]$ with total weight $\mathsf{DP}\left[u_{i_1}, u_{j_1}, \mathbf{w}(\rho[u_{i_1}, u_{j_1}])\right]$. Since (s^*, t^*, L^*) is the ratio maximizer, we know that

$$\begin{aligned}\mathsf{DP}\left[u_{i_1}, u_{j_1}, \mathbf{w}(\rho[u_{i_1}, u_{j_1}])\right] &\leq \beta \cdot \mathbf{w}(\rho[u_{i_1}, u_{j_1}]) \leq (1+\epsilon)\beta \cdot \mathbf{w}(e_1) \\ &\leq (1+\epsilon)\beta \cdot \frac{3(1+\epsilon)}{20}\mathsf{span}(e_1) \\ &\leq \frac{1}{5}\beta \cdot \mathsf{span}(f) = \frac{1}{5}\beta \cdot \mathbf{w}(\rho[u_{c_1}, u_{d_l}]).\end{aligned}$$

  The last inequality is due to the selection of f, as f was the edge such that $|K_f| > 20$ with maximum $\mathsf{span}(f)$.

- *Case 2:* $[i, j] \subseteq [i_2, j_2]$. Symmetrically, we can show that the total weight of all such copies of edge e is at most

$$\begin{aligned}\mathsf{DP}\left[u_{i_2}, u_{j_2}, \mathbf{w}(\rho[u_{i_2}, u_{j_2}])\right] &\leq \beta \cdot \mathbf{w}(\rho[u_{i_2}, u_{j_2}]) \leq (1+\epsilon)\beta \cdot \mathbf{w}(e_2) \\ &\leq (1+\epsilon)\beta \cdot \frac{3(1+\epsilon)}{20}\mathsf{span}(e_2) \\ &\leq \frac{1}{5}\beta \cdot \mathsf{span}(f) = \frac{1}{5}\beta \cdot \mathbf{w}(\rho[u_{c_1}, u_{d_l}]).\end{aligned}$$

- *Case 3:* $[i, j] \subseteq [j_1, i_2]$. Let $\mathcal{I}_1$ be the set of all such intervals $[i, j]$. Then, since all such intervals form a laminar family, we can find the set $\mathcal{I}_2 \subseteq \mathcal{I}_1$ of maximal intervals. Assume $\mathcal{I}_2 = \{[p_1, q_1], [p_2, q_2], \ldots, [p_z, q_z]\}$ with $q_i \leq p_{i+1}, 1 \leq i < z$. Then any $[i, j] \subseteq [j_1, i_2]$ which belongs to some set $K_e, |K_e| > 20$ must be contained in an interval $[p, q] \in \mathcal{I}_2$, which corresponds to one copy of e in the multi-set $P\left[u_p, u_q, \mathbf{w}(\rho[u_p, u_q])\right]$. Therefore, the total weight of edges e corresponding to such intervals $[i, j]$ is bounded by

$$\begin{aligned}\sum_{o=1}^{z} \mathsf{DP}\left[u_{p_o}, u_{q_o}, \mathbf{w}(\rho[u_{p_o}, u_{q_o}])\right] &\leq \beta \cdot \sum_{o=1}^{z} \mathbf{w}(\rho[u_{p_o}, u_{q_o}]) \\ &\leq \beta \cdot \mathbf{w}(\rho[u_{j_1}, u_{i_2}]) \\ &\leq \beta \cdot \mathbf{w}(\rho[u_{c_1}, u_{d_l}]) \\ &< 5\beta \cdot \left(\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)\right).\end{aligned}$$

  Here, the first inequality holds because β is the maximum ratio. As for the last inequality, we have

$$\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) \geq \sum_{o=1}^{l} \mathbf{w}(\rho[c_o, d_o]) \geq \frac{20}{3(1+\epsilon)} \cdot \mathbf{w}(f).$$

On the other hand, since $f = (x, y)$ is $\frac{1}{3(1+\epsilon)}$-hanging at both (u_{c_1}, u_{d_1}) and (u_{c_l}, u_{d_l}), by the triangle inequality we get

$$\mathbf{w}(\lambda) \leq \mathsf{dist}_G(u_{c_1}, x) + \mathbf{w}(f) + \mathsf{dist}_G(y, u_{d_l}) \leq (3 + 2\epsilon)\mathbf{w}(f).$$

Hence, we have $\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) \geq \frac{20}{3(1+\epsilon)(3+2\epsilon)}\mathbf{w}(\lambda) > 2\mathbf{w}(\lambda)$, and consequently we have:

$$\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda) \geq \frac{1}{5}\mathbf{w}(\rho[u_{c_1}, u_{d_l}]).$$

By the above case analysis, the total amount of edge weight we remove from all the sets K_e is at most

$$\begin{aligned}
&\frac{2}{5}\beta \cdot \mathbf{w}(\rho[u_{c_1}, u_{d_l}]) + 5\beta \cdot (\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)) \\
&\leq \frac{2}{5}\beta \cdot (\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)) + 5\beta \cdot (\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)) \\
&< 10\beta \left(\mathbf{w}(\rho[u_{c_1}, u_{d_l}]) - \mathbf{w}(\lambda)\right).
\end{aligned}$$

Hence, Invariant 4.11(2) still holds. □

By Lemma 4.14, we can repeatedly update the sets $\mathcal{I}$, $\mathcal{P}$, and $\{K_e : e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})\}$ until all the sets K_e have size at most 20. Then, we can derive a lower bound on $\sum_{e \in E(H) \setminus F^*} \mathbf{1}[J_e \neq \emptyset] \cdot \mathbf{w}(e)$ as follows:

$$\begin{aligned}
\sum_{e \in E(H) \setminus F^*} \mathbf{1}[J_e \neq \emptyset] \cdot \mathbf{w}(e) &\geq \frac{1}{20} \sum_{e \in E(H) \setminus F^*} |K_e| \cdot \mathbf{w}(e) \\
&\geq \frac{1}{20} \left(\sum_{e \in E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})} |J_e| \cdot \mathbf{w}(e) - \sum_{e \in E(H) \setminus F^*} |J_e \setminus K_e| \cdot \mathbf{w}(e) \right) \\
&\geq \frac{1}{20} \cdot \left(\frac{\beta}{4}\mathbf{w}(\rho) - \sum_{e \in E(H) \setminus F^*} |J_e \setminus K_e| \cdot \mathbf{w}(e) \right) \\
&\geq \frac{1}{20} \cdot \left(\frac{\beta}{4}\mathbf{w}(\rho) - 10\beta(\mathbf{w}(\rho) - \mathbf{w}(\eta)) \right) \\
&\geq \frac{1}{20} \cdot \left(\frac{\beta}{4}\mathbf{w}(\rho) - 10\beta \cdot \epsilon \cdot \mathbf{w}(\rho) \right) \\
&> \frac{\beta}{100}\mathbf{w}(\rho) \geq \frac{\alpha}{600}\mathbf{w}(\rho).
\end{aligned}$$

Recall that $\mathbf{w}(\rho) - \mathbf{w}(\eta) \leq (1 + \epsilon)\mathsf{dist}_G(s, t) - \mathsf{dist}_G(s, t) \leq \epsilon \cdot \mathbf{w}(\rho)$ and $\epsilon \leq 10^{-2}$. Therefore, when we update $F^{\mathrm{new}} \leftarrow F^{\mathrm{new}} \cup E(\rho[s^*, t^*, L^*])$ and $F^{\mathrm{old}} \leftarrow F^{\mathrm{old}} \cup P[s^*, t^*, L^*]$, then $\mathbf{w}\left(F^{\mathrm{new}}\right)$ increases by $\mathbf{w}(\rho)$ and F^{old} increases by at least $\frac{\alpha}{600}\mathbf{w}(\rho)$, where $\alpha = \mathbf{w}\left(E(H) \setminus (F^{\mathrm{new}} \cup F^{\mathrm{old}})\right) / \mathbf{w}\left(G_{\mathsf{opt},\epsilon}\right)$. Given this, the following statement helps to verify property (2) of Theorem 4.1.

Lemma 4.15. *Assume $\theta = \mathbf{w}(H)/\mathbf{w}(G_{\mathsf{opt},\epsilon})$. Then, when Algorithm 1 terminates, we have $\mathbf{w}(H_1) \leq O(\log \theta) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$.*

Proof. During the course of Algorithm 1, let $\alpha = \mathbf{w}(H \setminus F^{\mathrm{old}})/\mathbf{w}(G_{\mathsf{opt},\epsilon})$. For any fixed integer k, whenever $\alpha \in [2^k, 2^{k+1})$, each iteration of the while-loop decreases $\mathbf{w}(H \setminus F^{\mathrm{old}})$ by Δ and increases

$\mathbf{w}(F^{\text{new}})$ by at most $\frac{600}{2^k} \cdot \Delta$. Hence, while $\alpha \in [2^k, 2^{k+1})$, the weight $\mathbf{w}(F^{\text{new}})$ could increase by at most $\frac{600}{2^k} \cdot 2^{k+1} \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}) = O(1) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$. Therefore, in the end when $\beta < 1$, we have $\mathbf{w}(H \setminus (F^{\text{new}} \cup F^{\text{old}})) = \alpha \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}) \leq 600\mathbf{w}(G_{\mathsf{opt},\epsilon})$ and $\mathbf{w}(F^{\text{new}}) \leq O(\log \theta) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$, which finishes the proof. □

4.2.4 Stretch Analysis

Let us begin with a basic property of Algorithm 1.

Lemma 4.16. *During Algorithm 1, when an edge $e \in E$ joins F^{new}, then it stays in F^{new} until the end.*

Proof. This is evident because all the multi-sets $P[s,t,L]$ are contained in $E(H) \setminus (F^{\text{new}} \cup F^{\text{old}})$. □

Finally, let us analyze the stretch of graph $H_1 = F^{\text{new}} \cup (H \setminus (F^{\text{new}} \cup F^{\text{old}}))$, proving property (1) of Theorem 4.1.

Lemma 4.17. *The stretch of graph $H_1 = F^{\text{new}} \cup (H \setminus F^{\text{old}})$ is at most $1 + O(1) \cdot \delta$.*

Proof. Consider any pair of vertices $s, t \in V$. By assumption, we have $\mathsf{dist}_H(s,t) \leq (1+\delta) \cdot \mathsf{dist}_G(s,t)$. To bound the distance between s and t in H_1, let us conceptually maintain a short path ρ in $H_1 \cup H$ which is originally the shortest path between s, t in H and then gradually transform it to an st-path in H_1. To analyze the length of ρ, we use a potential function $\Phi(\rho)$ which is the total length of the edges in not in H_1. Initially, $\Phi(\rho)$ is at most $\mathbf{w}(\rho) \leq (1+\delta) \cdot \mathsf{dist}_G(s,t)$.

Let us iteratively update ρ so that ρ eventually belongs to H_1. While there is an edge $(u,v) \in E(\rho)$ not in H_1, by definition, (u,v) must belong to F^{old}. Consider the moment when (u,v) was added to F^{old} by Algorithm 1. According to the algorithm description, at the moment there must exist a path $\gamma_{u,v} \subseteq E(F^{\text{new}})$ (between vertices $x, y \in V$) such that (u,v) is $\frac{1}{3(1+\epsilon)}$-hanging at $\gamma_{u,v}$. Let ρ_1 and ρ_2 be the shortest paths between u, x and v, y in graph H. By Definition 4.2, we have

$$\begin{aligned}\mathbf{w}(\rho_1) + \mathbf{w}(\gamma_{u,v}) + \mathbf{w}(\rho_2) &\leq (1+\delta) \cdot \mathsf{dist}_G(u,x) + \mathbf{w}(\gamma_{u,v}) + (1+\delta) \cdot \mathsf{dist}_G(y,v) \\ &\leq (1+\delta) \cdot ((1+\epsilon)\mathbf{w}(u,v) - \mathbf{w}(\gamma_{u,v})) + \mathbf{w}(\gamma_{u,v}) \\ &\leq (1+\delta)(1+\epsilon) \cdot \mathbf{w}(u,v).\end{aligned}$$

Update $\rho \leftarrow \rho[s,u] \circ \rho_1 \circ \gamma_{u,v} \circ \rho_2 \circ \rho[v,t]$, and so $\mathbf{w}(\rho)$ would increase by at most

$$(1+\delta)(1+\epsilon) \cdot \mathbf{w}(u,v) - \mathbf{w}(u,v) \leq (\delta + \epsilon + \delta\epsilon) \cdot \mathbf{w}(u,v) < 3\delta \cdot \mathbf{w}(u,v).$$

The key point is that all edges on $\gamma_{u,v}$ are in F^{new} at the moment when (u,v) joined F^{old}, and by Lemma 4.16, these edges will stay in F^{new} til the end. Therefore, by replacing ρ with $\rho[s,u] \circ \rho_1 \circ \gamma_{u,v} \circ \rho_2 \circ \rho[v,t]$, the value of $\Phi(\rho)$ has decreased by at least $\mathbf{w}(\gamma_{u,v}) \geq \frac{1}{3(1+\epsilon)}\mathbf{w}(u,v)$, according to Definition 4.2.

As $\Phi(\rho)$ was originally at most $(1+\delta) \cdot \mathsf{dist}_G(s,t)$, the total amount of error increase would be bounded by

$$3\delta \cdot 3(1+\epsilon) \cdot (1+\delta) \cdot \mathsf{dist}_G(s,t) \leq 10\delta \cdot \mathsf{dist}_G(s,t).$$

If follows that $\mathsf{dist}_{H_1}(s,t) \leq (1+11\delta) \cdot \mathsf{dist}_G(s,t)$. □

4.3 Extension to Large Edge Weights

In this subsection, let us discuss how to deal with general edge weights when $W \geq n^2/\epsilon$. Recall that $G = (V, E, \mathbf{w})$ is an undirected weighted planar graph, where $\mathbf{w} : E \to \{1, 2, \ldots, W\}$. We may assume that $W = \max_{e \in E} \mathbf{w}(e)$, and for any edge $(u, v) \in E$, $\mathsf{dist}_G(u, v) = \mathbf{w}(u, v)$, since otherwise we could remove (u, v) from E. Under these assumptions, we know that $\mathbf{w}(G_{\mathsf{opt},\epsilon}) \geq W$, so we could always include all edges with weight less than W/n in a spanner of weight $O(\mathbf{w}(G_{\mathsf{opt},\epsilon}))$.

To compute a spanner, contract all the connected components spanned by edges of weights less than $\epsilon W/n^2$, and denote the contracted graph by $G' = (V', E', \mathbf{w}')$, where we round the edge weights as $\mathbf{w}'(u, v) = \left\lfloor \mathbf{w}(u, v) \cdot \frac{n^2}{W\epsilon} \right\rfloor$. Intuitively, the optimal spanner on G' will be the same as the optimal spanner on the original graph G since the total weight change is small. The advantage of the rounding procedure is that the maximum weight is now polynomial in n, and so the runtime would also be polynomial, according to the main algorithm.

Technically speaking, by definition of G', edge weights $\mathbf{w}'$ take integer values in $[1, n^2/\epsilon]$. Then, apply the main algorithm on graph G' to obtain a $(1 + \epsilon_0)$-spanner H' in time $\mathrm{poly}(n, \epsilon^{-1})$ such that

$$\mathbf{w}'(H') \leq C \cdot \mathbf{w}'(G'_{\mathsf{opt},2\epsilon}),$$

where $C = O(1)$, $\epsilon_0 = \epsilon \cdot 2^{O(\log^* 1/\epsilon)}$ and $G'_{\mathsf{opt},2\epsilon}$ is the optimal $(1 + 2\epsilon)$-spanner of G'. In the end, define

$$E_0 = \{e \in E : \mathbf{w}(e) \leq W/n\}$$

and return

$$H = H' \cup E_0$$

as the approximate spanner of G. Let us verify the stretch and weight of H below.

Lemma 4.18. *For any $(s, t) \in E$, we have $\mathsf{dist}_H(s, t) \leq (1 + \epsilon_0 + 2\epsilon) \cdot \mathbf{w}(s, t)$.*

Proof. If $\mathbf{w}(s, t) \leq W/n$, then by construction $(s, t) \in E(H)$, so $\mathsf{dist}_H(s, t) = \mathbf{w}(s, t)$. Otherwise, we may assume $\mathbf{w}(s, t) > W/n$. Since H' is a $(1 + \epsilon_0)$-spanner of G', we have

$$\mathsf{dist}_{H'}(s, t) \leq (1 + \epsilon_0)\mathbf{w}'(s, t) \leq (1 + \epsilon_0)\mathbf{w}(s, t) \cdot \frac{n^2}{W\epsilon}.$$

Let π' be the shortest path between s and t in the contracted graph H'. Unpack all the contracted nodes in G', then π' expands to a sequence of edges $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k) \in E(H)$, where v_{i-1}, u_i are in the same contracted node for all $1 \leq i \leq k + 1$ ($s = v_0, t = u_{k+1}$). Since H includes all edges whose weights are at most W/n under $\mathbf{w}$, the unpacking increases the distance between s and t by at most $\sum_{i=1}^{k+1} \mathsf{dist}_H(v_{i-1}, u_i) < n \cdot \frac{W\epsilon}{n^2} = \frac{W\epsilon}{n}$. Therefore, overall we have

$$\begin{aligned}\mathsf{dist}_H(s, t) &\leq \frac{W\epsilon}{n} + \sum_{i=1}^{k} \mathbf{w}(u_i, v_i) \leq \frac{W\epsilon}{n} + \sum_{i=1}^{k} \frac{W\epsilon}{n^2} \cdot (\mathbf{w}'(u_i, v_i) + 1) \\ &< (1 + \epsilon_0)\mathbf{w}(s, t) + \frac{2W\epsilon}{n} < (1 + \epsilon_0 + 2\epsilon)\mathbf{w}(s, t). \quad \square\end{aligned}$$

Lemma 4.19. $\mathbf{w}(H) \leq (4C + 4) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$.

Proof. Define $F = E(G_{\mathsf{opt},\epsilon}) \cup E_0$, and let $\widehat{G}$ be the graph obtained from (V, F) by contracting all edges in F whose weights are less than $\epsilon W/n^2$.

We claim that $\widehat{G}$ is a $(1+2\epsilon)$-spanner of G'. In fact, for any edge $(s,t) \in E$ such that $\mathbf{w}(s,t) > W/n$, we have

$$\begin{aligned}\mathsf{dist}_{\widehat{G}}(s,t) &\leq \mathsf{dist}_{G_{\mathsf{opt},\epsilon}}(s,t) \cdot \frac{n^2}{W\epsilon} \leq (1+\epsilon)\mathbf{w}(s,t) \cdot \frac{n^2}{W\epsilon} \\ &< (1+\epsilon) \cdot (\mathbf{w}'(s,t)+1) \leq (1+2\epsilon) \cdot \mathbf{w}'(s,t).\end{aligned}$$

The last inequality holds because $\mathbf{w}(s,t) > W/n$ and $\mathbf{w}'(s,t) = \left\lfloor \mathbf{w}(s,t) \cdot \frac{n^2}{W\epsilon} \right\rfloor \geq \lfloor n/\epsilon \rfloor \geq 1+\epsilon^{-1}$.

As $\widehat{G}$ is a $(1+2\epsilon)$-spanner of G', we have

$$\begin{aligned}\frac{n^2}{W\epsilon} \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}) + \mathbf{w}'(E_0) &\geq \mathbf{w}'(\widehat{G}) \geq \mathbf{w}'(G'_{\mathsf{opt},2\epsilon}) \geq \frac{1}{C} \cdot \mathbf{w}'(H') \\ &\geq \frac{1}{C} \cdot \big(\mathbf{w}'(H \setminus E_0)\big) \\ &\geq \frac{1}{C}\big(\mathbf{w}'(H) - \mathbf{w}'(E_0)\big) \\ &\geq \frac{1}{C}\left(\frac{n^2}{W\epsilon}\left(\mathbf{w}(H) - |E(H)|\right)\right) - \frac{1}{C}\mathbf{w}'(E_0).\end{aligned}$$

Rearranging the terms and using $|E(H)| \leq |E| < 3n$ and $3n \leq W \leq \mathbf{w}(G_{\mathsf{opt},\epsilon})$, we obtain

$$\begin{aligned}\mathbf{w}(H) &< C \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}) + (C+1)\frac{W\epsilon}{n^2} \cdot \mathbf{w}'(E_0) + 3n \\ &\leq C \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}) + (C+1) \cdot \frac{W}{n} \cdot |E_0| + 3n \\ &< (4C+4) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon}). \quad \square\end{aligned}$$

5 Hardness for Planar Spanners

In this section, we prove Theorem 1.3. For simplicity of notation, we assume that $\epsilon > 0$ is rational. The integer edge weight condition is achieved through proper scaling.

3SAT. Given a set X of n Boolean variables $x_1, x_2, \ldots, x_n$ and a Boolean formula ϕ in conjunctive normal form, where each clause has at most 3 literals, the 3SAT problem is to determine whether there is an assignment of `true` or `false` values to the variables such that ϕ is satisfied (i.e., evaluates true).

Incidence graph. Given a 3SAT instance $\mathcal{I}$ with variable set X and a Boolean formula $\phi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, the *incidence graph* $G = G(\mathcal{I})$ corresponding to $\mathcal{I}$ is a bipartite graph with partite sets corresponding to all variables in X and all clauses in ϕ; there is an undirected edge (x_i, c_j) in G if and only if the clause c_j contains x_i or $\neg x_i$. The edge (x_i, c_j) is a *positive edge* if c_j contains x_i and is a *negative edge* if c_j contains $\neg x_i$. For each clause c_j, let $|c_j|$ be the number of variables in c_j. For each variable x_i, let C_i^+ be the set of clauses containing x_i and C_i^- be the set of clauses containing $\neg x_i$.

Planar 3SAT. A 3SAT instance $\mathcal{I}$ is *planar* if its incidence graph G is planar. Furthermore, $\mathcal{I}$ is *planar rectilinear* if there exists a planar representation of G where the vertices are represented by horizontal line segments (specifically, the vertices representing variables are on the x-axis while

the segments representing clauses are above or below the x-axis); and the edges are represented by vertical segments. An instance $\mathcal{I}$ is *planar rectilinear monotone* if all edges above the x-axis are positive and all edges below are negative (see an example in Figure 9).

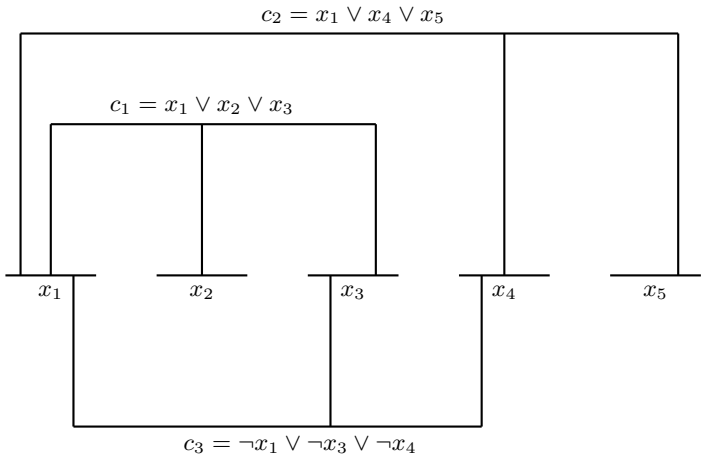


Figure 9: A planar representation of a planar rectilinear monotone 3SAT instance.

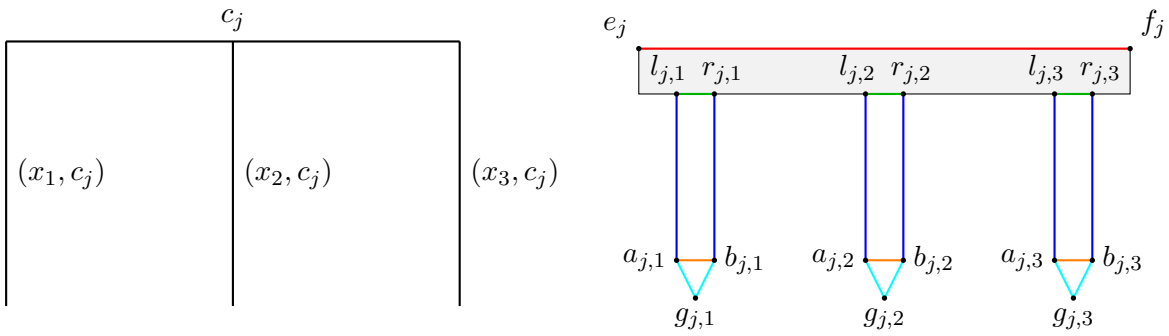
Theorem 5.1 ([dBK12])**.** *Planar rectilinear monotone 3SAT is NP-hard.*

Weight-k planar $(1+\epsilon)$-spanner. Given a planar graph $G = (V, E)$, is there a $(1+\epsilon)$-spanner of G of weight k? In this section, we reduce the weight-k planar $(1+\epsilon)$-spanner problem to the planar rectilinear monotone 3SAT.

Reduction. Given an instance $\mathcal{I}$ of planar rectilinear monotone 3SAT. Let G' be the planar rectilinear drawing of the incidence graph of $\mathcal{I}$. We construct an edge-weighted planar graph G such that G has a $(1+\epsilon)$-spanner of weight at most k (for some value of k defined later) if and only if $\mathcal{I}$ is satisfiable. We can assume that each variable x appears in both positive and negative clauses, since otherwise we can assign a `true`/`false` value to x according to a clause it appears in and delete all clauses containing x or $\neg x$.

Clause gadget. If clause c_j has 3 literals, we assume w.l.o.g. that $c_j = x_1 \wedge x_2 \wedge x_3$. In the drawing of G', we assume that the edges corresponding to (x_1, c_j), (x_2, c_j) and (x_3, c_j) appear from left to right. We then replace each clause segment with the gadget in Figure 10b. Formally, we create a cycle $(e_j, l_{j,1}, r_{j,1}, l_{j,2}, r_{j,2}, l_{j,3}, r_{j,3}, f_j)$. For $h \in \{1, 2, 3\}$, the vertices $l_{j,h}$ and $r_{j,h}$ are connected to a gadget of the literal x_h, which will be discussed later, via two edges $(l_{j,h}, a_{j,h})$ and $(r_{j,h}, b_{j,h})$. The order of those edges is shown in Figure 10b. The weight of each edge $(l_{j,h}, r_{j,h})$ is $2 + 2\epsilon$. We set the weight of each edge $(l_{j,h}, a_{j,h})$ and $(r_{j,h}, b_{j,h})$ to be ϵ. Set the weight of (e_j, f_j) to $\frac{6+10\epsilon}{1+\epsilon}$. All other edges have weight 0. If clause c_j contains two literals, say x_1 and x_2, then we construct the graph of the clause similar to the case with three literals, but without $l_{j,3}$ and $r_{j,3}$. The weight of $(l_{j,1}, r_{j,1})$ and $(l_{j,2}, r_{j,2})$ remains unchanged, while the weight of (e_j, f_j) is $\frac{4+6\epsilon}{1+\epsilon}$.

Literal gadget. We now construct the gadget for each literal x_i. Let $h_i = \max\{|C_i^+|, |C_i^-|\}$. Assume w.l.o.g. that $|C_i^+| \leq |C_i^-|$. The gadget contains two disjoint paths from a vertex s_i to a vertex t_i, each path has length $4h_i$. The two paths enclose a region. Inside the region, there is an



(a) Original Clause

(b) Clause Graph

Figure 10: The clause graph of $c_j = x_1 \vee x_2 \vee x_3$

edge (s_i, t_i) of weight $4h_i/(1+\epsilon)$. For each clause c_j, recall that $l_{j,i}, r_{j,i}$ are connected to the gadget of x_i via two edges, and let $a_{j,i}, b_{j,i}$ be the other endpoints of those edges. We set $\mathbf{w}(a_{j,i}, b_{j,i}) = 2$ and create a vertex $g_{j,i}$ inside the enclosed region of the two paths from s_i to t_i that is adjacent to $a_{j,i}$ and $b_{j,i}$. We set the weight of two edges $(g_{j,i}, a_{j,i})$ and $(g_{j,i}, b_{j,i})$ to be $1+\epsilon$. We arrange all edges $(a_{j,i}, b_{j,i})$ corresponding to positive clauses on the upper path from s_i to t_i and all edges corresponding to negative clauses on the lower path as in Figure 11. From left to right, the order of each $(a_{j,i}, b_{j,i})$ added to the path from s_i to t_i is the same as the order of their corresponding edges in the drawing of G'. If c_j contains x_i, we call the edge $(a_{j,i}, b_{j,i})$ a true edge. Otherwise, $(a_{j,i}, b_{j,i})$ is a false edge. We append an edge of weight $2h_i$ as the last non-zero weight edge in both the upper and lower paths from s_i to t_i. Call these edges (u_i, u_i') and (v_i, v_i').

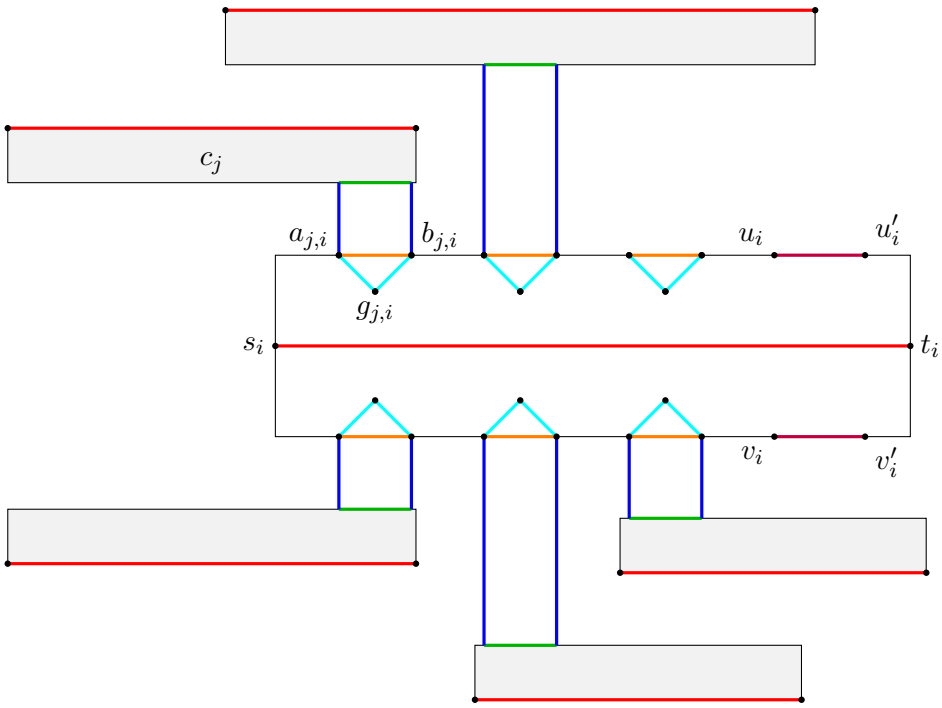


Figure 11: Literal Gadget

If the number of positive clauses is strictly less than the number of negative clauses, we add $|C_i^-| - |C_i^+|$ true edges, each of which does not connect to any clause gadgets in the upper path (see Figure 11). We connect true edges to form a path from s_i to t_i of length $4h_i$, that is, the edges connecting positive/negative edges have weight 0. We use $G[x_i]$ to denote the gadget corresponding to x_i and $G[c_j]$ for the gadget corresponding to a clause c_j.

The construction of G takes polynomial time of the total number of clauses and literals in $\mathcal{I}$. The graph G is planar by the planarity of G'. Let $W = 2\epsilon \sum_j |c_j| + 2(5+2\epsilon) \cdot \sum_i h_i$. We prove the following lemma:

Lemma 5.2. *$\mathcal{I}$ is satisfiable if and only if G has a spanner of total weight at most W.*

For the necessity, assume that $\mathcal{I}$ admits a satisfying truth assignment for the Boolean variables $x_1, x_2, \ldots, x_n$. We then construct a $(1+\epsilon)$-spanner of G of weight at most W.

Lemma 5.3. *If $\mathcal{I}$ is satisfiable, then G has a $(1+\epsilon)$-spanner of total weight at most W.*

Proof. Consider a truth assignment satisfying ϕ. We initialize H to be a subgraph containing all weight-0 edges in G. For each variable x_i, if $x_i = \texttt{true}$, we add all true edges in the literal gadget corresponding to x_i to H. Otherwise, we add all false edges in $G[x_i]$ to H. For each true or false edge $(a_{j,i}, b_{j,i})$, we also add $(a_{j,i}, g_{j,i})$ and $(b_{j,i}, g_{j,i})$ to H.

Then, for each clause c_j containing a variable x_i, we add the edges $(l_{j,i}, a_{j,i})$ and $(r_{j,i}, b_{j,i})$ (blue and green edges in Figure 11).

We show that H is a $(1+\epsilon)$-spanner of G, that is, for every edge (u, v) in G, we have $\mathsf{dist}_H(u, v) \leq (1+\epsilon)\mathbf{w}(u, v)$. Since H contains all edges of weight 0, we only need to consider the case when $\mathbf{w}(u, v) > 0$.

By our construction, we add all edges connecting clause gadgets to literal gadgets. Thus, we focus on the case that (u, v) is either in a clause gadget or in a literal gadget.

Let $c_j = x_1 \vee x_2 \vee x_3$ be a clause with the clause gadget of c_j as in Figure 10b. The case when c_j contains only two literals can be solved similarly. Consider the edge $(l_{j,1}, r_{j,1})$. If $x_1 = \texttt{true}$, we have

$$\mathsf{dist}_H(l_{j,1}, r_{j,1}) = \mathbf{w}(l_{j,1}, a_{j,1}) + \mathbf{w}(a_{j,1}, b_{j,1}) + \mathbf{w}(b_{j,1}, r_{j,1}) = 2 + 2\epsilon. \tag{4}$$

Else if $x_1 = \texttt{false}$, then

$$\mathsf{dist}_H(l_{j,1}, r_{j,1}) = \mathbf{w}(l_{j,1}, a_{j,1}) + \mathbf{w}(a_{j,1}, g_{j,1}) + \mathbf{w}(g_{j,1}, b_{j,1}) + \mathbf{w}(b_{j,1}, r_{j,1}) = 2 + 4\epsilon \leq (1+\epsilon)\mathbf{w}(l_{j,1}, r_{j,1}). \tag{5}$$

In both cases, the distances between $(l_{j,1}, r_{j,1})$ is preserved up to $(1+\epsilon)$-factor in H. Using a similar argument, the distance between $(l_{j,2}, r_{j,2})$ and $(l_{j,3}, r_{j,3})$ is also preserved. We now prove that $\mathsf{dist}_H(e_j, f_j) \leq (1+\epsilon)\mathbf{w}(e_j, f_j)$. Recall that we do not add the edge (e_j, f_j) to H. We have

$$\begin{aligned}\mathsf{dist}_H(e_j, f_j) =& \mathbf{w}(e_j, l_{j,1}) + \mathsf{dist}_H(l_{j,1}, r_{j,1}) + \mathbf{w}(r_{j,1}, l_{j,2}) + \mathsf{dist}_H(l_{j,2}, r_{j,2}) \\ &+ \mathbf{w}(r_{j,2}, l_{j,3}) + \mathsf{dist}_H(l_{j,3}, r_{j,3}) + \mathbf{w}(r_{j,3}, f_j) \\ =& \mathsf{dist}_H(l_{j,1}, r_{j,1}) + \mathsf{dist}_H(l_{j,2}, r_{j,2}) + \mathsf{dist}_H(l_{j,3}, r_{j,3}).\end{aligned}$$

Observe that Equations (4) and (5) yields $\mathsf{dist}_H(l_{j,1}, r_{j,1}), \mathsf{dist}_H(l_{j,2}, r_{j,2}), \mathsf{dist}_H(l_{j,3}, r_{j,3}) \in \{2 + 2\epsilon, 2 + 4\epsilon\}$. Since there is at least one literal in $\{x_1, x_2, x_3\}$ is true, there is at least one value, say $\mathsf{dist}_H(l_{j,1}, r_{j,1})$, equal to $2 + 2\epsilon$. Thus,

$$\begin{aligned}\mathsf{dist}_H(e_j, f_j) &= \mathsf{dist}_H(l_{j,1}, r_{j,1}) + \mathsf{dist}_H(l_{j,2}, r_{j,2}) + \mathsf{dist}_H(l_{j,3}, r_{j,3}) \\ &\leq 2 + 2\epsilon + 2 \cdot (2 + 4\epsilon) = 6 + 10\epsilon = (1+\epsilon)\mathbf{w}(e_j, f_j).\end{aligned}$$

For each literal gadget, since there is a path from s_i to t_i of total weight $4h_i$ containing only true or false edges, the distance between s_i and t_i in H is $4h_i$, and hence is $(1+\epsilon)$ times the distance in G which is $\mathbf{w}(s_i, t_i) = 4h_i/(1+\epsilon)$. For every edge $(a_{j,i}, b_{j,i})$ in G, since we add $(a_{j,i}, c_{j,i})$ and $(b_{j,i}, g_{j,i})$ to H, $\mathsf{dist}_H(a_{j,i}, b_{j,i}) \leq \mathbf{w}(a_{j,i}, g_{j,i}) + \mathbf{w}(b_{j,i}, g_{j,i}) = 2 + 2\epsilon = (1+\epsilon)\mathsf{dist}_G(a_{j,i}, b_{j,i})$.

We now analyze the total weight of H. We distinguish between four types of edges in H:

- $(l_{j,i}, a_{j,i})$ and $(r_{j,i}, b_{j,i})$: Each of these edges has weight ϵ. The total weight of those edges is $2\epsilon \cdot \sum_j |c_j|$.
- $(a_{j,i}, c_{j,i})$ and $(b_{j,i}, c_{j,i})$: Each literal gadget of x_i has $4\max\{|C_i^+|, |C_i^-|\}$ edges of this type. The total weight of these edges is $\sum_i (1+\epsilon) \cdot 4\max\{|C_i^+|, |C_i^-|\}$.
- (u_i, u_i') and (v_i, v_i'): Each literal gadget for x_i has 2 edges of this type. The total weight is then $4 \cdot \max\{|C_i^+|, |C_i^-|\}$.
- $(a_{j,i}, b_{j,i})$: H contains exactly $\max\{|C_i^+|, |C_i^-|\}$ edges of this type in each literal gadget. The total weight of this type is $\sum_i 2\max\{|C_i^+|, |C_i^-|\}$.

The total weight of H is

$$2\epsilon \cdot \sum_j |c_j| + \sum_i (1+\epsilon) \cdot 4\max\{|C_i^+|, |C_i^-|\} + 4 \cdot \max\{|C_i^+|, |C_i^-|\} + \sum_i 2 \cdot \max\{|C_i^+|, |C_i^-|\} = W,$$

as desired. □

We now show the converse direction.

Lemma 5.4. *If G has a $(1+\epsilon)$-spanner of total weight at most W, then $\mathcal{I}$ is satisfiable.*

Observe that if G has a spanner of total weight W, then there exists a $(1+\epsilon)$-spanner of G with the same total weight and containing all weight-0 edges. Hence, we can assume that any spanner contains all weight-0 edges.

Let H be a $(1+\epsilon)$-spanner of G containing all weight-0 edges. We show that H has weight at most W only if $\mathcal{I}$ is satisfiable.

Observation 5.5. *H contains all edges $(l_{j,i}, a_{j,i})$, $(r_{j,i}, b_{j,i})$, $(a_{j,i}, g_{j,i})$, $(b_{j,i}, g_{j,i})$, (u_i, u_i'), and (v_i, v_i').*

Recall that $h_i = \max\{|C_j^+|, |C_j^-|\}$. We then show that each literal gadget has bounded weight.

Lemma 5.6. *For every literal x_i, every $(1+\epsilon)$-spanner H of G must contain a subgraph of $G[x_i]$ of weight at least $2h_i \cdot (5+2\epsilon)$.*

Proof. Consider the gadget for x_i as in Figure 11. Let $H[x_i]$ be the intersection of H with $G[x_i]$. By Observation 5.5, for every clause c_j containing either x_i or $\neg x_i$, $H[x_i]$ must contain both edges $(a_{j,i}, g_{j,i})$ and $(b_{j,i}, g_{j,i})$. Furthermore, $H[x_i]$ also contains (u_i, u_i') and (v_i, v_i'). The total weight of these edges is $4h_i + 2h_i \cdot (2+2\epsilon) = 4h_i \cdot (2+\epsilon)$.

If $H[x_i]$ contains (s_i, t_i), then the total weight of $H[x_i]$ is at least $4h_i/(1+\epsilon) + 4h_i \cdot (2+\epsilon) = 4h_i\left(2+\epsilon+\frac{1}{1+\epsilon}\right) > 2h_i \cdot (5+2\epsilon)$ given $\epsilon < 1$.

If $H[x_i]$ does not contain (s_i, t_i), then the only $(1+\epsilon)$ path between s_i and t_i is either the path containing all true edges or the path containing all false edges in $G[x_i]$. Note that the total weight of true (resp., false) edges is $2h_i$. Hence, the total weight of $H[x_i]$ is at least $4h_i \cdot (2+\epsilon) + 2h_i = 2h_i(5+2\epsilon)$, as claimed. □

We can now prove Lemma 5.4.

Proof of Lemma 5.4. Since H must contain both edges $(l_{j,i}, a_{j,i})$ and $(r_{j,i}, b_{j,i})$ for all pairs (j, i) when those edges exist, we have:

$$\begin{aligned}\mathbf{w}(H) &= 2\epsilon \cdot \sum_j |c_j| + \sum_i (\mathbf{w}(H[x_i])) \\ &\geq 2\epsilon \cdot \sum_j |c_j| + 2(5+2\epsilon) \cdot \sum_i h_i \qquad \text{(by Lemma 5.6)} \\ &= W.\end{aligned} \tag{6}$$

Equality holds if and only if H does not contain any edge of positive weight in $G[c_j]$ for all j and the weight of each $H[x_i]$ achieves the minimum value $2h_i(5+2\epsilon)$ for all i. For the weight of each $H[x_i]$ to be $2h_i(5+2\epsilon)$, $H[x_i]$ does not contain the edge (s_i, t_i), but rather a path of all true or false edges from s_i to t_i. Observe that $H[x_i]$ cannot contain both true and false edges at the same time (otherwise the total weight is not minimum). Thus, if $H[x_i]$ contains a path of true edges from s_i to t_i, we set $x_i = \texttt{true}$, otherwise, we set $x_i = \texttt{false}$. We show that by this assignment, every clause is satisfied.

Consider a clause $c_j = x_1 \vee x_2 \vee x_3$ with $G[c_j]$ in Figure 10b. Since H is a spanner of G, $\mathsf{dist}_H(e_j, f_j) \leq (1+\epsilon)\mathbf{w}(e_i, f_i) = 6 + 10\epsilon$. Given that H does not contain any edge in $G[c_j]$ of positive weight, we prove that $\mathsf{dist}_H(l_{j,1}, r_{j,1}) \in \{2+2\epsilon, 2+4\epsilon\}$. Observe that there are only two cases for the shortest path P from $l_{j,1}$ to $r_{j,1}$ in H:

- P contains $(a_{j,1}, b_{j,1})$. In this case, $\mathsf{dist}_H(l_{j,1}, r_{j,1}) = 2\epsilon + \mathbf{w}(a_{j,1}, b_{j,1}) = 2\epsilon + 2$.
- P contains $(a_{j,1}, c_{j,1})$ and $(b_{j,1}, c_{j,1})$. In this case, $\mathsf{dist}_H(l_{j,1}, r_{j,1}) = 2\epsilon + \mathbf{w}(a_{j,1}, c_{j,1}) + \mathbf{w}(b_{j,1}, c_{j,1}) = 2 + 4\epsilon$.

Similarly, we obtain the same result for $\mathsf{dist}_H(l_{j,2}, r_{j,2})$ and $\mathsf{dist}_H(l_{j,3}, r_{j,3})$. On the other hand, observe that $\mathsf{dist}_H(e_j, f_j) = \mathsf{dist}_H(l_{j,1}, r_{j,1}) + \mathsf{dist}_H(l_{j,2}, r_{j,2}) + \mathsf{dist}_H(l_{j,3}, r_{j,3})$ as all other edges in the path from e_j to f_j have weight 0. If $\mathsf{dist}_H(l_{j,1}, r_{j,1}) = \mathsf{dist}_H(l_{j,2}, r_{j,2}) = \mathsf{dist}_H(l_{j,3}, r_{j,3}) = 2 + 4\epsilon$, $\mathsf{dist}_H(e_i, f_i) = 6 + 12\epsilon > (1+\epsilon)\mathbf{w}(e_i, f_i) = 6 + 10\epsilon$. Then, at least one of the edges among $(a_{j,1}, b_{j,1}), (a_{j,2}, b_{j,2}), (a_{j,3}, b_{j,3})$ is present in H. Since if one edge, say $(a_{j,1}, b_{j,1})$, appears in H, we set x_1 to be $\texttt{true}$ if $(a_{j,1}, b_{j,1})$ is a true edge and false otherwise. In both cases, c_j is satisfied. □

6 A Hard Instance for the Greedy Algorithm

In this section, we adapt the hard instance for Euclidean spaces of [LST+24] to give a hard instance for planar graphs where greedy spanners have heavy total weight compared with optimal $(1+\epsilon)$-spanners, even when relaxing the stretch of the greedy spanner to be $1 + x\epsilon$ for some large $x \gg 1$. Recall that the greedy t-spanner H for an edge-weighted graph $G = (V, E, \mathbf{w})$ is constructed as follows [ADD+93]: Initialize an empty graph $H = (V, \emptyset)$, and then consider the edges $(u, v) \in E$ sorted by nondecreasing weight (ties are broken arbitrarily), and if $\mathsf{dist}_H(u, v) > t \cdot \mathsf{dist}_G(u, v)$ in the current spanner H, then add (u, v) to H.

Theorem 6.1. *For every $\epsilon \in (0, \frac{1}{4})$ and every parameter $1 \leq x \leq \frac{1}{2}\sqrt{1/\epsilon}$, there exists an undirected planar graph $G = (V, E, \mathbf{w})$ with positive edge weights such that $\mathbf{w}(G_{\mathrm{gr}(x)}) \geq \Omega\left(\frac{\epsilon^{-1}}{x^2}\right) \cdot \mathbf{w}(G_{\mathsf{opt},\epsilon})$, where $G_{\mathrm{gr}(x)}$ is the greedy $(1+x\epsilon)$-spanner of graph G.*

Proof. This construction is adapted from a Euclidean example in [LST+24]. The graph G is built as following.

- **Vertices.** Let $V = \{w\} \cup \{x_1, x_2, \ldots, x_n\} \cup \{y_1, y_2, \ldots, y_n\}$ be a vertex set with $2n+1$ vertices, where $n = 1 + \lceil \frac{1}{x} + \frac{\epsilon^{-1}}{2x^2} \rceil$.
- **Edges.** For each $1 \le i \le n$, add edge (x_i, y_i) with weight 1. Connect all vertices via a path $\pi = \langle x_1, x_2, \ldots, x_n, w, y_1, y_2, \ldots, y_n \rangle$. For each $1 \le i < n$, set the weight of (x_i, x_{i+1}) and (y_i, y_{i+1}) to be $x\epsilon$, and add edge (x_n, y_1) with weight

$$\mathbf{w}(x_n, y_1) = 1 + \epsilon - (n-1)x\epsilon \in \left(1 - \frac{1}{2x} - x\epsilon, 1 - \frac{1}{2x}\right].$$

  Finally, set the weights of (x_n, w) and (w, y_1) to be $\frac{1}{2} \cdot (1 + x\epsilon) \cdot \mathbf{w}(x_n, y_1)$.

One can verify that G is a planar graph; check Figure 12 for a planar drawing.

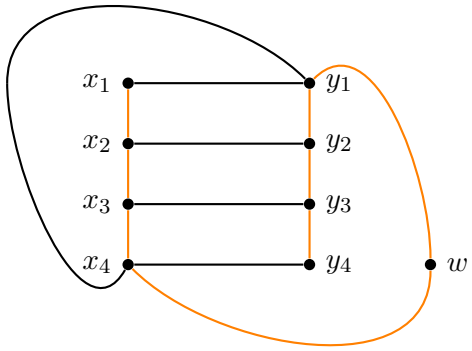


Figure 12: A planar drawing of the hard example against the greedy algorithm when $n = 4$. The path π is highlighted as the orange path.

First, we study the total weight of $G_{\mathsf{opt},\epsilon}$. Consider the graph $H = \pi \cup \{(x_n, y_1)\}$. Then, for any edge (x_i, y_i), by definition of n, we have

$$\mathsf{dist}_H(x_i, y_i) \le (n-1)x\epsilon + \mathbf{w}(x_n, y_1) \le 1 + \epsilon = (1+\epsilon)\mathbf{w}(x_i, y_i).$$

Therefore, H is a $(1+\epsilon)$-spanner, which implies:

$$\begin{aligned}
\mathbf{w}(G_{\mathsf{opt},\epsilon}) \le \mathbf{w}(H) &= 2(n-1) \cdot x\epsilon + (1 + x\epsilon)\mathbf{w}(x_n, y_1) + \mathbf{w}(x_n, y_1) \\
&\le 2\left(\epsilon + \frac{1}{2x}\right) + (2 + x\epsilon) \cdot \left(1 - \frac{1}{2x}\right) \\
&< 2\epsilon + 1/x + 2 + x\epsilon < 4.
\end{aligned}$$

Next, we study the total weight of $G_{\mathrm{gr}(x)}$. The greedy algorithm would first go over all edges on path π and add them to $G_{\mathrm{gr}(x)}$. After that, when it comes to the edge (x_n, y_1), since $\mathbf{w}(x_n, w) + \mathbf{w}(w, y_1) = (1 + x\epsilon)\mathbf{w}(x_n, y_1)$, it would not include edge (x_n, y_1).

Finally, it makes a pass over all edges (x_i, y_i) for all $1 \le i \le n$. We argue that the greedy algorithm must add edge (x_i, y_i) in $G_{\mathrm{gr}(x)}$, namely $\mathsf{dist}_{G_{\mathrm{gr}(x)}}(x_i, y_i) > 1 + x\epsilon$ for the current $G_{\mathrm{gr}(x)}$.

Consider any path ρ between x_i and y_i in the current version of $G_{\mathrm{gr}(x)}$. If $\rho = \pi[x_i, y_i]$, then we have

$$\begin{aligned}\mathbf{w}(\pi[x_i, y_i]) &= (n-1)x\epsilon + (1+x\epsilon)\mathbf{w}(x_n, y_1)\\ &= 1 + \epsilon + x\epsilon \cdot \mathbf{w}(x_n, y_1)\\ &> 1 + \epsilon + x\epsilon - (x^2\epsilon^2 + \epsilon/2)\\ &> 1 + x\epsilon + (\epsilon/2 - x^2\epsilon^2) \geq 1 + x\epsilon.\end{aligned}$$

Otherwise if $\rho \neq \pi[x_i, y_i]$, ρ would contain the edge (x_k, y_k) for some $1 \leq k < i$ along with the edges (x_k, x_{k+1}) and (y_k, y_{k+1}). Since all edges $(x_j, x_{j+1}), (y_j, y_{j+1})$ have weight $x\epsilon$, the total weight of ρ would be at least $1 + 2x\epsilon > 1 + x\epsilon$.

In the end, $G_{\mathrm{gr}(x)}$ would include the edges (x_i, y_i) for all $1 \leq i \leq n$, so $\mathbf{w}(G_{\mathrm{gr}(x)}) > n$. Hence, the approximation ratio is at least

$$\frac{\mathbf{w}\left(G_{\mathrm{gr}(x)}\right)}{\mathbf{w}(G_{\mathsf{opt},\epsilon})} > \frac{n}{4} > \frac{\epsilon^{-1}}{8x^2}. \quad \square$$

Acknowledgements. Hung Le and Cuong Than are supported by the NSF CAREER award CCF-2237288, the NSF grants CCF-2517033 and CCF-2121952, and a Google Research Scholar Award. Research by Csaba D. Tóth was supported by the NSF award DMS-2154347. Shay Solomon is funded by the European Union (ERC, DynOpt, 101043159). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Shay Solomon is also funded by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF). Work of Tianyi Zhang was done while at ETH Zürich when supported by funding from the starting grant "A New Paradigm for Flow and Cut Algorithms" (no. TMSGI2_218022) of the Swiss National Science Foundation.

References

[ABS+20] Abu Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Comput. Sci. Rev.*, 37:100253, 2020. doi:10.1016/J.COSREV.2020.100253.

[ADD+93] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993. doi:10.1007/BF02189308.

[AHJ+19] Abu Reyan Ahmed, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen G. Kobourov, Faryad Darabi Sahneh, and Richard Spence. Approximation algorithms and an integer program for multi-level graph spanners. In *Proc. Analysis of Experimental Algorithms (SEA²)*, volume 11544 of *LNCS*, pages 541–562. Springer, 2019. doi:10.1007/978-3-030-34029-2_35.

[Bak94] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

[BBM+11] Piotr Berman, Arnab Bhattacharyya, Konstantin Makarychev, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Improved approximation for the directed spanner problem. In *Proc. 38th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 6755 of *LNCS*, pages 1–12. Springer, 2011. doi:10.1007/978-3-642-22006-7_1.

[BCE+11] MohammadHossein Bateni, Chandra Chekuri, Alina Ene, Mohammad Taghi Hajiaghayi, Nitish Korula, and Dániel Marx. Prize-collecting Steiner problems on planar graphs. In *Proc. 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1028–1049, 2011. doi:10.1137/1.9781611973082.79.

[BCJW24] Fritz Bökler, Markus Chimani, Henning Jasper, and Mirko H. Wagner. Exact minimum weight spanners via column generation. In *Proc. 32nd European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 30:1–30:17. Schloss Dagstuhl, 2024. doi:10.4230/LIPICS.ESA.2024.30.

[BDHM16] MohammadHossein Bateni, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dániel Marx. A PTAS for planar group Steiner tree via spanner bootstrapping and prize collecting. In *Proc. 48th ACM Symposium on Theory of Computing (STOC)*, pages 570–583, 2016. doi:10.1145/2897518.2897549.

[BGJ+12] Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012. doi:10.1137/110826655.

[BH98] Ulrik Brandes and Dagmar Handke. NP-completeness results for minimum planar spanners. *Discrete Mathematics & Theoretical Computer Science*, 3, 1998. URL: http://eudml.org/doc/120593.

[BHM11] Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5), 2011. doi:10.1145/2027216.2027219.

[BKM09] Glencora Borradaile, Philip Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Trans. Algorithms*, 5(3):31:1–31:31, 2009. doi:10.1145/1541885.1541892.

[BRR10] Piotr Berman, Sofya Raskhodnikova, and Ge Ruan. Finding sparser directed spanners. In *Proc. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8 of *LIPIcs*, pages 424–435. Schloss Dagstuhl, 2010. doi:10.4230/LIPICS.FSTTCS.2010.424.

[CC13] Paz Carmi and Lilach Chaitman-Yerushalmi. Minimum weight Euclidean t-spanner is NP-hard. *J. Discrete Algorithms*, 22:30–42, 2013. doi:10.1016/J.JDA.2013.06.010.

[CCL+23] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): $O(1)$ trees suffice. In *Proc. 64th IEEE Symposium on Foundations of Computer Science (FOCS)*, page 2231–2261, 2023. doi:10.1109/focs57990.2023.00139.

[CDNS95] Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. *Int. J. Comput. Geom. Appl.*, 5:125–144, 1995. doi:10.1142/S0218195995000088.

[Che86] L. Paul Chew. There is a planar graph almost as good as the complete graph. In *Proc. 2nd ACM Symposium on Computational Geometry (SoCG)*, pages 169–177, 1986. doi:10.1145/10515.10534.

[CK94] Leizhen Cai and Mark Keil. Spanners in graphs of bounded degree. *Networks*, 24(4):233–249, 1994.

[Cla87] Kenneth Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pages 56–65, 1987. doi:10.1145/28395.28402.

[CW18] Shiri Chechik and Christian Wulff-Nilsen. Near-optimal light spanners. *ACM Trans. Algorithms*, 14(3):33:1–33:15, 2018. doi:10.1145/3199607.

[dBK12] Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geom. Appl.*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

[DFG11] Feodor F. Dragan, Fedor V. Fomin, and Petr A. Golovach. Approximation of minimum weight spanners for sparse graphs. *Theor. Comput. Sci.*, 412(8-10):846–852, 2011. doi:10.1016/J.TCS.2010.11.034.

[DK11] Michael Dinitz and Robert Krauthgamer. Directed spanners via flow-based linear programs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2011. doi:10.1145/1993636.1993680.

[DKR16] Michael Dinitz, Guy Kortsarz, and Ran Raz. Label cover instances with large girth and the hardness of approximating basic k-spanner. *ACM Trans. Algorithms*, 12(2):25:1–25:16, 2016. doi:10.1145/2818375.

[DNS95] Gautam Das, Giri Narasimhan, and Jeffrey Salowe. A new way to weigh malnourished Euclidean graphs. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 215–222, 1995. URL: http://dl.acm.org/citation.cfm?id=313651.313697.

[DZ16] Michael Dinitz and Zeyu Zhang. Approximating low-stretch spanners. In *Proc. 27th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 821–840, 2016. doi:10.1137/1.9781611974331.CH59.

[FKS19] Eli Fox-Epstein, Philip N. Klein, and Aaron Schild. Embedding planar graphs into low-treewidth graphs with applications to efficient approximation schemes for metric problems. In *Proc. 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1069–1088, 2019. doi:10.1137/1.9781611975482.66.

[FL22] Arnold Filtser and Hung Le. Low treewidth embeddings of planar and minor-free metrics. In *Proc. 63rd IEEE Symposium on Foundations of Computer Science (FOCS)*, page 1081–1092, 2022. doi:10.1109/focs54457.2022.00105.

[FN22] Arnold Filtser and Ofer Neiman. Light spanners for high dimensional norms via stochastic decompositions. *Algorithmica*, 84(10):2987–3007, 2022. doi:10.1007/S00453-022-00994-0.

[GKL23] Elena Grigorescu, Nithish Kumar, and Young-San Lin. Approximation algorithms for directed weighted spanners. *Proc. 26th Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX)*, 275:8:1–8:23, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.8.

[GMW23] Renzo Gómez, Flávio Keidi Miyazawa, and Yoshiko Wakabayashi. Improved NP-hardness results for the minimum t-spanner problem on bounded-degree graphs. *Theor. Comput. Sci.*, 947:113691, 2023. doi:10.1016/J.TCS.2023.113691.

[HHZ21] Bernhard Haeupler, D. Ellis Hershkowitz, and Goran Zuzic. Tree embeddings for hop-constrained network design. In *Proc. 53rd ACM Symposium on Theory of Computing (STOC)*, pages 356–369, 2021. doi:10.1145/3406325.3451053.

[Kei88] J. Mark Keil. Approximating the complete Euclidean graph. In *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 318 of *LNCS*, pages 208–213. Springer, 1988. doi:10.1007/3-540-19487-8_23.

[Kle05] Philip N. Klein. A linear-time approximation scheme for planar weighted TSP. In *Proc. 46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 647–657, 2005. doi:10.1109/SFCS.2005.7.

[Kob18] Yusuke Kobayashi. NP-hardness and fixed-parameter tractability of the minimum spanner problem. *Theor. Comput. Sci.*, 746:88–97, 2018. doi:10.1016/J.TCS.2018.06.031.

[Kor01] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30:432–450, 2001. doi:10.1007/S00453-001-0021-Y.

[KP94] Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *Journal of Algorithms*, 17(2):222–236, 1994. doi:10.1006/JAGM.1994.1032.

[LS22a] Hung Le and Shay Solomon. Near-optimal spanners for general graphs in (nearly) linear time. In *Proc. 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3332–3361, 2022. doi:10.1137/1.9781611977073.132.

[LS22b] Hung Le and Shay Solomon. Truly optimal Euclidean spanners. *SIAM Journal on Computing*, 0(0):FOCS19–135–FOCS19–199, 2022. doi:10.1137/20M1317906.

[LS23] Hung Le and Shay Solomon. A unified framework for light spanners. In *Proc. 55th ACM Symposium on Theory of Computing (STOC)*, pages 295–308, 2023. doi:10.1145/3564246.3585185.

[LST+24] Hung Le, Shay Solomon, Cuong Than, Csaba D Tóth, and Tianyi Zhang. Towards instance-optimal Euclidean spanners. In *Proc. 65th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1579–1609, 2024. doi:10.1109/FOCS61266.2024.00099.

[NS07] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, January 2007. doi:10.1017/cbo9780511546884.

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

[PU89] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989. doi:10.1137/0218050.

[RS91] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proc. 3rd Canadian Conference on Computational Geometry (CCCG)*, pages 207–210, 1991. URL: https://cccg.ca/proceedings/1991/paper50.pdf.

[Smi25] Michiel Smid. New references related to geometric spanner networks, 2025. https://people.scs.carleton.ca/~michiel/SpannerBook/newreferences.html, accessed March 2025.

[SZ04] Mikkel Sigurd and Martin Zachariasen. Construction of minimum-weight spanners. In *Proc. 12th European Symposium on Algorithms (ESA)*, volume 3221 of *LNCS*, pages 797–808. Springer, 2004. doi:10.1007/978-3-540-30140-0_70.