Optimal Fault-Tolerant Spanners in Euclidean and Doubling Metrics: Breaking the $\Omega(\log n)$ Lightness Barrier

Hung Le[*1], Shay Solomon[†2], and Cuong Than[‡3]

[1]University of Massachusetts at Amherst
[2]Tel Aviv University
[3]University of Massachusetts at Amherst

Abstract

An essential requirement of spanners in many applications is to be *fault-tolerant*: a $(1 + \epsilon)$-spanner of a metric space is called *(vertex) f-fault-tolerant (f-FT)* if it remains a $(1+\epsilon)$-spanner (for the non-faulty points) when up to f faulty points are removed from the spanner. Fault-tolerant (FT) spanners for Euclidean and doubling metrics have been extensively studied since the 90s.

For low-dimensional Euclidean metrics, Czumaj and Zhao in SoCG'03 [CZ03] showed that the optimal guarantees $O(fn)$, $O(f)$ and $O(f^2)$ on the size, degree and lightness of f-FT spanners can be achieved via a greedy algorithm, which naïvely runs in $O(n^3) \cdot 2^{O(f)}$ time.[1] An earlier construction, by Levcopoulos et al. [LNS98] from STOC'98, has a faster running time of $O(n \log n) + n2^{O(f)}$, but has a slack of $2^{\Omega(f)}$ in all the three involved parameters. The question of whether the optimal bounds of [CZ03] can be achieved via a fast construction has remained elusive, with the *lightness* parameter being the bottleneck: Any construction (other than [CZ03]) has lightness either $2^{\Omega(f)}$ or $\Omega(\log n)$. Moreover, in the wider family of doubling metrics, it is not even clear whether *there exists* an f-FT spanner with lightness that depends solely on f (even exponentially): all existing constructions have lightness $\Omega(\log n)$ since they are built on the *net-tree spanner*, which is induced by a hierarchical *net-tree* of lightness $\Omega(\log n)$.

In this paper we settle in the affirmative these longstanding open questions. Specifically, we design a construction of f-FT spanners **that is optimal with respect to all the involved parameters** (size, degree, lightness and running time): For any n-point doubling metric, any $\epsilon > 0$, and any integer $1 \le f \le n - 2$, our construction provides, within time $O(n \log n + fn)$, an f-FT $(1 + \epsilon)$-spanner with size $O(fn)$, degree $O(f)$ and lightness $O(f^2)$.

To break the $\Omega(\log n)$ lightness barrier, we introduce a new geometric object — the *light net-forest*. Like the net-tree, the light net-forest is induced by a hierarchy of nets. However,

[1]The *degree* of a spanner is the maximum degree of any vertex, and the spanner *lightness* is the ratio between the spanner *weight* (i.e., the sum of edge weights) and the MST weight.

to ensure small lightness, the light net-forest is inherently less "well-connected" than the net-tree, which, in turn, makes the task of achieving fault-tolerance significantly more challenging. Further, to achieve the optimal degree (and size) *together* with optimal lightness, and to do so within the optimal running time — we overcome several highly nontrivial technical challenges.

Contents

1 Introduction

1.1 Euclidean Spanners

Let P be a set of n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be any parameter. A spanning subgraph $H = (P, E, \|\cdot\|)$ of the complete Euclidean graph induced by P is called a (Euclidean) $(1+\epsilon)$-*spanner* for the point set P if $\forall p, q \in P$, there is a $(1+\epsilon)$-*spanner path* in H between p and q, i.e., a path of *weight* at most $(1+\epsilon) \cdot \|p-q\|$, where the weight of a path is the sum of all edge weights in it and $\|p-q\|$ denotes the Euclidean distance between p and q. Euclidean spanners find applications in various areas, including in geometric approximation algorithms, network topology design and distributed systems, and they have been studied extensively since the 80s [Che86, Cla87, KG92, ADD+93, ADM+95, AWY05]; see also the book by Narasimhan and Smid [NS07] titled "Geometric Spanner Networks", which is devoted to Euclidean spanners and their applications.

A natural requirement from a spanner, which is essential for real-life applications, is to be *robust against failures*, so that even when part of the network fails, we still have a good spanner for the functioning part of the network. Formally, a Euclidean spanner H for point set P is called a *(vertex) f-FT $(1+\epsilon)$-spanner*, for $1 \le f \le n-2$, if for any $F \subseteq P$ with $|F| \le f$, the graph $H \smallsetminus F$ (obtained by removing from H the vertices of F and their incident edges) is a $(1+\epsilon)$-spanner for $P \smallsetminus F$.[2] (The basic (non-FT) setting corresponds to the case $f = 0$.) To perform efficiently in these applications, we would like the underlying spanner to be "sparse". The *size* (number of edges) of the spanner is perhaps the most basic sparsity measure; the spanner *sparsity* is defined as the ratio of the spanner size to the minimum size $n-1$ of a connected spanning subgraph. The *weight* (sum of edge weights) of the spanner is a natural generalization of the size, and in many applications (such as for the metric TSP) we need to have small weight rather than small size; the spanner *lightness* is defined as the ratio of the spanner weight to the minimum spanning tree (MST) weight. The spanner size corresponds to the *average* degree of a vertex, yet the stronger property of a small *(maximum) degree* (over all vertices) is important in various applications in Computational Geometry, as well as for reducing the space usage in compact routing schemes and in distributed systems.

A construction of Euclidean $(1+\epsilon)$-spanners with constant degree (and sparsity) and lightness can be built in time $O(n \log n)$ [AS94, DN94, GLN02]. In their pioneering work, Levcopoulos, Narasimhan and Smid [LNS98] introduced the notion of FT spanners and generalized the basic construction of [AS94, DN94, GLN02] to obtain an f-FT $(1+\epsilon)$-spanner with degree (and sparsity) and lightness bounded by $2^{O(f)}$, within a running time of $O(n \log n) + n 2^{O(f)}$. Clearly, the degree of any vertex in any f-FT spanner (for any stretch) must be at least $f + 1$, and thus any f-FT spanner must have $\Omega(fn)$ edges. There are also simple point sets (even in 1 dimension), for which any f-FT spanner must have lightness $\Omega(f^2)$ [CZ03]. Finally, the time needed to compute a k-FT spanner is $\Omega(n \log n + fn)$: The term $\Omega(n \log n)$ is the time lower bound for computing a basic (non-FT) spanner in the *algebraic computation tree model* [CDS01], and the term $\Omega(fn)$ is the aforementioned lower bound on the size of any f-FT spanner.

There are also bunch of other constructions of Euclidean FT spanners (see Table 1 for a summary of FT constructions), and they can be grouped into two categories.

In the first category, which contains almost all known constructions, the lightness parameter is either ignored or bounded from below by $\Omega(\log n)$; this line of work was culminated with the con-

---

[2]We shall restrict the attention to vertex faults, but that does not lose generality: Any FT $(1+\epsilon)$-spanner that is resilient to f vertex faults is also resilient to f edge faults (see [LNS98, NS07]), while the lower bounds discussed below — of $\Omega(f)$ on sparsity and degree and $\Omega(f^2)$ on lightness of f-FT spanners — apply also to edge faults.

Reference	Sparsity	Degree	Lightness	Runtime	Metric
[LNS98]	$2^{O(f)}$	$2^{O(f)}$	$2^{O(f)}$	$O(n \log n) + n2^{O(f)}$	Euclidean
[LNS98]	$O(f^2)$	unspecified	unspecified	$O(n \log n + f^2 n)$	Euclidean
[LNS98]	$O(f \log n)$	unspecified	unspecified	$O(fn \log n)$	Euclidean
[Luk99]	$O(f)$	unspecified	unspecified	$O(n \log^{d-1} n + fn \log \log n)$	Euclidean
[CZ03]	$O(f)$	$O(f)$	$O(f^2)$	$\Omega(n^3) \cdot 2^{O(f)}$	Euclidean
[CZ03]	$O(f)$	$O(f)$	$O(f^2 \log n)$	$O(fn \log^d n + nf^2 \log f)$	Euclidean
[CLN12]	$O(f)$	unspecified	unspecified	unspecified	doubling
[CLN12]	$O(f^2)$	$O(f^2)$	unspecified	unspecified	doubling
[CLNS13]	$O(f^2)$	$O(f^2)$	$O(f^2 \log n)$	$O(n \log n + f^2 n)$	doubling
[Sol14]	$O(f)$	$O(f)$	$O(f^2 + f \log n)$	$O(n \log n + fn)$	**doubling**
New	$\boldsymbol{O(f)}$	$\boldsymbol{O(f)}$	$\boldsymbol{O(f^2)}$	$\boldsymbol{O(n \log n + fn)}$	**doubling**

Table 1: A comparison between previous and our constructions of FT spanners with small size, degree, lightness and runtime, for low-dimensional Euclidean and doubling metrics. The O-notation ignores dependencies on ϵ and d.

struction of Solomon [Sol14] from STOC'14, which achieves the optimal degree $O(f)$ and running time $O(n \log n + nf)$, and it applies to the wider family of *doubling metrics* (see Section 1.2). In addition to optimal degree and running time, the construction of [Sol14] also achieves a near-optimal tradeoff of $O(f^2 + f \log n)$ versus $O(\log n)$ between the lightness and another property called the *(hop-)diameter*;[3] importantly, any construction of diameter $O(\log n)$ must have lightness $\Omega(\log n)$, and more precisely this tradeoff between lightness and diameter is optimal up to a factor $\log f$ slack on the diameter and a factor $\min\{\log f, \frac{\log n}{f}\} = O(\log \log n)$ slack on the lightness. (For a detailed discussion on the lower bound tradeoff between the lightness and diameter, we refer to [Sol14].)

The second category concerns "light" spanners, and there are only two such constructions to date. The first is the one by [LNS98] mentioned above, and the second is a *greedy* construction due to Czumaj and Zhao from SoCG'03 [CZ03], and it achieves the optimal lightness guarantee of $O(f^2)$ together with the optimal degree (and sparsity) of $O(f)$. However, a naïve implementation of the greedy FT spanner construction requires time $\Omega(n^3) \cdot 2^{\Omega(f)}$ or more precisely $\Omega(n^2 \cdot \texttt{Paths}(n, f+1, 1+\epsilon))$, where $\texttt{Paths}(n, f+1, 1+\epsilon)$ is the time needed to check whether an n-vertex Euclidean graph with $O(fn)$ edges contains $f+1$ vertex-disjoint $(1+\epsilon)$-spanner paths between an arbitrary pair of vertices. Indeed, in the greedy FT construction, the $\binom{n}{2}$ edges of the underlying Euclidean metric are traversed by nondecreasing weights, and each edge (x, y) is added to the current spanner iff it does not contain $f+1$ vertex-disjoint $(1+\epsilon)$-spanner paths between x and y. We note that a more sophisticated implementation of the basic (non-FT) greedy algorithm takes time $O(n^2 \log n)$ in Euclidean and doubling metrics [BCF+10], but it is unclear if this implementation can be extended to the FT greedy algorithm; moreover, even if such an extension is possible and even if we completely ignore the dependence on f, it would still lead to a super-quadratic in n runtime.

Up to this date no construction of Euclidean f-FT spanners with runtime better than the $\Omega(n^3) \cdot 2^{\Omega(f)}$ bound of [CZ03], let alone the optimal $O(n \log n + fn)$ runtime bound, could achieve lightness $\min\{o(\log n), 2^{o(f)}\}$, let alone the optimal lightness of $O(f^2)$. In particular, the following

---
[3] A spanner for point set P is said to have a *(hop-) diameter* of k if it provides a $(1+\epsilon)$-spanner path with at most k edges, for every $p, q \in P$.

question, stated in the book of Narasimhan and Smid [NS07], has remained open even for 2-dimensional point sets since the STOC'98 work of [LNS98].

Question 1 (Open Problem 28 in [NS07])**.** *Is there an algorithm that constructs, within $O(n \log n + fn)$ time, an f-FT $(1+\epsilon)$-spanner with lightness $O(f^2)$?*

Further, is it possible to construct, ideally still in time $O(n \log n + fn)$, a single construction that combines the optimal lightness $O(f^2)$ with the optimal degree (and sparsity) $O(f)$?

Question 2. *Is there an algorithm that constructs, within $O(n \log n + fn)$ time, an f-FT $(1+\epsilon)$-spanner with lightness $O(f^2)$ and degree $O(f)$?*

1.2 Doubling Metrics

A metric is called *doubling* if its *doubling dimension* is constant, where the latter is the smallest value d such that every ball B in the metric can be covered by at most 2^d balls of half the radius of B. We note that the doubling dimension generalizes the standard Euclidean dimension, since the doubling dimension of the Euclidean space $\mathbb{R}^d$ is $\Theta(d)$. Spanners for doubling metrics have been intensively studied; see [GGN04, CGMZ05, CG06, HPM06, GR08a, GR08b, Smi09, CLN12, ES15, CLNS13, Sol14, BLW19, LT22, KLMS22], and the references therein. Many of these works share a common theme, namely, to devise spanners for doubling metrics that are just as good as the analog Euclidean spanner constructions.

Some of the constructions mentioned in Section 1.1 apply to doubling metrics; see Table 1. Much weaker variants of Questions 1 and 2 from Section 1.1 can be asked for the wider family of doubling metrics. Indeed, in such metrics, it is not even clear whether *there exists* an f-FT spanner with lightness that depends solely on f (even exponentially): All existing constructions have lightness $\Omega(\log n)$ since they are built on the *net-tree spanner*, which is induced by a hierarchical *net-tree* of lightness $\Omega(\log n)$. The net-tree incurs a lightness of $\Omega(\log n)$ even for line metrics!

Question 3. • *Does there exist, for any doubling metric, an f-FT $(1+\epsilon)$-spanner with lightness $O(f^2)$? Does there exist such a spanner that also achieves degree $O(f)$?*

• *Further, is there an algorithm that constructs, for any doubling metric, within $O(n \log n + fn)$ time, an f-FT $(1+\epsilon)$-spanner with lightness $O(f^2)$ and degree $O(f)$?*

1.3 Our Contribution

The main result of this work is the following theorem.

Theorem 1. *Let (X, δ) be an n-point doubling metric, with an arbitrary doubling dimension d. For any $0 < \epsilon < \frac{1}{2}$ and any integer $1 \le f \le n - 2$, an f-FT $(1+\epsilon)$-spanner with lightness $\epsilon^{-O(d)} \cdot f^2$ and degree $\epsilon^{-O(d)} \cdot f$ can be built within $\epsilon^{-O(d)}(n \log n + fn)$ time.*

The construction provided by Theorem 1 is **optimal with respect to all the involved parameters** and it settles all the aforementioned questions, Questions 1-3, in the affirmative. We note that our construction improves the previous state-of-the-art constructions of FT spanners (with sub-cubic runtime) not only for doubling metrics, but also for Euclidean ones.

A central challenge that we faced on the way to proving Theorem 1 is breaking the $\Omega(\log n)$ lightness barrier. To this end we introduce a new geometric object — the *light net-forest*. Like the

net-tree, the light net-forest is induced by a hierarchy of nets. However, to ensure small lightness, the light net-forest is inherently less "well-connected" than the net-tree, which, in turn, makes the task of achieving fault-tolerance significantly more challenging. We demonstrate the power of the light net-forest in achieving the optimal lightness, but our construction does not stop there. Further, to achieve the optimal degree (and size) *together* with optimal lightness, and to do so within the optimal running time — we overcome several highly nontrivial technical challenges. In the following section we describe the technical and conceptual contributions of this work.

1.4   Technical Overview and Conceptual Highlights

The previous constructions of Euclidean FT spanners [LNS98, Luk99, CZ03, NS07] rely on geometric properties of low-dimensional Euclidean metrics, such as the *gap property* [AS94] and the *leapfrog property* [DHN93]. In particular, achieving small lightness crucially relies on the leapfrog property, which is not known to extend to arbitrary doubling metrics. On the other hand, the previous constructions of doubling FT spanners [CLN12, CLNS13, Sol14] use standard packing arguments of doubling metrics, but they all rely on the standard *net-tree spanner* of [GGN04, CGMZ05], which is induced by a hierarchical *net-tree $T = T(X)$* that corresponds to a *hierarchical partition* of the metric (X, δ). The lightness of the net-tree alone is $\Omega(\log n)$, even in 1-dimensional Euclidean spaces; as such, all the constructions of [CLN12, CLNS13, Sol14] incur a lightness of $\Omega(\log n)$ even when ignoring the dependencies on f.

   As in all previous constructions that apply to arbitrary doubling metrics, the starting point of our construction is the net-tree spanner $T = T(X)$. To break the lightness barrier of $\Omega(\log n)$, we will not be able to use the entire net-tree, and consequently we will not be able to use the entire net-tree spanner that is derived from it. We start with a brief overview of the net-tree spanner and the previous constructions of [CLN12, CLNS13, Sol14].

   Any tree node x is associated with a single point $\text{rep}(x)$ that belongs to the point set $L(x)$ of its descendant leaves. For any pair x, y of level-i tree nodes that are close together with respect to the *distance scale* (or *radius*) 2^i at that level, a *cross edge* (x, y) is added (edge (x, y) translates to edge $(\text{rep}(x), \text{rep}(y))$; for brevity we sometimes write (x, y) as a shortcut for $(\text{rep}(x), \text{rep}(y))$); specifically, the weight of any level-i cross edge (x, y) is at most $\lambda 2^i$, where $\lambda = \Theta(\frac{1}{\epsilon})$ and $\text{rad}(x) = \text{rad}(y) = 2^i$. The net-tree spanner is the union of the tree edges (i.e., the edges of T) and the cross edges. For every pair u, v of points, a $(1+\epsilon)$-spanner path, denoted by $\Pi_T(u, v)$, goes up in the tree T from a leaf x corresponding to u (i.e., $\text{rep}(x) = u$) to some ancestor x' of u, then takes a cross edge $(\text{rep}(x'), \text{rep}(y'))$ from x' to an ancestor y' of y in the net-tree, and finally goes down in T from y' to y, where $\text{rep}(y) = v$. The reason $\Pi_T(u, v)$ is a $(1+\epsilon)$-spanner path is due to the following key observation of the net-tree, which implies that the weight of the cross edge constitutes almost the entire weight of $\Pi_T(u, v)$:

Observation 1. *Both $\delta(\text{rep}(x), \text{rep}(x'))$ and $\delta(\text{rep}(y), \text{rep}(y'))$ are at most $\epsilon\delta(u, v)$. Thus $(1 - \epsilon)\delta(u, v) \leq \delta(\text{rep}(x'), \text{rep}(y')) \leq (1 + \epsilon)\delta(u, v)$.*

   To achieve fault-tolerance, the general idea in [CLNS13] was to associate each tree node x with a *surrogate set $S(x)$* of (up to) $f + 1$ points from $L(x)$ rather than a single point; the FT spanner is obtained by replacing each edge (x, y) of the basic net-tree spanner by a bipartite clique between the corresponding sets $S(x)$ and $S(y)$. To achieve a degree of $O(f^2)$, [CLNS13] used a "rerouting" technique from [GR08b] that assigns representative points for the tree nodes so as to minimize the maximum degree, but achieving degree $o(f^2)$ using this approaches is doomed for two reasons.

4

"Global" reason: If each edge (x, y) of the basic net-tree spanner is replaced by a bipartite clique between $S(x)$ and $S(y)$, then since $S(x)$ and $S(y)$ may contain $\Omega(f)$ points each, the size of this bipartite clique may be $\Omega(f^2)$. As the basic net-tree spanner has $\Omega(n)$ edges, the FT spanner obtained in this way will contain $\Omega(f^2 n)$ edges (and will have degree $\Omega(f^2)$).

"Local" reason: As a node x is associated with a set $S(x)$ of (up to) $f + 1$ points from $L(x)$, the same leaf point p may belong to $\Omega(f)$ different sets $S(x)$ of internal nodes x. For each edge (x, y) of the basic net-tree spanner that is incident on any of these $\Omega(f)$ nodes, $p \in S(x)$ is connected via edges to all $\Omega(f)$ points of $S(y)$, and so the degree of p will be $\Omega(f^2)$.

The key idea of [Sol14]. The idea of associating nodes of net-trees and other hierarchical tree structures, such as split trees and dumbbell trees, with points from their descendant leaves has been widely used in the geometric spanner literature; see [ADM$^+$95, GGN04, CGMZ05, CG06, NS07, GR08b, CLN12], and the references therein. Indeed, by Observation 1, any point in $L(x)$ is close to the original net-point $\text{rep}(x)$ of x, with respect to the distance scale $\text{rad}(x)$ of x. Instead of associating nodes x with points chosen exclusively from $L(x)$, the key idea of [Sol14] is to consider a wider set $B(x)$ of all points in the ball of radius $O(\text{rad}(x))$ centered at $\text{rep}(x)$. By associating nodes x with points from $B(x)$, one obtains a *hierarchical cover* of the metric (rather than a hierarchical partition). As the doubling dimension is constant, this cover has a constant *degree*, i.e., every point belongs to $O_{\epsilon,d}(1)$ sets $B(\cdot)$ at each level of the tree T.

Similarly to [CLNS13], [Sol14] associates each tree node x with a set $S(x)$ of (up to) $f + 1$ points called *surrogates*, but as mentioned the surrogates in [Sol14] are chosen from the superset $B(x)$ of $L(x)$. Moreover, [Sol14] doesn't naively replace each edge (x, y) of the basic net-tree spanner by a bipartite clique between $S(x)$ and $S(y)$ as in [CLNS13], since (due to the "Global" reason above) that would lead to $\Omega(f^2 n)$ edges. Instead, whenever the number of surrogates in $S(x)$ and $S(y)$ is $f + 1$, a bipartite matching suffices for achieving fault-tolerance. [Sol14] assigns the surrogates bottom-up (first for level-0 nodes (leaves) in the net-tree T, then for level-1 nodes, etc.) via a complex procedure that guarantees that, for any level i and any level-i node x in T, there are enough points of small degree in $B(x)$ to choose surrogates from, which ultimately leads to the desired degree bound of $O(f)$. However, as mentioned, the lightness of the construction of [Sol14], as well as any other construction that applies to doubling metrics, is lower bounded by the lightness $\Omega(\log n)$ of the underlying net-tree; more precisely, the state-of-the-art lightness of any construction of FT spanners in doubling metrics prior to this work, due to [Sol14], is $\Theta(f^2 + f \log n)$.

1.4.1 Our approach

To breach the $\Omega(\log n)$ lightness barrier incurred by the net-tree, our first insight is that a "light" $(1 + \epsilon)$-spanner G of X, which is given as input, can be used for computing a light subtree of the net-tree — which we name the *light net-forest* and abbreviate as LNF. Equipped with the LNF, a natural approach would be to (i) apply the standard net-tree spanner construction on top of the LNF (instead of the net-tree) in order to get a light net-tree spanner, and (ii) transform the light net-tree spanner into a light FT spanner by replacing each edge (x, y) of the spanner with a bipartite clique / matching between $S(x)$ and $S(y)$ similarly to [Sol14]. However, since the LNF is not a tree but rather a forest, it is inherently less "well-connected" than the net-tree, which renders step (ii) of achieving fault-tolerance highly challenging, as we next describe.

In the standard net-tree spanner, as mentioned, for every pair u, v of points, there is a path $\Pi_T(u, v)$ that uses a *single cross edge*, which we denote here by (u', v'), where u' and v' are ancestors

of u and v in the net-tree T, respectively. A non-faulty $(1 + \epsilon)$-spanner path in the FT spanner constructions of [CLNS13, Sol14], for a **non-faulty pair** u, v of points, consists of going up in T from u and from v to some non-faulty surrogates of $S(u')$ and $S(v')$, respectively, and it includes a single cross edge between those surrogates. There are non-faulty surrogates in $S(u')$ and $S(v')$ due to two reasons: (1) u and v are non-faulty descendant leaves of u' and v', respectively, and (2) there is only one cross edge in the path. Indeed, if $|S(u')| < f + 1$ then, since u is a descendant leaf of u', one can guarantee that $u \in S(u')$; otherwise u may not belong to $S(u')$, but in that case too $S(u')$ contains a non-faulty surrogate; the same goes for $S(v')$. However, when using the LNS, we no longer have such a path $\Pi_T(u, v)$ for any pair u, v of points. Instead, we can afford to use such a path $\Pi_T(u, v)$ only for edges (u, v) that belong to the light $(1 + \epsilon)$-spanner G (with constant lightness) that we receive as input; it can be shown that the union of all those paths over the edges of G has constant lightness. Consider now a pair u, v of points that are not incident in G, and let $\Pi_G(u, v) = (u_1 = u, u_2, \ldots, u_k = v)$ be a shortest path (which is a $(1 + \epsilon)$-spanner path) between u and v in G. Naturally, we would like to translate this path $\Pi_G = \Pi_G(u, v)$ into a union of non-faulty subpaths of the net-tree plus cross edges, as that union should provide a non-faulty $(1 + \epsilon)$-spanner path between u and v. However, we cannot argue that ancestors u_i' of intermediate nodes $u_i, u_i \neq u, v$ on the path contain non-faulty surrogates, since u and v are not necessarily descendant leaves of u_i'! **This is the crux in achieving a light FT-spanner from the LNS,** and it entails multiple challenges.

When translating Π_G into a non-faulty path, the basic idea is to replace every cross edge by a bipartite matching. First note that the cross edges corresponding to the edges of Π_G may be located at different levels. Indeed, for an edge $e_i = (u_i, u_{i+1})$ in G, the respective cross edge, denoted by $\texttt{cross}(e_i) = (\hat{u}_i^i, \hat{u}_{i+1}^i)$, lies at level roughly $\log \delta(u_i, u_{i+1})$, and edge weights in the path Π_G may be very different one from another; thus $\hat{u}_{i+1}^i$ and $\hat{u}_{i+1}^{i+1}$, which are the "second" endpoint of $\texttt{cross}(e_i)$ and the "first" endpoint of $\texttt{cross}(e_{i+1})$, respectively, may lie at very different levels of the net-tree. Next, we stress that replacing a cross edge by a bipartite matching is possible only in the case that, for each cross edge $\texttt{cross}(e_i) = (\hat{u}_i^i, \hat{u}_{i+1}^i)$ along the path, there are $f + 1$ "nearby" points around the two endpoints $\hat{u}_i^i$ and $\hat{u}_{i+1}^i$ of the edge. By nearby we mean within distance that is an $O(\epsilon)$-fraction of the distance between $\hat{u}_i^i$ and $\hat{u}_{i+1}^i$, and thus these nearby points can serve as part of the surrogate sets of $S(\hat{u}_i^i)$ and $S(\hat{u}_{i+1}^i)$ of $\hat{u}_i^i$ and $\hat{u}_{i+1}^i$, respectively. Let us consider this simpler case first. Then the distance between a pair of surrogates of $\hat{u}_i^i$ and $\hat{u}_{i+1}^i$ is the same, up to a factor of $1 + O(\epsilon)$, as the distance between $\hat{u}_i^i$ and $\hat{u}_{i+1}^i$. It thus suffices to take a perfect matching between $S(\hat{u}_i^i)$ and $S(\hat{u}_{i+1}^i)$ for every i, and then at least one of the matching edges must function for every i (following at most f vertex faults). One technicality is that when going from $\hat{u}_{i-1}^{i-1}$ to $\hat{u}_i^{i-1}$ through a matching edge we end up at a surrogate of $\hat{u}_i^{i-1}$, say s_i^{i-1}, and then when continuing from $\hat{u}_i^i$ to $\hat{u}_{i+1}^i$ through a matching edge we start at a surrogate of $\hat{u}_i^i$ (rather than $\hat{u}_i^{i-1}$), say s_i^i, and these two surrogates s_i^{i-1} and s_i^i might be different, so in general the union of the matched edges along the path does not form a valid path; however, as the distance between s_i^{i-1} and s_i^i is negligible with respect to the weight of e_i, our construction will provide a non-faulty path between s_i^{i-1} and s_i^i by induction. The challenging case is when for some cross edges along the path, there are less than $f + 1$ "nearby" points around the two endpoints; we refer to such edges as *irreplaceable*. Dealing with irreplaceable edges requires special care; we will get back to this issue towards the end of this section, when we discuss how to construct a non-faulty spanner path. For now we focus on the following fundamental issue.

A major issue is that there could be nodes with less than $f + 1$ nearby points. For an edge

e_i, there might be no functioning vertex near the endpoints of $\texttt{cross}(e_i)$, which means that there might be no functioning edge which is a good approximation of e_i in $H - F$ (in terms of its weight). We resolve this issue by adding more cross edges to guarantee fault tolerance. Specifically, for every node x for which we cannot find $f + 1$ surrogates, we add the bipartite clique between $S(x)$ and $S(y)$ for all y within distance $O(\text{rad}(x)/\epsilon)$ from x. There are two possible cases when we cannot find $f + 1$ surrogates for a node x. The first (and obvious) case is when x is at a low level and there are not enough (less than $f + 1$) points near $\text{rep}(x)$. The second (and more challenging) case is that the choices of surrogates must be subjected to other constraints such as bounded degree; if we choose the surrogates carelessly, there might be fewer usable vertices in the vicinity of $\text{rep}(x)$ since all vertices close to $\text{rep}(x)$ were overused as surrogates by nodes at levels lower than x. In both cases, we add the bipartite clique between $S(x)$ to the surrogate set of every nearby node. However, each case contributes differently to the lightness. While adding many edges at low levels does not affect the lightness significantly, adding those in higher levels could blow up the lightness by a factor of $\log n$; recall that we want to avoid using all the cross edges since they add a $\log n$ factor to the lightness. Thus, we must choose the surrogates carefully to ensure that the second case essentially does not happen. This is where our aforementioned LNF (light net forest) comes to the rescue. Our key insight is that all nodes in the net-tree with less than $f + 1$ nearby surrogates form an LNF, which consists of node-disjoint subtrees of T that span all those nodes, such that the total radii associated with those nodes exceeds the weight of $\text{MST}(X)$ by at most a constant factor, implying that the LNF is light. The structure of the LNF depends heavily on the input spanner G and on our strategy to select surrogates.

The strategy of selecting surrogates directly affects all important parameters of our spanner: stretch, lightness and degree. First, the surrogates of a node x must be close to $\text{rep}(x)$ to guarantee the stretch. Second, to guarantee the bounded degree property, we must avoid overusing any point as a surrogate (i.e., using it as a surrogate for too many nodes in the tree). One idea is to use the leaves of a node as surrogates since the distance between (the representative of) a node x to its leaves is small compared to any cross edge incident to x. However, using only leaves for surrogates might increase the degree of a single vertex to $\Omega(n)$; for example, there might be a long branch of T with only one leaf, say z, where we have to add edges to all the nodes in the branch to guarantee a good stretch, leaving z with many incident edges. To overcome this issue, we choose the surrogates of x in a ball centered at $\text{rep}(x)$ with radius $c \cdot \text{rad}(x)$, for an appropriate constant c. Once the degree (in the spanner) of a vertex reaches a certain threshold, it is forbidden for that vertex to serve as a surrogate ever again.

A new problem arises: choosing surrogates from a ball as suggested above does not guarantee the bounded lightness property. Balls centered at net points at different levels might interact in a complex manner, and as a result, we might not be able to find enough (at least $f + 1$) surrogates for some nodes at high levels, since overused vertices are forbidden to be used again. If we follow the suggestion outlined in the previous paragraph, we will have to add bipartite cliques from the surrogates of these nodes to other nodes's surrogates. Unfortunately, this will break the property of the LNF and, in particular, it will be harder to control the lightness. To resolve this problem, we will guarantee the following property: if a node has at least $f + 1$ surrogates, any of its ancestors will also have $f + 1$ surrogates. To this end, our key idea is to *prioritize* the choice of high-degree vertices in a large ball $\mathbf{B}(x, 16\text{rad}(x))$ over vertices of lower degree in $\mathbf{B}(x, 4\text{rad}(x))$. This is somewhat counter-intuitive as one might expect to prioritize low degree vertices (to have a better chance of bounding the degree of the spanner) over high degree vertices. However, the intuition

is that by prioritizing high degree vertices, we actually "save" the low degree ones to be used as "fresh" surrogates of other nodes at higher levels. Furthermore, when a high degree vertex becomes overused, many of its neighbors of low degree are closer to its ancestors, relative to the radii of the ancestors, leaving the ancestors more room to choose surrogates.

We stress that even if we only try to control the lightness bound (regardless of the degree and even the size of the spanner), and even if we are aiming for a suboptimal dependence on f in the lightness bound, it is still highly challenging to get an FT-spanner with lightness $o(\log n)$. As discussed above, using cross edges restricted to the input light spanner is not enough to guarantee fault-tolerance. The challenge is to identify (or even just prove the existence of) a set of cross edges that has small lightness on the one hand and that can guarantee fault-tolerance on the other; achieving these two contradictory requirements simultaneously is highly non-trivial.
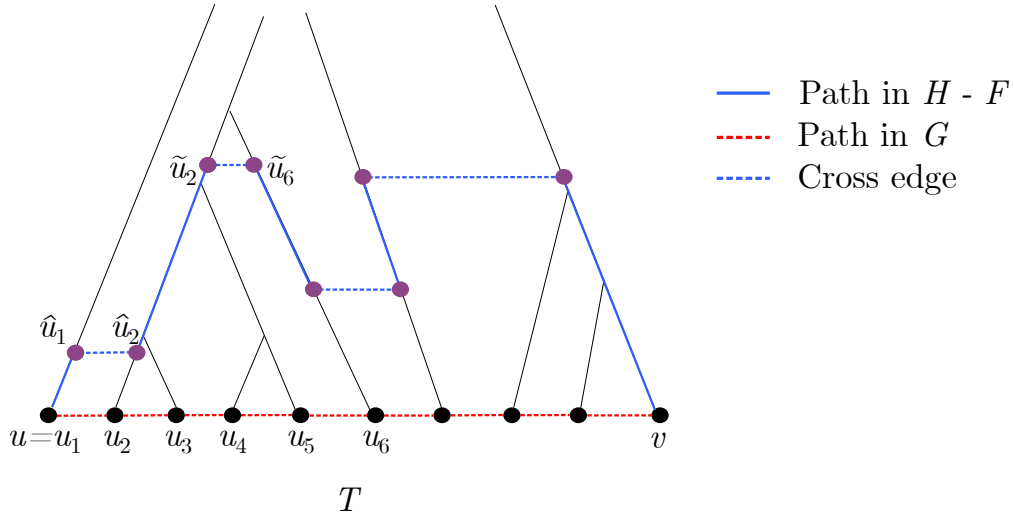

Figure 1: A path from u to v in $H \smallsetminus F$. Every purple node has $f + 1$ surrogates. We replace some edges and some subpaths of the shortest path from u to v in G by the cross edges between nodes having $f+1$ surrogates. It might be misleading that $\delta(\tilde{u}_2, \tilde{u}_6)$ is much smaller than $\delta(u_2, u_6)$. Indeed, $\delta(\tilde{u}_2, \tilde{u}_6)$ is a good approximation of the length of $(u_2, u_3, u_4, u_5, u_6)$.

Finally, we return to the issue of finding a non-faulty spanner path between any two points. Recall that our idea is to carefully choose some (but not all) cross edges of T and add the edges between the respective surrogates of each chosen cross edge to the VFT spanner. For any two points u and v in X, if the lowest good approximation (in terms of its weight) cross edge $(\ddot{u}, \ddot{v})$ of (u, v) is at a "low" level, meaning that there are not enough (less than $f + 1$) surrogates for either $\ddot{u}$ or $\ddot{v}$, our construction guarantees that $(\ddot{u}, \ddot{v})$ is chosen, and there is an non-faulty edge between $S(\ddot{u})$ and $S(\ddot{v})$. If $\ddot{u}$ and $\ddot{v}$ are at a high enough level, then $(\ddot{u}, \ddot{v})$ might not be chosen. In that case, we follow the shortest path $P = \{u = u_1, u_2, u_3, \ldots u_l = v\}$ between u and v in the light spanner G (if there are multiple shortest paths, we fix one arbitrarily). If (u_1, u_2) is *replaceable*, meaning that there is at least one non-faulty edge in the bipartite matching between $S(\hat{u}_1)$ and $S(\hat{u}_2)$, where $\hat{u}_1$ and $\hat{u}_2$ are ancestors of u_1 and u_2, respectively, at an appropriate level, we replace (u_1, u_2) by a non-faulty edge in the matching. Otherwise, the edge (u_1, u_2) is *irreplaceable*, and we find a good approximation cross edge of (u_1, u_i) for some $i > 2$. Our construction guarantees that we choose a good approximation cross edge $(\tilde{u}_1, \tilde{u}_i)$ of $\mathbf{dist}_P(u_1, u_i)$ for some $i > 2$; **proving this is highly**

challenging and is the key in our argument for finding a non-faulty spanner path. Then, we replace the prefix $(u_1, u_2, \ldots u_i)$ of P by a non-faulty edge in the bipartite matching between $S(\tilde{u}_1)$ and $S(\tilde{u}_i)$ and continue the process recursively with the remaining suffix of P. Finally, the replaced edges can be "glued together" by induction via subpaths of total length negligible with respect to the weight of P; in this way, we have provided a spanner path from $u = u_1$ to $u_l = v$. See Figure 1 for an illustration.

2   Preliminaries

Given a graph G, we denote by $V(G)$ and $E(G)$ the vertex and edge set of G, respectively. Given a metric (X, δ), a graph $G = (V, E, \mathbf{w})$ is a *geometric graph* in (X, δ) if $V \subseteq X$ and $\mathbf{w}(u, v) = \delta(u, v)$ for every edge $(u, v) \in E$. In this paper, we only consider geometric graphs, so we will refer to points and vertices interchangeably. The weight of a graph G, denoted by $\mathbf{w}(G)$, is the total weight of all edges in G. For any set of edges E', $\mathbf{w}(E') = \sum_{(u,v) \in E'} \delta(u, v)$. For any two vertices u and v in G, the distance between u and v in G is denoted by $\mathbf{dist}_G(u, v)$. The minimum spanning tree of X, denoted by $\mathrm{MST}(X)$, is the minimum spanning tree of $(X, \binom{X}{2}, \mathbf{w})$ with $\binom{X}{2}$ is the set containing all pairs of points in X.

We say that G is a *t-spanner* of (X, δ) if G is a geometric graph in (X, δ) with X being its vertex set and for any two points $u, v \in X$, $\mathbf{dist}_G(u, v) \leq t\delta(u, v)$. A path between u and v in a geometric graph G in (X, δ), which might not be a spanner of (X, δ), is a t-spanner path if the total weight of edges of the path is at most $t\delta(u, v)$. A t-spanner G of X is an *f-vertex-fault-tolerant* (f-VFT) if for any set of points $F \subseteq X$ such that $|F| \leq f$, the graph $G - F$ obtained by removing F from G is still a t-spanner of $X \setminus F$.

Let (X, δ) be a metric of doubling dimension d. We denote by $\mathbf{B}(p, r)$ the set of points in X with distance at most r from p. The *spread* of P is the ratio between the maximum distance and minimum distance between points in P. We say that P is *r-separated* if the distance between every two distinct points in P is at least r. The following lemma is well known:

Lemma 1 (Packing bound). *Let $R \geq r > 0$ and Y be an r-separated set contained in a ball of radius R. Then, $|Y| \leq \left(\frac{4R}{r}\right)^d$.*

Net.   An r-net of a subset of points $P \in X$ is an r-separated subset N of P such that for each point v in P, there exists $u \in N$ such that $\delta(u, v) \leq r$.

3   VFT Spanner Construction Algorithm

In this section, we present a construction of an $(f + 1)$-VFT $(1 + \epsilon)$-spanner with degree $O_{\epsilon,d}(f)$ and lightness $O_{\epsilon,d}(f^2)$ as described in Theorem 1. We focus on presenting the ideas of the construction; a fast implementation will be delayed to Section 7.

3.1   Net tree, surrogate sets, and bipartite connections

By scaling, we assume that the minimum and maximum distance in X is 64 and Δ, respectively. We set the minimum distance to 64 to handle some corner cases more gratefully. Throughout this paper, for every positive integer a, we use the notation $\log a = \log_5 a$ to avoid the subscript.

Definition 1 (Greedy Net Tree). *Let $r_0 = 1, N_0 = X, \zeta = \lceil \log \Delta \rceil$, and $r_i = 5^i r_0$ for every $i \in [1, \zeta]$. Let $N_0 \supsetneq N_1 \supsetneq \ldots \supsetneq N_\zeta$ be a hierachy of nets where N_i is an r_i-net of N_{i-1}. The hierarchy of nets induces a hierarchical tree T where:*

1. *For every level $i \in [0, \zeta]$, there is one-to-one correspondence between nodes in at level i in T and points in N_i.*
2. *The parent in T of a point in N_{i-1} is the closest point in N_i for $1 \le i \le \zeta$, breaking tie arbitrarily.*

We refer to points in the nets and nodes in the tree T interchangeably. As a point u can belong to multiple nets, to avoid confusion, we sometimes write $(u, i) \in N_i$ to indicate the copy of u in N_i. For each node (u, i), the value r_i is called the radius of (u, i). In our net-tree, we use the radius $r_i = 5^i$ instead of 2^i as other works, e.g., [CGMZ16], because of a specific property that we need: the sum of radii associated with any set of nodes such that no two of them are ancestors of each other is bounded by a constant time the weight of the minimum spanning tree; see Lemma 8.

For a node $x = (u, i)$, we sometimes write x in place of u in, e.g., the distance function. For example $\delta(x, y)$ for two nodes $x = (u, i)$ and $y = (v, j)$ refers to $\delta(u, v)$. A point w is a leaf of x if $(u, 0)$ is a leaf of the subtree with root x. We use the notation $\mathrm{lvl}(x)$ for the level of x. (If $x = (u, i)$ then $\mathrm{lvl}(x) = i$.)

For each node $x = (u, i)$, let $L(x)$ or $L(u, i)$ be the set of leaves of x. The distance between any node to any of its descendants is at most a constant times the node's radius.

Claim 1. *Let (u, i) be a node in T and (v, j) be a descendant of (u, i) $(j < j)$, then $\delta(u, v) \le 5r_i/4$, implying that $\delta(u, v) \le 2r_i$.*

The second bound $\delta(u, v) \le 2r_i$ is usually used when we want to work with integers and the distance between a node to one of its leaves does not contribute significantly to the total distance.

Proof of Claim 1. By Definition 1, the distance between a node at level i to its child is at most r_i. Hence, $\delta(u, v) \le \sum_{k=j+1}^{i} r_k$. Using geometric sum,

$$\delta(u, v) \le \sum_{k=j+1}^{i} r_k \le \sum_{k=0}^{i} r_k = r_i(1 + 1/5 + 1/5^2 + \ldots) \le 5r_i/4, \tag{1}$$

as claimed. $\qquad\square$

A net tree is an important tool in almost all spanner constructions in doubling metric, e.g., see [CGMZ16]. In the construction of f-VT $(1 + \epsilon)$-spanner, Solomon [Sol14] introduced the notion of *surrogate sets*, which was used on top of the net-tree spanners.

Definition 2 (Surrogate Sets). *Each node $x \in T$ at level $i \in [0, \zeta]$ is associated with a set of points $S(x) \in X \subseteq \mathbf{B}(x, 16 \cdot r_i)$ of size at most $f + 1$ called a surrogate set.*

An useful operation that we will use on top of the surrogate sets is forming a bipartite connection.

Definition 3 (Bipartite Connection). *Let $S(x)$ and $S(y)$ be the surrogate sets of two differrent nodes x and y in T, a bipartite connection between them is a set of edges, denoted by $M(S(x), S(y))$, that is defined as folllows. If $\min\{|S(x)|, |S(y)|\|\} < f + 1$ then $M(S(x), S(y))$ includes all edges between $S(x)$ and $S(y)$, i.e, $M(S(x), S(y)) = S(x) \times S(y)$. Otherwise, by Definition 2, $|S(x)| = S(y) = f + 1$ and in this case, $M(S(x), S(y))$ is an (arbitrary) perfect matching between $S(x)$ and $S(y)$.*

3.2 The construction algorithm

Our construction is fairly simple compared to existing fault-tolerant spanner constructions. Let $\lambda = 5^{20}(1+1/\epsilon)$. Let x,y be two nodes of T at the same level i_{xy}. We say that (x,y) is a *cross edge* if $\delta(x,y) \le \lambda r_{i_{uv}}$. The notion of cross edges is central in all spanner constructions based on the net tree. If (x,y) is a cross edge, we say that x and y are *cross neighbors* of each other. Let $NC(x)$ be the set of cross neighbors of x, and $NC[x] = NC(x) \cup \{x\}$.

Let A be a set of nodes at some level i of T, we denote by $\mathrm{Cross}(A)$ be the set of cross edges between nodes in A.

For every node $x \in T$, let $T(x)$ be the subtree rooted at x of T. Let i_x be the level of x and $j \in [0,\zeta]$, we define $\mathrm{Aug}_j(x)$ as follows: if $j < i_x$, then $\mathrm{Aug}_j(x)$ is the union of $\mathrm{Cross}(NC[y])$ for all *descendant nodes* y at level j of x; otherwise, $\mathrm{Aug}_j(x) = \mathrm{Cross}(NC[y])$ where y is the ancestor at level j of x. We view $\mathrm{Aug}_j(x)$ as a set of *augmented cross edges* at level j of x. Let $\mathrm{Aug}(x,l,h) = \bigcup_{j=i_x+l}^{i_x+h} \mathrm{Aug}_j(x)$.

Let $(u,0)$ and $(v,0)$ be two nodes at level 0 of T. We say that $(\hat{u},\hat{v})$ is an *original cross edge* of $(u,0)$ and $(v,0)$ if $\hat{u}$ and $\hat{v}$ are two ancestors at *lowest level*, say i_{uv}, of $(u,0)$ and $(v,0)$ respectively such that $\delta(\hat{u},\hat{v}) \le \lambda r_{uv}$. It could be that $\hat{u} = (u,0)$ and $\hat{v} = (v,0)$.

Algorithm 1 describes the construction: it can be divided into two phases: Phase 1 is from lines 1 to 9 and Phase 2 from lines 10 to 22.

In the first phase, we start with a $(1+\epsilon)$-spanner G of (X,δ) with lightness $O(\epsilon^{-O(d)})$ that can be constructed in $O(n \log n)$ time [FS16]. The edges of G will serve as guidance for our construction, as we will later bound the lightness of the output VFT spanner H by charging to edges of G. The goal of this phase is to construct a set of cross edges E^*. Unlike other (both fault-tolerant and non-fault-tolerant) spanner constructions [Sol14, CGMZ16], where one would add a cross edge for *every pair of points in X*, we only add cross edges corresponding to edges of G. Specifically for each edge $(u,v) \in E(G)$, we first add to E^* the original cross edge, say $(\hat{u},\hat{v})$ of $(u,0)$ and $(v,0)$ (line 5). It is not hard to see that $\delta(\hat{u},\hat{v})$ is approximately $\delta(u,v)$. Next, we add to E^* the augmented cross edges from the ancestors that are within $O(\log \lambda)$ levels from $\hat{u}$ and $\hat{v}$. Note that: (i) we only add edges that are long enough (line 8) and (ii) by definition of $\mathrm{Aug}(\cdot,\cdot,\cdot)$, E^* includes not only cross edges incident to the ancestors of $\hat{u}$ and $\hat{v}$, but also those that are between the cross neighbors of the ancestors. As augmented cross edges are not much longer than $(\hat{u},\hat{v})$, we can later show that $\mathbf{w}(E^*) = \epsilon^{-O(d)}\mathbf{w}(G)$, which is a part of the proof of Lemma 5.

In the second phase, we use edges in E^* found in the first phase to add edges to the spanner H to guarantee the fault-tolerant property. Specifically, we visit the levels of T from lower to higher, and for each edge (x,y) in E^* at level i (set E_i^* in line 18), we would add a bipartite connection (Definition 3) between two surrogate sets $S(x)$ and $S(y)$ as in line 22. (Suppose for now that $S(x)$ and $S(y)$ are chosen arbitrarily following Definition 2). However, this is not enough. In particular, the nodes x that are marked as small in line 13 are problematic. That is if there exists a small descendant within $O(\log \lambda)$ levels of x that is small and has at most f leaves in its subtree (line 14 and line 15), then we may not able to find $f+1$ vertices for $S(x)$. (We later prove that the set of small nodes, each of which has at most f leaves, can be partitioned into LNF.) To fix this issue, we add (long enough) cross edges between nodes in $NC[x]$ in line 16 to E^*. Using the argument outlined in Section 1.4, which is the key technical contribution of our work, we are able to show that adding the bipartite connection between two surrogate sets $S(x)$ and $S(y)$ for every edge $(x,y) \in E^*$ at level i suffices to guarantee f-VFT, and furthermore, the spanner will have lightness $O(f^2)$. However, the degree could be $\Omega(n)$ if $S(x)$ and $S(y)$ are chosen arbitrarily.

Algorithm 1: Spanner Construction

Input : Doubling metric (X, δ) of dimension d
Output: An $(f+1)$-FT $(1+5\epsilon)$-spanner H with degree $O_{\epsilon,d}(f)$ and lightness $O_{\epsilon,d}(f^2)$

1 Construct a light $(1+\epsilon)$-spanner G of X and a net-tree T ;
2 $E^* \leftarrow \varnothing, H \leftarrow (X, \varnothing), \lambda = 5^{20}(1+1/\epsilon)$;
3 **for** $e = (u,v) \in E(G)$ **do**
4   Let $(\hat{u}, \hat{v})$ be the *original cross edge* of $(u,0)$ and $(v,0)$;
5     $E^* \leftarrow E^* \cup \{(\hat{u}, \hat{v})\}$;
6   **for** $(x,y) \in \text{Aug}(\hat{u}, 0, 5\log\lambda) \cup \text{Aug}(\hat{v}, 0, 5\log\lambda)$ **do**
7   Let i_{xy} be the level of x (and y) in T;
8   **if** $\delta(x,y) \geq 64 \cdot r_{i_{xy}}$ **then**
9           $E^* \leftarrow E^* \cup \{(x,y)\}$;

10 $\xi = \epsilon^{-O(d)}, c_1 = 50\log\lambda \cdot \xi, c_2 = 51\log\lambda \cdot \xi, c_3 = 55\log\lambda \cdot \xi$;
11 **for** $i \in [0 \ldots \zeta - 1]$ **do**
12   **for** *each node x at level i of T* **do**
13   Mark x as *small* if every leaf of x has degree at most $c_1 \cdot f$ in H ;
14        $D_x \leftarrow$ set of all descendants of x at levels from $i - 5\log\lambda$ to i;
15   **if** *there exists $w \in D_x$ s.t w is small and $T(w)$ has at most f leaves* **then**
16           $E^* \leftarrow E^* \cup \{(y,z) \mid y,z \in NC[x] \text{ and } 64r_i \leq \delta(y,z) \leq \lambda r_i\}$;

17   Mark all vertices with degree in H larger than $c_3 f$ as *saturated*;
18     $E_i^* \leftarrow$ set of edges $(x,y) \in E^*$ s.t both x and y are at level i;
19   **for** *each edge $(x,y) \in E_i^*$* **do**
20        $S(x) \leftarrow \text{SelectSurrogate}(x)$;
21        $S(y) \leftarrow \text{SelectSurrogate}(y)$;
22        $E(H) \leftarrow E(H) \cup M(S(x), S(y))$;

23 **return** H;

Algorithm 2 shows how to choose $S(x)$ carefully to reduce the degree all the way down to $O_{\epsilon,d}(f)$ while keeping the fault tolerant property in check. If the degree in H of a node is at least $c_3 \cdot f$, the algorithm will mark it as *saturated* in line 17, and it will not be used in any surrogate set of the nodes considered in future iterations. (Only edges incident to points in surrogate sets are added to H.) If x is small (line 1), then one can show that every leaf of $T(x)$ is not saturated by definition and hence can safely be added to S (line 2). Otherwise, we consider two sets $S' \subseteq \mathbf{B}(x, 16r_i)$ (line 4) and $S'' \subseteq \mathbf{B}(x, 4r_i)$ (line 5) containing unsaturated vertices . We prefer adding vertices of S' to $S(x)$ over vertices of S'' as those in S' are closer to being saturated. While it is not hard to see that our final spanner has degree $O_{\epsilon,d}(f)$ due to the choice of the surrogate sets, showing that it remains f-VFT is extremely challenging. Indeed, this is another major technical contribution of our paper.

The main result in our paper is that Algorithm 1 returns a light and bounded degree VFT spanner in optimal time.

Theorem 2. *The output graph H of Algorithm 1 is a f-VFT $(1+\epsilon)$-spanner of X with maximum degree $2c_3 f$ and lightness $O_{\epsilon,d}(f^2)$. Furthermore, Algorithm 1 can be implemented to run in*

Algorithm 2: SelectSurrogate

Input : A node $x \in V(T)$

Output: A surrogate set $S(x)$

1 if x *is small* **then**

2 $S(x) \leftarrow$ arbitrary $f + 1$ leaves of $T(x)$ if $T(x)$ has at least $f + 1$ leaves; otherwise
      $S(x) \leftarrow$ all leaves of $T(x)$;

3 else

4 $S' \leftarrow \{u \in X \cap \mathbf{B}(x, 16r_i) : \deg_H(u) \geq c_2 \cdot f$ and u is not saturated$\}$;

5 $S'' \leftarrow \{u \in X \cap \mathbf{B}(x, 4r_i) : \deg_H(u) < c_2 \cdot f\}$;

6   **if** $|S'| \geq f + 1$ **then**

7 $S(x) \leftarrow$ arbitrary $f + 1$ vertices in S';

8   **else**

9 $S(x) \leftarrow S' \cup \{$arbitrary $f + 1 - |S'|$ vertices in $S''\}$;

10 return $S(x)$;

$O(n(\log n + f))$ *times.*

We show that H has maximum degree $2c_3 f$ in Section 4, $O(f^2)$ lightness in Section 5 and prove that H is f-vertex fault tolerant in Section 6. All together prove Theorem 2. We construct a fast implementation of our algorithm in Section 7.

4 Degree Analysis

Observe from Algorithm 1's for loop (line 11) that before iteration i, each non saturated point has a degree less than $c_3 f$. After being saturated, the degree of a point does not increase. To show that H has a bounded maximum degree, it remains to prove that at the last iteration before any point u become saturated, the degree of u increases by at most $O_{\epsilon,d}(f)$. Indeed, we prove a stronger result: the degree of each point increases by at most $O_{\epsilon,d}(f)$ after each iteration of the for loop in line 11.

Since after adding a complete bipartite connection to H, the degree of each point increases by at most f, we need to show that any point u is in a constant number of surrogate sets (with multiplicity) at level i. Note that u can belong to the surrogate set of a node x multiple times. An edge in H is a level-i edge for a non-negative integer $i \leq \zeta$ if it is in $M(S(x), S(y))$ for some $x, y \in V(T)$. If an edge is added to H at multiple levels, we choose the lowest one.

Lemma 2. *Every point* $v \in X$ *belongs to* $O_{\epsilon,d}(1)$ *surrogate sets (with multiplicity) at level i for every* $i \in [0 \ldots \zeta]$. *Furthermore, v is incident to at most* ξf *level-i edges in H with* $\xi = O(1/\epsilon)^d$ *chosen in line 10.*

To prove Lemma 2, we first show that any node in $V(T)$ is only incident to $O_{\epsilon,d}(1)$ edges in E^*.

Observation 2. *Every node* (u, i) *is incident to* $O(1/\epsilon)^d$ *cross edges in* E^*.

Proof. Let $N(u, i) = \{v_1, v_2, \ldots\}$ be the set of cross neighbors of (u, i). By Algorithm 1, for every $v_j \in N(u, i)$, $\delta(v_j, u) \leq \lambda r_i$, implying that $v_j \in \mathbf{B}(u, \lambda r_i)$. Additionally, $N(u, i)$ is an r_i-separated

set since it is a subset of an r_i-net. Hence, by packing bound (Lemma 1), $|N(u,i)| \le \left(\frac{4\lambda r_i}{r_i}\right)^d = O(1/\epsilon)^d$. □

We now prove Lemma 2.

Proof of Lemma 2. For any node (u,i) such that v is in the surrogate set of (u,i), $\delta(u,v) \le 16r_i$, implying that $u \in \mathbf{B}(v,16r_i)$. By Observation 2, for each u, v is chosen to be a surrogate of (u,i) for $O(1/\epsilon)^d$ times. Let $S_v = \{u_1,u_2,\ldots\}$ be the set of points in X satisfying for each $u_k \in S_v$, v is a surrogate of (u_k,i). By Algorithm 2, $\delta(v,u_k) \le 16r_i$ for every $u_k \in S_v$. Since S_v is a subset of an r_i-net, $|S_v| \le \left(\frac{4 \times 16 r_i}{r_i}\right)^d = 2^{6d}$ by Lemma 1.

Because each node (u_k,i) $(u_k \in S_v)$ is incident to $O(1/\epsilon)^d$ cross edges and adding the bipartite connection of each cross edge increases the degree of a point in $S(u_k,i)$ by at most f in line 22, there are $2^{6d} \cdot O(1/\epsilon)^d \cdot f$ level-i edges incident to v. □

We are now ready to bound the degree of H.

Lemma 3. *H has a maximimum degree bounded by $2c_3 f$.*

Proof. Observe that when a point is saturated, it will never be used later in the algorithm. Thus, it is sufficient to bound the degree of any point after the last iteration when it is not marked as saturated in line 17 of Algorithm 1. For each point u, let i_u be the highest level such that u is not saturated. By Lemma 2, the degree of u in H increases by at most ξf after iteration i_u. Since u is not marked as saturated in line 17, the degree of u before we add the bipartite connection of cross edges at level i_u is at most $c_3 f$. Hence, the degree of u after we add level-i_u edges to H is at most $(c_3 + \xi)f \le 2c_3 f$ by the choice of c_3. □

5 Lightness

In this section, we analyze the lightness of H. Recall that for each edge (u,v) in G, we add the original cross edge $(\hat{u},\hat{v})$ of (u,v) and the cross edges in $\mathrm{Aug}(\hat{u},0,5\log\lambda) \cup \mathrm{Aug}(\hat{v},0,5\log\lambda)$ to E^*; let E_O^* be the set contains all these cross edges for every $(u,v) \in E(G)$. Let E_O be the set of edges added to H from every bipartite connection of all cross edges in E_O^*. Formally,

$$E_O = \{M(S(x),S(y)) : (x,y) \in E_O^*\} \qquad . \tag{2}$$

In Lemma 5 , we show that $\mathbf{w}(E_O) = O(f^2)\mathbf{w}(\mathrm{MST}(X))$. In Lemma 6, we show that $\mathbf{w}(E(H) \smallsetminus E_O) = O(f^2)\mathbf{w}(\mathrm{MST}(X))$. Given the two lemmas, we now prove:

Lemma 4. *H has lightness $\epsilon^{-O(d)}f^2$.*

Proof. Observe that $\mathbf{w}(E(H)) \le \mathbf{w}(E_O) + \mathbf{w}(E(H) \smallsetminus E_O)$. Since $\mathbf{w}(E_O) = \epsilon^{-O(d)}\mathbf{w}(\mathrm{MST}(X))$ by Lemma 5 and $\mathbf{w}(E(H) \smallsetminus E_O) = \epsilon^{-O(d)}\mathbf{w}(\mathrm{MST}(X))$ by Lemma 6, $\mathbf{w}(E(H)) \le \epsilon^{-O(d)}\mathbf{w}(\mathrm{MST}(X))$, which implies the lemma. □

First, we list some properties that we need for the proofs of Lemma 5 and Lemma 6.

Property 1. *We have the following properties:*

1. *Let (x,y) be a cross edge in E^*. For every $(u,v) \in M(S(x),S(y))$, $\delta(u,v) \le 2\delta(x,y)$.*

14

2. *Let x be an arbitrary level-i node in T. For every non-negative integer γ, $\mathbf{w}(\text{Aug}(x,0,\gamma)) = \epsilon^{-O(d)}5^\gamma r_i$. Recall that $\text{Aug}(x,0,\gamma) = \bigcup_{j=i}^{i+\gamma}\text{Aug}_j(x)$ with $\text{Aug}_j(x)$ $(j \geq 0)$ is the set of cross edges between cross neighbors of the ancestor of x at level j.*

3. *For every original cross edge $(\hat{u},\hat{v})$ at level i, $\delta(\hat{u},\hat{v}) \geq \lambda r_i/6$.*

4. *Let $e = (u,v)$ be a pair of points in X and $(\hat{u},\hat{v})$ be the original cross edge of e. Then, $\delta(\hat{u},\hat{v}) \leq (1+\epsilon)\delta(u,v)$.*

Proof. Item 1: Let i be the level of x and y. Note that $\delta(x,u),\delta(y,v) \leq 16r_i$ by Definition 2. By triangle inequality, $\delta(u,v) \leq \delta(x,y) + \delta(x,u) + \delta(y,v) \leq \delta(x,y) + 32r_i \leq 2\delta(x,y)$ since $\delta(x,y) \geq 64r_i$ by the choice of cross edges in E^*.

Item 2: Let k be a level such that $k \geq i$. The weight of a cross edge at level k is at most $\lambda r_k = \lambda r_i \cdot 5^{k-i}$ by the definition of cross edges. Let x_k be the ancestor of x at level k. Recall that $\text{Aug}_k(x)$ is the set of cross edges with both ends in $NC[x_k]$. Since $NC[x_k]$ is a subset of a r_k-net with diameter $2\lambda r_k$, $|NC[x_k]| \leq (4\lambda)^d = \epsilon^{-O(d)}$ by packing bound (Lemma 1). Then, $|\text{Aug}_k(x)| \leq |NC[x_k]|^2 = \epsilon^{-O(d)}$. Hence,

$$\mathbf{w}(\text{Aug}(x,0,\gamma)) = \sum_{k=i}^{i+\gamma}\mathbf{w}(\text{Aug}_k(x)) = \epsilon^{-O(d)}\lambda r_i \cdot \sum_{k=i}^{i+\gamma}5^{k-i} \leq 5^\gamma\epsilon^{-O(d)}r_i \quad,$$

as claimed.

Item 3: Let (u,v) be the pair of points in X of which $(\hat{u},\hat{v})$ is the original cross edge. Let $(u',i-1)$ and $(v',i-1)$ be the ancestors at level $i-1$ of $(u,0)$ and $(v,0)$, respectively. Since $\hat{u}$ and $\hat{v}$ are parents of $(u',i-1)$ and $(v',i-1)$, $\delta(\hat{u},u'),\delta(\hat{v},v') \leq r_i$. By the minimality of i, $\delta(u',v') > \lambda r_{i-1}$. Using the triangle inequality, we have:

$$\delta(\hat{u},\hat{v}) \geq \delta(u',v') - (\delta(\hat{u},u') + \delta(\hat{v},v')) \geq \lambda r_{i-1} - 2r_i \geq (\lambda/5 - 2)r_i > \lambda r_i/6 \quad,$$

as $\lambda = 5^{20}(1+\epsilon^{-1})$.

Item 4: Let i be the level of $(\hat{u},\hat{v})$. By triangle inequality, $\delta(\hat{u},\hat{v}) \leq \delta(u,v) + \delta(\hat{u},u) + \delta(\hat{v},v)$. Recall from Claim 1 that $\delta(\hat{u},u),\delta(\hat{v},v) \leq 5r_i/4$. Therefore,

$$\frac{\delta(u,v)}{\delta(\hat{u},\hat{v})} \geq \frac{\delta(\hat{u},\hat{v}) - 5r_i/4 - 5r_i/4}{\delta(\hat{u},\hat{v})} \geq 1 - \frac{5r_i}{2\delta(\hat{u},\hat{v})} \geq 1 - \frac{15}{\lambda} \quad,$$

since $\delta(\hat{u},\hat{v}) \geq \lambda r_i/6$ by Item 4. Hence, $\delta(\hat{u},\hat{v}) \leq \frac{\delta(u,v)}{1-15/\lambda} \leq (1+\epsilon)\delta(u,v)$. $\qquad\square$

5.1 Weight of edges due to light spanner

Lemma 5. $\mathbf{w}(E_O) = \epsilon^{-O(d)}f^2\mathbf{w}(\text{MST}(X))$.

Proof. For each cross edge $(x,y) \in E_O^*$, the bipartite connection $M(S(x),S(y))$ contains at most $(f+1)^2$ edges, each of those has weight at most $2\delta(x,y)$ by Item 1 of Property 1. Hence, by the definition of E_O in Equation (2), $\mathbf{w}(E_O) \leq 2(f+1)^2\mathbf{w}(E_O^*)$. It remains to bound $\mathbf{w}(E_O^*)$.

Let O be the set of original cross edges of edges in $E(G)$. By Item 4 of Property 1, $\mathbf{w}(O) \leq (1+\epsilon)\mathbf{w}(G)$. Observe that:

$$E_O^* \subseteq \bigcup_{(\hat{u},\hat{v})\in O}[\{(\hat{u},\hat{v})\} \cup \text{Aug}(\hat{u},0,5\log\lambda) \cup \text{Aug}(\hat{v},0,5\log\lambda)]$$

15

Recall that for each node x, $\mathrm{lvl}(x)$ is the level of x. We have:

$$
\begin{aligned}
\mathbf{w}(E_O^*) &\leq \sum_{(\hat{u},\hat{v}) \in O} \left(\delta(\hat{u},\hat{v}) + \mathbf{w}(\mathrm{Aug}(\hat{u},0,5\log\lambda)) + \mathbf{w}(\mathrm{Aug}(\hat{v},0,5\log\lambda)) \right) \\
&\leq \sum_{(\hat{u},\hat{v}) \in O} \left(\delta(\hat{u},\hat{v}) + 2 \cdot 5^{5\log\lambda} \epsilon^{-O(d)} r_{\mathrm{lvl}(\hat{u})} \right) \qquad \text{(by Item 2 of Property 1)} \\
&\leq \sum_{(\hat{u},\hat{v}) \in O} \left(\delta(\hat{u},\hat{v}) + 2 \cdot \lambda^5 \epsilon^{-O(d)} \frac{\delta(\hat{u},\hat{v})}{\lambda/4} \right) \qquad \text{(by Item 3 of Property 1)} \\
&\leq \sum_{(\hat{u},\hat{v}) \in O} \epsilon^{-O(d)} \delta(\hat{u},\hat{v}) = \epsilon^{-O(d)} \mathbf{w}(O) \\
&\leq \epsilon^{-O(d)} \mathbf{w}(G) \leq \epsilon^{-O(d)} \mathbf{w}(\mathrm{MST}(X)) \qquad \text{(since G is a light spanner of X),}
\end{aligned}
$$

as desired. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

5.2 Weight of remaining edges

Throughout this section, we show that:

Lemma 6. $\mathbf{w}(E(H) \smallsetminus E_O) = \epsilon^{-O(d)} f^2 \mathbf{w}(\mathrm{MST}(X))$.

We partition $E(H) \smallsetminus E_O$ into two sets E_{inc} and E_{com}, whose formal definition will be given later. We then bound the total weight of each set in Lemma 9 and Lemma 10. Recall that each edge in $E(H) \smallsetminus E_O$ is added to H by some bipartite connection of cross edges in $E^* \smallsetminus E_O^*$. Those cross edges are added to E^* by line 15–16.

To formally define E_{inc} and E_{com}, we need more notation. A node is *large* if it is not small, i.e., there exists a leaf in its subtree with degree at least $c_1 f$. A node is *incomplete* if it is small and has at most f leaves, otherwise it is *complete*. That is a node is complete if it is either large or has at least $f+1$ leaves. By line 15–16, if a node x is incomplete or has an incomplete descendant within $5\log\lambda$ levels, we add all long enough cross edges from $\mathrm{Cross}(NC[x])$ to E^*. Recall that $\mathrm{Cross}(NC[x])$ is the set of cross edges with both ends in $NC[x]$. Hence, for each cross edge (y,z) in $E^* \smallsetminus E_O^*$, both y and z are cross neighbors of some node x such that x is either incomplete or is an ancestor within $5\log\lambda$ level of an incomplete node. Let E_{com}^* be the set of cross edges in $E^* \smallsetminus E_O^*$ having both complete end nodes. More formally, $E_{com}^* = \{(x,y) \mid (x,y) \in E^* \smallsetminus E_O^* \text{ and } x,y \text{ are complete}\}$. Let $E_{inc}^* = E^* \smallsetminus (E_O^* \cup E_{com}^*)$, meaning that E_{inc}^* contains every cross edge in $E^* \smallsetminus E_O^*$ that has at least one incomplete end node. We denote by E_{com} and E_{inc} the sets of edges added to H by the bipartite connection of edges in E_{com}^* and E_{inc}^*. Formally,

$$
E_{com} = \{M(S(x),S(y)) : (x,y) \in E_{com}^*\} \quad \text{and} \quad E_{inc} = \{M(S(x),S(y)) : (x,y) \in E_{inc}^*\}. \tag{3}
$$

The key to our proof is the following lemma:

Lemma 7. *If a node x in T is complete, $S(x)$ always has size $f+1$.*

While the statement of Lemma 7 is simple and easy to understand, the proof of Lemma 7 is intricate. Here, we sketch the ideas of our proof. The full proof is deferred to Section 5.2.1. If x is small, we know that the subtree at root x has at least $f+1$ leaves with a low degree. Therefore,

there are always $f + 1$ points with a low degree to be added to $S(x)$. If x is large, then one of its leaves, say v, is incident to many edges. We keep track of the degree change of points close to v. By Lemma 2, the degree of v in level $i - 2\log\lambda$ is still significantly large (at least $48\log\lambda \cdot \xi f$), then for any edge (v,w) incident to a point close to v, all the low degree points close to w will be in $\mathbf{B}(x, 4r_i)$. We prove after the iterations from $i - 2\log\lambda$ to $i - 1$, there are still $4f + 4$ points with degree less than $c_2 f$ in $\mathbf{B}(x, 4r_i)$. For this to hold, we have to choose the surrogates carefully, which explains the choice of points with high and low degrees in Algorithm 2. Hence, there are always enough candidates for $S(x)$, which will prove Lemma 7.

Observe that the set of incomplete nodes in T can be viewed as a set of disjoint subtrees of T. We call those subtrees a light net-forest (LNF).

Definition 4. *A light net-forest of T with respect to G, denoted by $\mathrm{LNF}_G(T)$, is a subgraph of T whose vertex set containing all incomplete nodes in T and edge set is the set of edges in T between any two incomplete nodes. If the net-tree T and the light spanner G are clear in the context, we use the notation* LNF.

By the definition of E^*_{inc}, every cross edge in E^*_{inc} is incident to at least one node in LNF. Then, we will bound $\mathbf{w}(E^*_{inc})$ by $\mathbf{w}(\mathrm{LNF})$. To do that, we first claim that LNF is a set of (rooted) subtrees of T.

Observation 3. *Every leaf node of T is in* LNF.

Proof. By definition of a small node, every leaf node $(u, 0)$ is small since the degree of every u before the first iteration is $0 < c_1 f$. Since $(u, 0)$ has only one leaf in its subtree (which is itself), $(u, 0)$ is incomplete. $\square$

The following claim implies that the ancestor of every almost complete node is complete.

Claim 2. *The parent of a complete node is complete.*

Proof. For each point $v \in X$ and each level $i \in [0, \zeta]$, let $\deg_{<i}(v)$ be the degree of v before i^{th} iteration. Let x be a complete node and p be the parent of x. If x is large, then by definition of a large node, there exists a leaf v of $T(x)$ such that $\deg_{<i}(v) \geq c_1 f$. Therefore, since $\deg_{<i}(v)$ is non-decreasing as level i gets larger, p also has a leaf v of degree at least $c_1 f$, implying that p is large and thus complete.

If x is small, then $T(x)$ has at least $f + 1$ leaves. If p is small, $T(p)$ also has at least $f + 1$ leaves since p is the parent of x, implying that p is complete by definition. If p is large then p is also complete. $\square$

From Claim 2, we obtain that the LNF contains a set of node-disjoint subtrees of T. Hence, we will bound the weight of cross edges incident to nodes in LNF by the total radius of the roots, which are called almost complete nodes. Formally, an *almost complete* node is an incomplete node whose parent is complete. From Claim 2, any two almost complete nodes do not have the ancestor-descendant relationship. Thus, there is no almost complete node that is an ancestor/descendant of another almost complete node, meaning that the subtrees in LNF are node-disjoint.

By a relatively simple argument, one can show that the total radius of the root nodes of the LNF is a good approximation of the weight of $\mathrm{MST}(X)$. Note that Lemma 8 is not true for arbitrary construction of the net-tree T. We need the property that our net-tree is constructed by a greedy method, meaning the parent of a net-point at level i is its closest net-point at level $i + 1$.

Lemma 8. *Let A be a subset of $V(T)$ such that there is no pair of nodes in A having the ancestor-descendant relationship. Then, $\sum_{x \in A} r_{\mathrm{lvl}(x)} = O(\mathbf{w}(\mathrm{MST}(X)))$.*

Proof. The crucial property to prove Lemma 8 is the greedy property of the net-tree. Recall that in our net-tree construction (Definition 1), the parent of a node (u, i) is the one corresponding to the closest point to u in N_{i+1}. Let $\mathcal{B} = \{\mathbf{B}(x, r_{\mathrm{lvl}(x)}/5) : x \in A\}$.

We claim that every point $v \in X$ is in at most one ball in $\mathcal{B}$. Since no node in A is an ancestor of another, it is sufficient to show that if a ball $\mathbf{B}(x, r_{\mathrm{lvl}(x)}/5)$ in $\mathcal{B}$ contains v for some x, v has to be a leaf of x. Then, all nodes x such that $\mathbf{B}(x, r_{\mathrm{lvl}(x)}/5)$ containing v must lie on the path from $(v, 0)$ to the root of T, and only one node in that path can belong to A.

Assume that there exists a node $x = (u, i)$ such that $\mathbf{B}(u, r_i/5)$ contains v and v is not a leaf of (the subtree rooted) x. Let $y = (w', i-1)$ and $z = (w, i)$ be the ancestor of $(v, 0)$ at level $i-1$ and i, respectively. By our choice of children for each node in the net, $\delta(u, w') \geq \delta(w, w')$, implying that:

$$\delta(u, w') \geq (\delta(u, w') + \delta(w', w))/2 \geq \delta(u, w)/2 \geq r_i/2 \qquad . \tag{4}$$

On the otherhand, by Claim 1, $\delta(w', v) \leq 5r_{i-1}/4$. Using the triangle inequality, we obtain:

$$\delta(u, w') \leq \delta(u, v) + \delta(v, w') \leq r_i/5 + 5r_{i-1}/4 = 9r_i/20 < r_i/2 \qquad ,$$

contradicting to Equation (4).

Therefore, each point in X belongs to at most one ball in $\mathcal{B}$. Thus, for any two nodes $x, y \in A$, $\delta(x, y) \geq \max\{r_{\mathrm{lvl}(x)}, r_{\mathrm{lvl}(y)}\}/5$.

Let X_A be the set of points in A (we translate each node in A by its representative in X). By a folkore result, $\mathbf{w}(\mathrm{MST}(X_A)) \leq 2 \cdot \mathbf{w}(\mathrm{MST}(X))$. For each node $x \in A$, let $e(x)$ be an arbitrary edge incident to x in $\mathrm{MST}(X_A)$. Hence, $\mathbf{w}(e(x)) \geq r_{\mathrm{lvl}(x)}/5$, implying that:

$$\sum_{x \in A} \mathbf{w}(e(x)) \geq \sum_{x \in A} r_{\mathrm{lvl}(x)}/5 \qquad . \tag{5}$$

On the other hand, since each edge in $\mathrm{MST}(X_A)$ is incident to at most 2 nodes in A, we have:

$$\sum_{x \in A} \mathbf{w}(e(x)) \leq 2 \sum_{e \in E(X_A)} \mathbf{w}(e) \leq 2\mathbf{w}(\mathrm{MST}(X_A)) \qquad . \tag{6}$$

By Equation (5) and Equation (6), $\sum_{x \in A} r_{\mathrm{lvl}(x)}/5 \leq 2\mathbf{w}(\mathrm{MST}(X_A)) \leq 4\mathbf{w}(\mathrm{MST}(X))$. Therefore, $\sum_{x \in A} r_{\mathrm{lvl}(x)} = O(\mathbf{w}(\mathrm{MST}(X)))$.

$\square$

We are now ready to bound $\mathbf{w}(E_{inc})$ (in Lemma 9) and $\mathbf{w}(E_{com})$ (in Lemma 10).

Lemma 9. $\mathbf{w}(E_{inc}) = \epsilon^{-O(d)} f^2 \mathbf{w}(\mathrm{MST}(X))$.

Proof. Let A be the set of roots of subtrees in LNF (or the set of almost complete nodes of T). There is no node in A being the ancestor of another. Hence, by Lemma 8,

$$\sum_{x \in A} r_{\mathrm{lvl}(x)} = O(\mathbf{w}(\mathrm{MST}(X))). \tag{7}$$

By Definition 4 and Claim 2, each incomplete node has exactly one almost complete ancestor in A. For each point $u \in X$, let i_u be the level of the root of the subtree in LNF containing $(u, 0)$. Let

18

$E_{inc}(u)$ be the set of all edges with level from 0 to i_u incident to u in E_{inc}, then $E_{inc} = \bigcup_{u \in X} E_{inc}(u)$. Note that there might be other edges in E_{inc} that are incident to u; however, each of those edges must be incident to some vertex v with i_v larger than the level of that edge.

By Lemma 2, u is incident to at most ξf level-i edges for every $0 \le i \le \zeta$. Hence, the total weight of level-i edges incident to u is at most $\xi f \cdot \lambda r_i$. This gives:

$$\mathbf{w}(E_{inc}(u)) \le \sum_{0 \le i \le i_u} \xi f \cdot \lambda r_i \le 2\lambda\xi f \cdot r_{i_u}$$

by geometric sum.

As $E_{inc} = \bigcup_{u \in X} E_{inc}(u)$, $\mathbf{w}(E_{inc}) \le \sum_{u \in X} 2\lambda\xi f \cdot r_{i_u}$. By Observation 3, we partition the set X into the representatives of leaves within the same subtree in LNF. Thus,

$$\mathbf{w}(E_{inc}) \le 2\lambda\xi f \cdot \sum_{u \in X} r_{i_u} \le \epsilon^{-O(d)} f \cdot \sum_{x \in A} \sum_{u \in L(x)} \underbrace{r_{i_u}}_{=r_{\mathrm{lvl}(x)}} \le \epsilon^{-O(d)} f \cdot \sum_{x \in A} |L(x)| \cdot r_{\mathrm{lvl}(x)} \tag{8}$$

Since each node in LNF is incomplete, each of its subtrees has at most f leaves by the definition of incomplete. Hence, from Equation (8), we have:

$$\mathbf{w}(E_{inc}) \le \epsilon^{-O(d)} f \cdot \sum_{x \in A} |L(x)| \cdot r_{\mathrm{lvl}(x)} \le \epsilon^{-O(d)} f^2 \cdot \sum_{x \in A} r_{\mathrm{lvl}(x)}$$

$$\le \epsilon^{-O(d)} f^2 \cdot \mathbf{w}(\mathrm{MST}(X)) \qquad \text{(by Equation (7))},$$

as claimed. $\qquad \square$

Lemma 10. $\mathbf{w}(E_{com}) = \epsilon^{-O(d)} f^2 \mathbf{w}(\mathrm{MST}(X))$.

Proof. We bound the weight of E_{com} by the weight of E^*_{com}. Since the bipartite connection of each cross edge (y, z) in E^*_{com} has exactly $f + 1$ edges. Each edge in $M(S(y), S(z))$ has both ends in $\mathbf{B}(y, 16r_{\mathrm{lvl}(y)})$ and $\mathbf{B}(z, 16r_{\mathrm{lvl}(z)})$ ($\mathrm{lvl}(y) = \mathrm{lvl}(z)$), and hence has weight at most $(\lambda + 32)r_{\mathrm{lvl}(y)}$ by the triangle inequality. We obtain that:

$$\begin{aligned}
\mathbf{w}(E_{com}) &\le \sum_{(y,z) \in E^*_{com}} (f+1)(\lambda + 32)r_{\mathrm{lvl}(y)} \\
&\le \sum_{(y,z) \in E^*_{com}} (f+1)(\lambda + 32)\delta(y,z)/64 \qquad \text{(since } \delta(y,z) \ge 64r_{\mathrm{lvl}(y)}\text{)} \\
&\le \sum_{(y,z) \in E^*_{com}} (f+1)\lambda\delta(y,z) = (f+1)\lambda\mathbf{w}(E^*_{com})
\end{aligned} \tag{9}$$

By line 15–16, every cross edge(y, z) in $E^* \smallsetminus E^*_O$ is in $\mathrm{Cross}(NC[x])$ for some node x such that x is either in the LNF or x is an ancestor within $5\log\lambda$ levels of the root of some subtree in LNF. Let $A = \{a_1, a_2, \dots a_k\}$ be the set contains all roots of subtrees in LNF. By Lemma 8,

$$\sum_{i=1}^{k} r_{\mathrm{lvl}(a_i)} = O(\mathbf{w}(\mathrm{MST}(X))). \tag{10}$$

We have:

$$E^* \smallsetminus E^*_O \subseteq \bigcup_{x \in A} \mathrm{Aug}(x, 0, 5\log\lambda) \bigcup \bigcup_{y \in \mathrm{LNF}} \mathrm{Cross}(NC[y]).$$

Since $E^*_{com} = E^* \smallsetminus (E^*_O \cup E^*_{inc}) \subseteq E^* \smallsetminus E^*_O$, we have the following bound on $\mathbf{w}(E^*_{com})$:

$$\mathbf{w}(E^*_{com}) \leq \mathbf{w}(E^* \smallsetminus E^*_O) \leq \sum_{x \in A} \mathbf{w}(\mathrm{Aug}(x, 0, 5\log\lambda)) + \sum_{y \in LNF} \mathbf{w}(\mathrm{Cross}(NC[y])). \qquad (11)$$

The reason behind the exclusion of E^*_{inc} is to make sure that each bipartite connection of cross edges in E^*_{com} contains exactly $f + 1$ edges in H.

By Item 2 of Property 1, we get:

$$\mathbf{w}(\mathrm{Aug}(x, 0, 5\log\lambda) \smallsetminus E^*_{inc}) \leq \mathbf{w}(\mathrm{Aug}(x, 0, 5\log\lambda)) = \epsilon^{-O(d)} r_{\mathrm{lvl}(x)} \qquad . \qquad (12)$$

We then bound $\sum_{y \in LNF} \mathbf{w}(\mathrm{Cross}(NC[y]))$. By the packing bound (Lemma 1), $NC[y]$ contains at most $\epsilon^{-O(d)}$ nodes. Thus, $\mathrm{Cross}(NC[y])$ contains at most $\epsilon^{-O(d)}$ cross edges, each of them has weight $\lambda r_{\mathrm{lvl}(y)}$. Thus,

$$\sum_{y \in LNF} \mathbf{w}(\mathrm{Cross}(NC[y])) \leq \sum_{y \in LNF} \lambda r_{\mathrm{lvl}(y)}. \qquad (13)$$

We partition the nodes in LNF into the set of nodes in subtrees of T. Let $\{T_1, T_2, \ldots T_k\}$ is the set of subtrees in T with root $\{a_1, a_2, \ldots a_k\}$ respectively ($T_1, T_2, \ldots T_k$ are node-disjoint). Then $LNF = T_1 \cup T_2 \cup \ldots T_k$. For each $i \in [1, k]$, let $T_{i,j}$ be the set of nodes of T_i at level j. Let $A = \{a_1, a_2, \ldots a_k\}$ be roots of $T_1, T_2, \ldots T_k$.

Since each subtree has at most f leaves, $|T_{i,j}| \leq f$ for every $i \in [1, k]$ and $j \leq \mathrm{lvl}(a_i)$. We have:

$$
\begin{aligned}
\sum_{y \in LNF} \lambda r_{\mathrm{lvl}(y)} &= \lambda \sum_{i=1}^{k} \sum_{y \in T_i} r_{\mathrm{lvl}(y)} = \lambda \sum_{i=1}^{k} \sum_{j=0}^{\mathrm{lvl}(a_i)} \sum_{y \in T_{i,j}} r_{\mathrm{lvl}(y)} = \lambda \sum_{i=1}^{k} \sum_{j=0}^{\mathrm{lvl}(a_i)} \sum_{y \in T_{i,j}} r_j \\
&= \lambda \sum_{i=1}^{k} \sum_{j=0}^{\mathrm{lvl}(a_i)} |T_{i,j}| r_j \leq \lambda \sum_{i=1}^{k} \sum_{j=0}^{\mathrm{lvl}(a_i)} f r_j \qquad (\text{since } |T_{i,j}| \leq f) \\
&\leq \lambda f \sum_{i=1}^{k} 2 r_{\mathrm{lvl}(a_i)} \qquad (\text{by geometric sum}) \\
&\leq \lambda f \cdot \mathbf{w}(\mathrm{MST}(X)) \qquad (\text{by Equation (10)}).
\end{aligned} \qquad (14)
$$

By Equation (9) and Equation (11), we have:

$$
\begin{aligned}
\mathbf{w}(E_{com}) &\leq \lambda(f+1) \cdot \mathbf{w}(E^*_{com}) \leq \lambda(f+1) \left[\sum_{x \in A} \mathbf{w}(\mathrm{Aug}(x, 0, 5\log\lambda)) + \sum_{y \in LNF} \mathbf{w}(\mathrm{Cross}(NC[y])) \right] \\
&\leq \lambda(f+1) \left[\sum_{x \in A} \epsilon^{-O(d)} r_{\mathrm{lvl}(x)} + \sum_{y \in LNF} \mathbf{w}(\mathrm{Cross}(NC[y])) \right] \qquad (\text{by Equation (12)}) \\
&\leq \lambda(f+1) \left[\sum_{x \in A} \epsilon^{-O(d)} r_{\mathrm{lvl}(x)} + \sum_{y \in LNF} \lambda r_{\mathrm{lvl}(y)} \right] \qquad (\text{by Equation (13)}) \\
&\leq \lambda(f+1) \left[\epsilon^{-O(d)} \mathbf{w}(\mathrm{MST}(X)) + \lambda f \cdot \mathbf{w}(\mathrm{MST}(X)) \right] \qquad (\text{by Eq. 10 and Eq. 14}) \\
&= \epsilon^{-O(d)} f^2 \cdot \mathbf{w}(\mathrm{MST}(X)),
\end{aligned}
$$

as desired. $\qquad\qquad\square$

We now ready to prove Lemma 6.

Proof of Lemma 6. Observe that $E(H) \smallsetminus E_O = E_{com} \cup E_{inc}$, we obtain $\mathbf{w}(E(H) \smallsetminus E_O) \leq \mathbf{w}(E_{com}) + \mathbf{w}(E_{inc}) = \epsilon^{-O(d)} f^2 \cdot \mathbf{w}(\mathrm{MST}(X))$ by Lemma 9 and Lemma 10. $\qquad\square$

5.2.1 Proof of Lemma 7

In this section, we provide a detailed proof of Lemma 7. First, we introduce some notation. A point v is clean if it has a degree of at most $c_2 f$ and is semi-saturated if it is not marked as saturated and has a degree larger than $c_2 f$. Point v is saturated if it is marked as saturated in line 17 of Algorithm 1. Note that "clean" and "semi-saturated" are *time-sensitive* properties, meaning that they change over time. Specifically, a vertex may change from clean to semi-saturated after the execution of line 22 of Algorithm 1. In this section, we say a point is "clean" or "semi-saturated" with respect to some specific moment while running Algorithm 1; this usually happens when we select surrogates in Algorithm 2 or before/after the execution of line 22 in some iteration.

For every node (u, i), recall that we choose the surrogate set of (u, i) by:

- If (u, i) is small and incomplete, $S(u, i) = L(u, i)$ where $L(u, i)$ is the set of leaves of the subtree of T with root (u, i).

- If (u, i) is small and complete, $S(u, i)$ contains $f + 1$ arbitrary non-saturated points in $L(u, i)$.

- If (u, i) is large, we find all semi-saturated points in $\mathbf{B}(u, 16r_i)$ and select arbitrary $f + 1$ of them to $S(u, i)$. If there are not enough $f + 1$ such points, we add clean points in $\mathbf{B}(u, 4r_i)$ to $S(u, i)$ until $|S(u, i)|$ reaches $f + 1$.

To prove Lemma 7, we show that there are always more than $f + 1$ points in $\mathbf{B}(u, 4r_i)$ if (u, i) is large. Hence, the surrogate set of (u, i) always contains $f + 1$ points.

For each point v, let $\deg_{<i}(v)$ and $\deg_{\leq i}$ be the degree of v before and after the ith iteration, respectively. A point v is *i-clean* if $\deg_{<i}(v) \leq c_2 f$ and is *i-saturated* if $\deg_{<i}(v) > c_3 f$. Recall that a small node is complete if it has at least $f + 1$ leaves and a large node is always complete.

Lemma 11. *For every large node (u, i), the number of i-clean points in $\mathbf{B}(u, 4r_i)$ is at least $(4f + 4)$.*

We will show that Lemma 7 follows from Lemma 11.

Proof of Lemma 7. Let $x = (u, i)$ be a large node. We consider two cases:

Case 1. If x is small, then x has at least $f + 1$ leaves by the definition of a complete node. We claim that there are $f + 1$ non-saturated points in $L(u, i)$. For every leaf v of x, i.e., $v \in L(u, i)$, the degree of v before iteration i is at most $c_1 f$ since x is small. Hence, v is not marked as saturated before iteration i since $c_1 f \leq c_3 f$. (Note that v is not marked as saturated during the execution of level i.) Therefore, v is either clean or semi-saturated when we update $S(u, i)$. (In Algorithm 1, $S(u, i)$ might be updated multiple times.) Since all leaves of x are either clean or semi-saturated, there must be at least $f + 1$ non-saturated points in $L(u, i)$ as x has at least $f + 1$ leaves. By line 2 in Algorithm 2, $|S(u, i)| = f + 1$.

Case 2. If x is large, by Lemma 11, the number of i-clean points in $\mathbf{B}(u, 4r_i)$ is at least $4f + 4$. Let C be the set of clean points in $\mathbf{B}(u, 4r_i)$. During the execution of level i, the points in C are not marked as saturated. Therefore, they either remain clean or become semi-saturated while updating $S(u, i)$. From line 4–9, $S(u, i)$ is formed by choosing semi-saturated points in $\mathbf{B}(u, 16r_i)$

and clean points in $\mathbf{B}(u, 4r_i)$. As $C \subseteq \mathbf{B}(u, 4r_i) \subseteq \mathbf{B}(u, 16r_i)$, all points in C are eligible to be selected to $S(u, i)$. Thus, there are enough $f + 1$ "candidates" for surrogates in $S(u, i)$, implying $|S(u, i)| = f + 1$. $\qquad\square$

We now focus on proving Lemma 11. First, we list some properties of the net-tree T. Recall that for a given node (u, i), $D(u, i)$ is the set of descendants of (u, i).

Property 2. *We have the following properties:*

1. *Let (u, i) be an arbitrary complete node in T. For any $(v, j) \in D(u, i)$ and any $\alpha \geq 2$, $\mathbf{B}(v, \alpha r_j) \subseteq \mathbf{B}(u, \alpha r_i)$.*

2. *Let (u, i) be an arbitrary node in T, v be a point in $\mathbf{B}(u, 3r_i)$ and (v, w) be a level-k edge with $k \leq i - 2\log\lambda$. Then, $w \in \mathbf{B}(u, 4r_i)$.*

3. *For every level-i edge (u, v), $\delta(u, v) \geq 32r_i$.*

Proof. Item 1: Since $i > j$, we have $r_j \leq r_i/5$. By Claim 1, $\delta(u, v) \leq 5r_i/4$. Thus, for each point $w \in \mathbf{B}(v, \alpha r_j)$, we get:

$$\delta(u, w) \leq \delta(u, v) + \delta(v, w) \leq 5r_i/4 + \alpha r_j \leq (5/4 + \alpha/5)r_i \leq \alpha r_i \qquad , \tag{15}$$

since $\alpha \geq 2$.

Item 2: By triangle inequality, $\delta(v, w) \leq (\lambda + 32)r_k \leq \frac{\lambda+32}{\lambda^2}r_i \leq r_i/2$. Then, by the triangle inequality, $\delta(u, w) \leq \delta(u, v) + \delta(v, w) \leq 3r_i + r_i/2 \leq 4r_i$.

Item 3: Let $(\tilde{u}, \tilde{v})$ be the level-i cross edge such that $(u, v) \in M(S(\tilde{u}), S(\tilde{v}))$. By construction, $\delta(\tilde{u}, \tilde{v}) \geq 64r_i$. Since $u \in \mathbf{B}(\tilde{u}, 16r_i)$ and $v \in \mathbf{B}(\tilde{v}, 16r_i)$, we have $\delta(u, v) \geq \delta(\tilde{u}, \tilde{v}) - \delta(u, \tilde{u}) - \delta(v, \tilde{v}) \geq 32r_i$. $\qquad\square$

We show that for any set A of small diameter, any large node using a clean point in A as a surrogate must also use all semi-saturated points in A. This property is due to the fact that we prioritize the use of semi-saturated points over clean ones.

Claim 3. *Let i and i' be two levels such that $i' \geq i$, x be a node at level i' and $S(x)$ be the result of SelectSurrogate(x) (Algorithm 2). For any set A of diameter at most $8r_i$, if $S(x)$ contains any clean point in A, then $S(x)$ also contains all semi-saturated points in A.*

Proof. Since $|S(x)| \leq f + 1$, $|A \cap S(x)| \leq f + 1$. Let $x = (u, i')$ and v be a clean point in $A \cap S(x)$. Recall that in the construction of $S(x)$, first, we find all semi-saturated points in $\mathbf{B}(u, 16r_{i'})$. If there are not enough $f + 1$ points in $\mathbf{B}(u, 16r_{i'})$, we pick some clean points in $\mathbf{B}(u, 4r_{i'})$. Thus, $v \in \mathbf{B}(u, 4r_{i'})$. Since $\mathrm{diam}(A) \leq 8r_i$, for every $w \in A$,

$$\delta(w, u) \leq \delta(w, v) + \delta(u, v) \leq \mathrm{diam}(A) + 4r_{i'} \leq 8r_i + 4r_{i'} \leq 12r_{i'} \qquad , \tag{16}$$

implying that $A \subseteq \mathbf{B}(u, 16r_{i'})$. Let A_s be the set of semi-saturated points in A. Observe that $A_s \subset S(x)$ since otherwise, there is no clean point in $S(x)$. $\qquad\square$

Note that in the proof of Claim 3, we only need that for every node x with $i = \mathrm{lvl}(x)$, all the semi-saturated vertices in $\mathbf{B}(x, 12r_i)$ must be in $S(x)$ before any clean vertex is added to $S(x)$. Hence, we prioritize selecting semi-saturated vertices in $\mathbf{B}(x, 16r_i)$ over the clean vertices in

$\mathbf{B}(x, 4r_i)$ in line 4. Throughout Algorithm 1's analysis, the proof of Claim 3 is the first (and also the only) proof that requires some geometric property other than the radius of the set where we choose the semi-saturated vertices from in Algorithm 2 (which is $\mathbf{B}(x, 16r_i)$). Indeed, for all the other proofs in Section 4, Section 5 and Section 6, we only need: (i) all semi-saturated vertices in $S(x)$ have distance at most $16r_i$ from x and (ii) the set where we choose the semi-saturated vertices in line 5 contains the set where we choose clean vertices in line 4.

Remark 1. *Claim 3 still holds if we replace the $\mathbf{B}(x, 16r_i)$ (i is the level of x) in line 4 of Algorithm 2 by any subset of $\mathbf{B}(x, 16r_i)$ containing $\mathbf{B}(x, 12r_i)$. Furthermore, the correctness of Algorithm 1 still holds if we choose the semi-saturated vertices in S' (in line 4 of Algorithm 2) from any subset of $\mathbf{B}(x, 16r_i)$ containing $\mathbf{B}(x, 12r_i)$.*

By Claim 3, if a surrogate set $S(x)$ contains a clean point in a low-diameter set A, then the total number of semi-saturated points in A is less than $f + 1$. More importantly, assuming that we are considering a cross edge (x, y) in line 19, then after adding $M(S(x), S(y))$ to H, there are still $f + 1$ semi-saturated points in A, since we only change the degree of at most $f + 1$ points in A and prioritize using semi-saturated points over clean ones. Then, we have the following direct corollary of Claim 3:

Corollary 1. *Let i and i' be two levels such that $i' \geq i$, A be a set of points with a diameter at most $8r_i$ and (x, y) be a level-i' cross edge in E^*. If $S(x)$ contains a clean point in A before $M(S(x), S(y))$ is added to H, then the total number of semi-saturated points in A after the adding of $M(S(x), S(y))$ is at most $f + 1$.*

For each point $u \in X$, let $\deg_i(u)$ be the number of level-i edges incident to u in H. We have $\deg_{\leq i}(u) = \sum_{k=0}^{i} \deg_k(u)$ and $\deg_{<i}(u) = \sum_{k=0}^{i-1} \deg_k(u)$. For two integers i_1 and i_2 that $0 \leq i_1 \leq i_2 \leq \zeta$, let $\deg_{[i_1, i_2]}(u)$ be the total number of edges at a level within the range $[i_1, i_2]$. Formally, $\deg_{[i_1, i_2]}(u) = \sum_{k=i_1}^{i_2} \deg_k(u)$.

We now prove that if the maximum degree increase of a set is less than the gap between saturated and clean, there are at most $f + 1$ points in that set becoming semi-saturated. For every $A \subseteq X$, let $\text{maxdeg}_i(A) = \max_{w \in A} \deg_i(w)$ and $\text{maxdeg}_{[i,j]}(A) = \max_{w \in A} \deg_{[i,j]}(w)$

Lemma 12. *Let i be a level and A be a set of i-clean points with a diameter at most $8r_i$. For every k such that $\text{maxdeg}_{[i, i+k]}(A) \leq (c_3 - c_2)f$, the number of $(i + k)$-clean points in A is at least $|A| - (f + 1)$.*

Proof. Let $A = \{a_1, a_2, \ldots a_{|A|}\}$. If $|A| \leq f + 1$, Lemma 12 trivially holds. Assume that $|A| > f + 1$. We prove that at most $f + 1$ points in A are semi-saturated before the execution of level $i + k$ in line 11 of Algorithm 1. For any $j \in [1 \ldots |A|]$, since a_j is i-clean, $\deg_{<i}(a_j) \leq c_2 f$. Then, $\deg_{\leq i+k}(a_j) \leq c_2 f + \max_{w \in A} \deg_{[i, i+k]}(w) \leq c_3 f$ by the assumption of the lemma. Thus, no point in A is saturated before level $i + k$.

We prove by contradiction that A contains at least $|A| - (f + 1)$ clean points before iteration $i + k$. Let (x, y) be the first cross edge such that after adding $M(S(x), S(y))$ to $E(H)$, A contains less than $|A| - (f + 1)$ clean points. Let i_{xy} be the level of (x, y); we have $i \leq i_{xy} \leq i + k$. Thus, no point in A is saturated during the execution of level i_{xy}, implying that A contains only clean and semi-saturated points before and after adding $M(S(x), S(y))$. Since some points in A become semi-saturated after adding $M(S(x), S(y))$, either $S(x)$ or $S(y)$ contains some clean points in A. Without loss of generality, assume that $S(x)$ does. We claim that $S(y)$ does not contain any point

in A. Let u be a point in $S(x) \cap A$. By the construction of $S(x)$ in Algorithm 2, $\delta(x,u) \le 16r_{i_{xy}}$. For every point $v \in A$, we have:

$$\delta(v,y) \ge \delta(x,y) - \delta(x,u) - \delta(u,v) \ge 64r_{i_{xy}} - 16r_{i_{xy}} - \mathrm{diam}(A) \ge 40r_{i_{xy}} \quad ,$$

since $\mathrm{diam}(A) \le 8r_i \le 8r_{i_{xy}}$. Hence, $S(y)$ does not contain any point in A because by Algorithm 2, points in $S(y)$ are selected from $\mathbf{B}(y, 16r_{i_{xy}})$. By Corollary 1, there are at most $f+1$ semi-saturated points in A after adding $M(S(x), S(y))$, a contradiction. $\qquad\square$

By Lemma 2, each point's degree can only increase by ξf after each iteration, which means that after a small number of iterations, a set of clean points has at most $f+1$ points turning into non-clean.

Corollary 2. *Let $x = (u,i)$ be a node in T and $x' = (u', i')$ be a descendant of x with $i' \ge i - 4\log\lambda$. If $\mathbf{B}(u', 4r_{i'})$ contains a set A of i'-clean points, then the number of i-clean points in $\mathbf{B}(u, 4r_i)$ is at least $|A| - (f+1)$.*

Proof. By Lemma 2, $\mathrm{maxdeg}_{[i',i]}(w) \le (i-i')\cdot\xi f \le 4\log\lambda \cdot \xi f < (c_3 - c_2)f$. By Lemma 12, the number of i-clean points in A is at least $|A| - (f+1)$. From Item 1 of Property 2, $A \subseteq \mathbf{B}(u', 4r_{i'}) \subseteq \mathbf{B}(u, 4r_i)$. Therefore, the number of i-clean points in $\mathbf{B}(u, 4r_i)$ is at least $|A| - (f+1)$. $\qquad\square$

The next lemma shows that for each node (u,i), $\mathbf{B}(u, 4r_i)$ contains some clean points of the balls in lower levels close to some descendant (u', i') of (u,i). To find such balls, we monitor the changes in the degree of points near u' and prove that for every degree gained, there are a corresponding number of clean points "contributed" to $\mathbf{B}(u, 4r_i)$. If there is an edge from a point near u' to a leaf of a small node, the leaves of that node are clean and we can reuse them after a constant number of levels. If the edge is between a point near u' to a leaf of a large node, we assume that there are some clean points close to the representative of each large node. This assumption will be our induction hypothesis in the proof of Lemma 11.

Lemma 13. *Let (u,i) be a node at level $i > \lceil 2\log\lambda\rceil + 1$ and (u', i') be a descendant of (u,i) at level $i' = i - (\lceil 2\log\lambda\rceil + 1)$. Assume that for every large node (u'', i') at level i', the number of i'-clean points in $\mathbf{B}(u'', 4r_{i'})$ is at least $4f+4$. Then, there exists a subset A of $\mathbf{B}(u, 4r_i)$ containing only i-clean points such that $|A| \ge \lceil \frac{\mathrm{maxdeg}_{i'}\mathbf{B}(u', 4r_{i'})}{\xi} \rceil$ and $A \cap \mathbf{B}(u', 4r_{i'}) = \varnothing$.*

Proof. Let z be a point in $\mathbf{B}(u', 4r_{i'})$ such that $\deg_{i'}(z) = \mathrm{maxdeg}_{i'}(\mathbf{B}(u', r_{i'}))$. Let W be the set of points connected to z by level-i' edges. Thus, $|W| = \deg_{i'}(z)$. We consider two cases:

Case 1: If $\deg_{<i'}(w) \le c_1 f$ for every $w \in W$, then by Lemma 2, $\deg_{<i'}(w) \le c_1 f + (\lceil 2\log\lambda\rceil + 1)\cdot\xi f \le c_2 f$. Hence, for all $w \in W$, w is $(i' + \lceil 2\log\lambda\rceil + 1)$-clean (or i-clean). By setting $A = W$, we claim that A satisfies all properties in Lemma 13. First, W is a set of i-clean points contained in $\mathbf{B}(u_i, 4r_i)$ and $|W| = \deg_{i'}(z) = \mathrm{maxdeg}_{i'}(\mathbf{B}(u, r_{i'})) \ge \lceil \frac{\mathrm{maxdeg}_{i'}\mathbf{B}(u', 4r_{i'})}{\xi} \rceil$. Furthermore, for every $w \in W$, $\delta(w, z) \ge 32r_{i'}$ by Item 3 in Property 2. By triangle inequality, $\delta(w, u') \ge \delta(w, z) - \delta(z, u') \ge 32r_{i'} - 4r_{i'} = 28r_{i'}$, implying that $w \notin \mathbf{B}(u', 4r_{i'})$. Thus, $W \cap \mathbf{B}(u', 4r_{i'}) = \varnothing$.

Case 2: There exists a point $w_0 \in W$ such that $\deg_{<i'}(w_0) > c_1 f$. Hence, the ancestor of $(w_0, 0)$ at level i', denoted by (w', i'), is a large node. By the lemma's assumption, the number of i'-clean points in $\mathbf{B}(w', 4r_{i'})$ is at least $4f+4$. Let C be the set of i'-clean points in $\mathbf{B}(w', 4r_{i'})$. By Lemma 2, for every level k, $\mathrm{maxdeg}_k(C) \le \xi f$; hence, $\sum_{k=i'}^{i} \mathrm{maxdeg}_k(C) \le (\lceil 2\log\lambda\rceil + 1)\cdot\xi f \le (c_3 - c_2)f$ by

24

the choice of c_2 and c_3 in Algorithm 1. Thus, by Lemma 12, the number of i-clean points in C is at least $|C| - (f+1) \geq 3(f+1) \geq \lceil \frac{\text{maxdeg}_{i'}\mathbf{B}(u',4r_{i'})}{\xi} \rceil$ since $\text{maxdeg}_{i'}\mathbf{B}(u',4r_{i'}) \leq \xi f$ by Lemma 2.

Let A be the set of i-clean points in C satisfies conditions in Lemma 13. We show that A satisfies the conditions in Lemma 13. Since $A \subseteq C \subseteq \mathbf{B}(u',4r_{i'})$, $\text{diam}(A) \leq \text{diam}(C) \leq 8r_{i'}$. For every point $v \in C$, by the triangle inequality,

$$
\begin{aligned}
\delta(u,v) &\leq \delta(u,u') + \delta(u',v) \leq 2r_i + \delta(u',v) \qquad \text{(by Claim 1)}\\
&\leq 2r_i + \underbrace{\delta(u',z)}_{\leq 4r_{i'}} + \delta(z,w_0) + \underbrace{\delta(w_0,w')}_{\leq 2r_{i'}} + \underbrace{\delta(w',v)}_{4r_{i'}} \leq 2r_i + 10r_{i'} + \delta(z,w_0)\\
&\leq 2r_i + 10r_{i'} + (\lambda + 32)\underbrace{r_{i'}}_{\leq r_i/\lambda^2} \qquad \text{(since (z,w_0) is a level-i edge)} \\
&\leq 2r_i + \frac{\lambda + 42}{\lambda^2}r_i \leq 4r_i.
\end{aligned}
$$

(17)

Therefore, $C \subseteq \mathbf{B}(u,4r_i)$, implying that $A \subseteq \mathbf{B}(u,4r_i)$. Furthermore, for each $v \in C$, we have:

$$
\begin{aligned}
\delta(v,u') &\geq \delta(w_0,z) - \delta(v,w_0) - \delta(u',z) \qquad \text{(by the triangle inequality)}\\
&\geq 32r_{i'} - \delta(v,w_0) - \underbrace{\delta(u',z)}_{\leq 4r_{i'}} \qquad \text{(since $\delta(z,w_0) \geq 32r_{i'}$)}\\
&\geq 32r_{i'} - \underbrace{\delta(v,w')}_{\leq 4r_{i'}} - \underbrace{\delta(w',w_0)}_{\leq 2r_{i'}} - 4r_{i'} \geq 32r_{i'} - 4r_{i'} - 2r_{i'} - 4r_{i'} \geq 22r_{i'} > 4r_{i'}.
\end{aligned}
$$

(18)

This implies that $v \notin \mathbf{B}(u',4r_{i'})$ for every $v \in C$. Since $A \subseteq C$, it follows that $A \cap \mathbf{B}(u',4r_{i'}) = \varnothing$. □

We now return to the proof of Lemma 11. The idea is to track the degree change of points in some path P of T from (u,i) to one of its leaves. If there is a point whose degree changed, then there is at least one cross edge between (a node in) P to another node, say (v,i'). The clean points in $\mathbf{B}(v,4r_{i'})$ become "closer" to the P at higher levels relative to the level radius and eventually can be used as surrogates by nodes in P.

Proof of Lemma 11. We induct on the level i. When $i = 0$, Lemma 11 holds trivially since there is no large node at level 0. Assume that Lemma 11 holds for all large nodes at levels lower than i. Since x is large, there exists a leaf v of x whose $\deg_{<i}(v) \geq c_1 f$. Let $i' = i - 2\log\lambda$ and x' be the ancestor of $(v,0)$ at level i'. By Lemma 2, $\deg_{<i'}(v) \geq c_1 f - 2\log\lambda \cdot \xi f$. We show that the number of i'-clean points in $\mathbf{B}(u',4r_{i'})$ is at least $6f + 6$.

For $k \in [0\ldots i']$, let (v_k,k) be the ancestor of $(v,0)$ at level k. Let $l = \lceil 2\log\lambda \rceil$ and i'' be the highest level such that $\sum_{k=i''}^{i'} \text{maxdeg}_k(\mathbf{B}(v_k,4r_k)) \geq (l+1) \cdot 2\xi \cdot (6f+6)$. The level i'' exists since $v \in \mathbf{B}(v_k,4r_k)$ for every k by Claim 1 and

$$\deg_{<i'}(v) \geq c_1 f - 2\log\lambda \cdot \xi f \geq 48\log\lambda \cdot \xi \cdot f \geq (l+1) \cdot 2\xi \cdot (6f+6) \qquad \text{(by the choice of c_1).}$$

We partition the set $I = \{i'', i''+1, \ldots i'\}$ into congruent classes $I_0, I_1, I_2, \ldots I_l$ of modulo $l+1$. Formally, $I_\alpha = \{k \in I : k - \alpha \equiv 0 \mod l+1\}$ for each $\alpha \in [0\ldots l]$. By the pigeonhole principle, there exists an integer $\beta \in [0\ldots l]$ such that:

$$\sum_{k \in I_\beta} \text{maxdeg}_k(\mathbf{B}(v_k,4r_k)) \geq 2\xi \cdot (6f+6)$$

(19)

25

For simplicity, assume that $i' \equiv i'' \equiv \beta \mod l + 1$. Let $I_\beta = \{i'' = i_0, i_1, i_2, \ldots i_s = i'\}$ with $i_j = i'' + j \cdot (l+1)$ and $s = \frac{i'-i''}{l+1}$.

By Lemma 13, for each $k \in [0, 1, \ldots s-1]$, there exists a set A_{i_k} of i_{k+1}-clean points in $\mathbf{B}(v_{i_{k+1}}, r_{i_{k+1}})$ such that $|A_{i_k}| \geq \lceil \frac{\text{maxdeg}_{i_k}(\mathbf{B}(v_{i_k}, 4r_{i_k}))}{\xi} \rceil$. For each $h \leq k$, A_{i_h} is a subset of $\mathbf{B}(v_{i_{h+1}}, 4r_{i_{h+1}})$ and hence is a subset of $\mathbf{B}(v_{i_{k+1}}, 4r_{i_{k+1}})$. Since $A_{i_{k+1}} \cap \mathbf{B}(v_{i_{k+1}}, 4r_{i_{k+1}}) = \varnothing$, $A_{i_h} \cap A_{i_{k+1}} = \varnothing$ for every $h \leq k$, implying that $A_{i_1}, A_{i_2}, \ldots A_{i_{s-1}}$ are pairwise-disjoint. See Figure 2 for an illustration of the inclusion relation between A_{i_1}, A_{i_2} and A_{i_3} with $\mathbf{B}(v_{i_2}, 4r_{i_2}), \mathbf{B}(v_{i_3}, 4r_{i_3})$ and $\mathbf{B}(v_{i_4}, 4r_{i_4})$. One can see from Figure 2 that, if we keep going up the path, we gain more clean points.
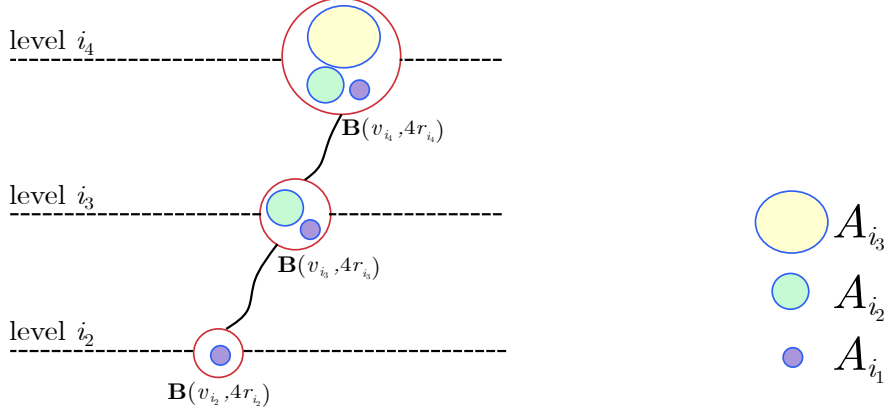

Figure 2: An example of A_{i_1}, A_{i_2} and A_{i_3}. Here, A_{i_1}, A_{i_2} and A_{i_3} are disjoint. For each $k' \in \{1, 2, 3\}$, $A_{i_{k'}} \subseteq \mathbf{B}(v_{i_k}, 4r_{i_k})$ for all $k > k'$.

We then bound the total number of points in A_{i_k} for all $k \in [1, s-1]$:

$$
\begin{aligned}
\left| \bigcup_{k=1}^{s-1} A_{i_k} \right| = \sum_{t=1}^{s-1} |A_{i_k}| &\geq \sum_{k=1}^{s-1} \lceil \frac{\text{maxdeg}_{i_k} \mathbf{B}(v_{i_k}, 4r_{i_k})}{\xi} \rceil \geq \sum_{k=1}^{s-1} \frac{\text{maxdeg}_{i_k} \mathbf{B}(v_{i_k}, 4r_{i_k})}{\xi} \\
&= \left(\sum_{k=1}^{s} \frac{\text{maxdeg}_{i_k} \mathbf{B}(v_{i_k}, 4r_{i_k})}{\xi} \right) - \frac{\text{maxdeg}_{i_s} \mathbf{B}(v_{i_s}, 4r_{i_s})}{\xi} \geq \frac{2\xi(6f+6) - \xi f}{\xi} \geq 6f + 6.
\end{aligned}
\tag{20}
$$

In the next part of the proof, we show that only $f + 1$ of those clean points gained from lower levels turned into semi-saturated.

Let $A = \bigcup_{k=1}^{s-1} A_{i_k}$. We prove by contradiction that at most $f + 1$ points in A are not i_s-clean, which will give us the lemma. We have two observations. First, if a point is k-clean for some k, it is also k'-clean for every $k' \leq k$. Similarly, if a point is k'-saturated, it is also k-saturated for every $k \geq k'$. Hence, every point in A is i_0-clean. Second, every point in A is not saturated before iteration i_s.

Claim 4. *There is no point in A becoming saturated before iteration i_s.*

Proof. Let w be an arbitrary point in A and $k \in [0, s-1]$ be the index that $w \in A_{i_k}$. Since w is i_{k+1}-clean, $\deg_{<i_{k+1}}(w) \leq c_2 f$. On the other hand, $w \in \mathbf{B}(v_{i_{k+1}}, 4r_{i_{k+1}})$. Recall that for every $i_0 \leq h' \leq h \leq i_s$, (v_h, h) is the ancestor of $(v_{h'}, h')$ and hence $\mathbf{B}(v_{h'}, 4r_{h'}) \subseteq \mathbf{B}(v_h, 4r_h)$. Thus,

for every $h \in [i_{k+1}, i_s]$, w is also in $\mathbf{B}(v_h, 4r_h)$, implying that $\deg_h(w) \le \operatorname{maxdeg}_h(\mathbf{B}(v_h, 4r_h))$. Therefore,

$$\deg_{<i_s}(w) = \deg_{<i_{k+1}}(w) + \sum_{h=i_{k+1}}^{i_s-1} \deg_h(w) \le c_2 f + \sum_{h=i_{k+1}}^{i_s-1} \operatorname{maxdeg}_h(\mathbf{B}(v_h, 4r_h))$$

$$\le c_2 f + \sum_{h=i''}^{i'} \operatorname{maxdeg}_h(\mathbf{B}(v_h, 4r_h)) \qquad (\text{since } i'' = i_0 \le i_{k+1} \le i_s = i') \tag{21}$$

Recall that i'' is the highest index such that $\sum_{k=i''}^{i'} \operatorname{maxdeg}_k(\mathbf{B}(v_k, 4r_k)) \ge (l+1) \cdot 2\xi \cdot (6f+6)$. By the maximality of i'', $\sum_{k=i''+1}^{i'} \operatorname{maxdeg}_k(\mathbf{B}(v_k, 4r_k)) < (l+1) \cdot 2\xi \cdot (6f+6)$, implying that:

$$\sum_{k=i''}^{i'} \operatorname{maxdeg}_k(\mathbf{B}(v_k, 4r_k)) \le \sum_{k=i''+1}^{i'} \operatorname{maxdeg}_k(\mathbf{B}(v_k, 4r_k)) + \operatorname{maxdeg}_{i''}(\mathbf{B}(v_{i''}, 4r_{i''})) \tag{22}$$

$$\le (l+1) \cdot 2\xi \cdot (6f+6) + \xi f \le (c_3 - c_2)f.$$

The last equation holds by the choice of c_2 and c_3. By Equation (21) and Equation (22), we get $\deg_{<i_s}(w) \le c_3 f$ and hence, w is non-saturated before iteration i_s. □

Assume that A contains more than $f+1$ non-clean points before iteration i_s. Let (x, y) be the first cross edge in E^* such that after $M(S(x), S(y))$ is added to H, A has more than $f+1$ non-clean points. Let j be the level of (x, y) and t be index such that $i_t \le j < i_{t+1}$ $(0 \le t < s)$. Let $\tau(x, y)$ be the time when $M(S(x), S(y))$ is added to H. Since all points in A are not i_s-saturated by Claim 4, A contains only clean and semi-saturated points before $\tau(x, y)$. Furthermore, the number of semi-saturated points in A is at most $f+1$ before $\tau(x, y)$. Let $A_{<i_t} = \bigcup_{k<t} A_{i_k}$ and $A_{\ge i_t} = A \setminus A_{<i_t} = \bigcup_{k \ge t} A_{i_k}$. Since all points in $A_{\ge i_t}$ are i_{t+1}-clean, they are also clean after $\tau(x, y)$. Then, all semi-saturated points in A before and after $\tau(x, y)$ are in $A_{<i_t}$. Recall that $A_{<i_t} \subseteq \mathbf{B}(u_{i_t}, 4r_{i_t})$; hence, $\operatorname{diam}(A_{<i_t}) \le 8r_{i_t} \le 8r_j$ as $i_t \le j$. Since the number of clean points in $A_{<i_t}$ before $\tau(x, y)$ is smaller than that after $\tau(x, y)$, there must be a clean point in $A_{<i_t}$ becoming semi-saturated. Thus, $S(x)$ or $S(y)$ must contain a clean point in $A_{<i_t}$. However, by Corollary 1, there are at most $f+1$ semi-saturated points in $A_{<i_t}$ after $\tau(x, y)$, contradicted to the assumption that A contains more than $f+1$ non-clean points after $\tau(x, y)$.

Therefore, the number of i_s-clean points in A is at least $(6f+6) - (f+1) = 5f+5$. Since $A \subseteq \mathbf{B}(u_{i_s}, 4r_{i_s}) = \mathbf{B}(u_{i'}, 4r_{i'})$ (recall $i_s = i'$), $\mathbf{B}(u_{i'}, 4r_{i'})$ contains at least $5f+5$ clean points at level i'. Using Corollary 2, we obtain that the number of i-clean points in $\mathbf{B}(u, 4r_i)$ is at least $(5f+5) - (f+1) = 4f+4$.

□

6 Fault-tolerance

In this section, we prove the fault tolerance property, meaning that after removing any f points from H, the remaining graph is still a spanner. Throughout this section, we assume that $\epsilon \le \frac{1}{20}$.

Lemma 14. *Algorithm 1 produces a f-VFT $(1+5\epsilon)$-spanner H of X, i.e., for any set F of at most f points in X, $H[X-F]$ is a $(1+5\epsilon)$-spanner of $(X-F, \delta)$.*

Equivalently, we need to show that for every pair of points (u,v) and every set $F \subseteq X \setminus \{u,v\}$ of at most f points, there is a path from u to v in $H - F$ whose length is at most $(1+\epsilon)\delta(u,v)$. We prove by induction on the length $\delta(u,v)$. To find a short path from u to v in $H - F$, we follow the shortest path P from u to v in G. Intuitively, we find a cross edge, denoted by (x,y), in E^* from an ancestor x of (the leaf corresponding to) u to the ancestor y of some point u' between u and v in P. Hence, we create a path in $H - F$ from u to a surrogate of x to a surrogate of y, and recursively do the same for the path from u' to v. For this method to work, we need (x,y) to have two properties:

- $\delta(x,y)$ is a good approximation of (u,u'). This is the case when (x,y) is a low ancestor of the original cross edge of (u,u') (the formal definition of an ancestor of a cross edge will be given later).

- Each surrogate set of x and y has $f + 1$ points; otherwise, if one surrogate set, say $S(x)$, has less than $f + 1$ points, then there is no non-faulty edges in the bipartite connection $M(S(x), S(y))$ in case $F = S(x)$ (a faulty edge is an edge with at least one end in F). By Lemma 7, a complete node always has $f + 1$ points in its surrogate. Hence, we find x among the complete ancestors of $(u,0)$.

For each incomplete node x in T, the *lowest complete ancestor* (LCA) of x, denoted by $\mathrm{LCA}(x)$, is the parent of the almost complete ancestor of x. Recall that an almost complete node is an incomplete node whose parent is complete. For each point $u \in X$, let $\mathrm{LCA}(u) = \mathrm{LCA}(u,0)$. For each cross edge (x,y), a cross edge $(\tilde{x}, \tilde{y})$ is an ancestor (parent) of (x,y) if $\tilde{x}$ and $\tilde{y}$ are ancestors (parents) of x and y, respectively. For each pair $(u,v) \in X^2$, assume that $(\hat{u}, \hat{v})$ is the original cross edge of (u,v). The κ-*cross edge* of (u,v) is the ancestor of $(\hat{u}, \hat{v})$ at level $\mathrm{lvl}(\hat{u}) + \kappa$. If $\kappa \leq 10$, we call (x,y) a *good cross edge* of (u,v).

Recall that an original cross edge of (u,v) is the lowest-level cross edge (x,y) such that x and y are ancestor of $(u,0)$ and $(v,0)$. For each level-i node x and a level j, consider two cases:

- If $j \geq i$, $\mathrm{Aug}_j(x)$ is $\mathrm{Cross}(NC[x'])$, with x' is the ancestor of x at level j.

- If $j < i$, $\mathrm{Aug}_j(x)$ is the union among all descendants x'' of x at level j of $\mathrm{Cross}(NC[x''])$.

Recall that $\mathrm{Cross}(A)$ is the set of cross edges between nodes in A for every $X \subseteq V(T)$.

Property 3. *We have the following properties:*

1. *Let (u,v) be any pair of points in X, i be the level of the original cross edge of (u,v) and κ be any integer in $[0, \zeta - i]$. For every κ-cross edge $(\tilde{u}, \tilde{v})$ at level $j = i + \kappa$, $\delta(\tilde{u}, \tilde{v}) \geq r_j \cdot \left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)$. Furthermore, if $\kappa \leq 10$, $\delta(\tilde{u}, \tilde{v}) \geq 64 r_j$.*

2. *Let (u,v) be any pair of points in X, $(\tilde{u}, \tilde{v})$ be any good cross edge of (u,v) and j be the level of $(\tilde{u}, \tilde{v})$. For every two points $u' \in \mathbf{B}(\tilde{u}, 16 r_j)$ and $v' \in \mathbf{B}(\tilde{v}, 16 r_j)$, $1 - 5^{-3}\epsilon \leq \frac{\delta(u',v')}{\delta(u,v)} \leq 1 + 5^{-3}\epsilon$.*

3. *Let x be any node in T, i be a level in T such that $\mathrm{lvl}(x) \leq i \leq \zeta$ and (y,z) be any level-i cross edge in $\mathrm{Aug}_i(x)$. Then, $\delta(x,y) \leq (\lambda + 2) r_i$.*

4. *For any original cross edge $(\hat{u}, \hat{v})$, $\log \frac{\delta(\hat{u},\hat{v})}{\lambda} \leq \mathrm{lvl}(\hat{u}) < \log \frac{\delta(\hat{u},\hat{v})}{\lambda} + 2$.*

5. Let (x,y) be a cross edge at level i ($\delta(x,y) \le \lambda r_i$). For any ancestor (x',y') of (x,y) at level i', (x',y') is also a cross edge, i.e., $\delta(x',y') \le \lambda r_{i'}$, which implies that $x' \in NC[y']$ and $y' \in NC[x']$.

Proof. Item 1: Let $(\hat{u},\hat{v})$ be the original cross edge of (u,v). By Item 3 of Property 1, $\delta(\hat{u},\hat{v}) \ge \lambda r_i/6$. By Claim 1, $\delta(\hat{u},\tilde{u}), \delta(\hat{v},\tilde{v}) \le 2r_j$. By the triangle inequality,

$$\delta(\tilde{u},\tilde{v}) \ge \delta(\hat{u},\hat{v}) - \delta(\tilde{u},\hat{u}) - \delta(\tilde{v},\hat{v}) \ge \lambda r_i/6 - 4r_j = r_j\left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right) \quad , \tag{23}$$

as claimed. When $\kappa \le 10$, $\delta(\tilde{u},\tilde{v}) \ge r_j \cdot \left(\frac{\lambda}{6 \cdot 5^{10}} - 4\right) = r_j \cdot (5^8(1+\epsilon^{-1}) - 4) \ge 64 r_j$ as $\lambda = 5^{20}(1+\epsilon^{-1})$.

Item 2: Let $(\hat{u},\hat{v})$ be the original cross edge of (u,v) and $i = \text{lvl}(\hat{u})$. By Claim 1, $\delta(\tilde{u},\hat{u}), \delta(\tilde{v},\hat{v}) \le 2r_j$. Using the triangle inequality, we have:

$$\begin{aligned}
\delta(u',v') &\le \delta(\hat{u},\hat{v}) + \delta(u',\hat{u}) + \delta(v',\hat{v}) \\
&\le \delta(\hat{u},\hat{v}) + \underbrace{\delta(u',\tilde{u})}_{\le 16r_j} + \underbrace{\delta(v',\tilde{v})}_{\le 16r_j} + \delta(\tilde{u},\hat{u}) + \delta(\tilde{v},\hat{v}) \\
&\le \delta(\hat{u},\hat{v}) + 16r_j + 16r_j + 2r_j + 2r_j \le \delta(\hat{u},\hat{v}) + 36r_j \le \delta(\hat{u},\hat{v}) + 5^{13} r_i.
\end{aligned} \tag{24}$$

Similarly, $\delta(u',v') \ge \delta(\hat{u},\hat{v}) - 5^{13} r_i$.

$$\begin{aligned}
\frac{\delta(u',v')}{\delta(u,v)} &\le \frac{\delta(\hat{u},\hat{v}) + 5^{13} r_i}{\delta(\hat{u},\hat{v}) - 4r_i} = 1 + \frac{5^{13} r_i + 4r_i}{\delta(\hat{u},\hat{v}) - 4r_i} = 1 + \frac{5^{13} + 4}{\delta(\hat{u},\hat{v})/r_i - 4} \\
&\le 1 + \frac{5^{13} + 4}{\lambda/6 - 4} \qquad \text{(by Item 3 of Property 1)} \\
&= 1 + \frac{5^{13} + 4}{5^{19}(1 + 1/\epsilon) - 6} \le 1 + 5^{-3}\epsilon \quad ,
\end{aligned} \tag{25}$$

Using similar argument, $\frac{\delta(u',v')}{\delta(u,v)} \ge 1 - 5^{-3}\epsilon$.

Item 3: Let x_i be the ancestor of x at level i. Since $(y,z) \in \text{Aug}_i(x)$, y and z are both in $NC[x_i]$. Hence, $\delta(x_i,y) \le \lambda r_i$. By the triangle inequality, we have $\delta(x,y) \le \delta(x,x_i) + \delta(x_i,y) \le 2r_i + \lambda r_i \le (\lambda + 2)r_i$ since $\delta(x,x_i) \le 2r_i$ by Claim 1.

Item 4: Let $i = \text{lvl}(\hat{u})$. By the definition of cross edges, $\delta(\hat{u},\hat{v}) \le \lambda r_i$. By Item 3 of Property 1, $\delta(\hat{u},\hat{v}) \ge \lambda r_i/6$. Taking the logarithm of both sides, we obtain the desired inequality.

Item 5: We only need to show that Item 5 is true when (x',y') is the parent of (x,y) ($i' = i+1$), the result for any ancestor of (x,y) follows by induction. By Claim 1, $\delta(x',x), \delta(y',y) \le 2r_{i'}$. Using the triangle inequality,

$$\begin{aligned}
\delta(x',y') &\le \delta(x',x) + \delta(x,y) + \delta(y,y') \le 2r_{i'} + \lambda r_i + 2r_{i'} \\
&\le (\lambda/5 + 4)r_{i'} \le \lambda r_{i'} \qquad \text{(since } r_{i'} = 5r_i \text{ and } \lambda > 5),
\end{aligned} \tag{26}$$

as claimed. $\qquad\square$

By Item 1, a good cross edge (x,y) at level i is always longer than $64r_i$. Hence, when Algorithm 1 discovers a good cross edge (x,y) either in line 6 or line 16, (x,y) is always added to E^*.

Given a path $P = (u_1, u_2, \ldots u_l)$ in G, a cross edge (x,y) is a *P-detour* if:

- (x, y) is a good cross edge of (u_1, u_l) or

- (x, y) is a good cross edge of (u_1, u_k) for some integer $k \in (1, l)$ and both x and y are complete.

We consider some properties of a P-detour:

Property 4. *Let u and v be two points in X and P be the shortest path from u to v in G. Let (x, y) be a P-detour in E^* and $i = \text{lvl}(x)$ $(= \text{lvl}(y))$. We have the following properties:*

1. There exists a non-faulty edge (w, z) in H such that $w \in \mathbf{B}(x, 16r_i)$ and $z \in \mathbf{B}(y, 16r_i)$.

2. $r_i \leq 5^{-8}\epsilon \cdot \min\{\delta(u, v), \delta(x, y)\}$.

Proof. Item 1: Consider the time we add $M(S(x), S(y))$ to H in line 22. We show that there is at least one remaining edge in $M(S(x), S(y))$ after f points (not including u or v) is deleted.

If x and y are complete, $S(x)$ and $S(y)$ contains $f + 1$ points each, and by Definition 3, $M(S(x), S(y))$ is a matching of size $f + 1$. Hence, when we delete f points, there is still at least one remaining edge in $M(S(x), S(y))$. By Algorithm 2, $S(x) \subseteq \mathbf{B}(x, 16r_i)$ and $S(y) \subseteq \mathbf{B}(y, 16r_i)$, which completes our proof.

If either $S(x)$ or $S(y)$ is incomplete, then x and y are ancestors of $(u, 0)$ and $(v, 0)$ by the definition of a P-detour. By Definition 3, $M(S(x), S(y))$ is the complete bipartite graph between $S(x)$ and $S(y)$. Hence, we only need to show that $S(x) \setminus F$ and $S(y) \setminus F$ are non-empty. This is true if $|S(x)| = f + 1$ or $|S(y)| = f + 1$. If $|S(x)| \leq f$, since the surrogate set of an incomplete node must contain all of its leaves, $S(x)$ must contain u. Then, $S(x) \setminus F$ is non-empty since $u \notin F$. Similarly, $S(y) \setminus F$ is non-empty.

Item 2: Since (x, y) is a good cross edge of (u, u') for some $u' \in P$, (x, y) is a κ-cross edge of (u, u') with $\kappa \leq 10$. From Item 1, we have $\delta(x, y) \geq r_i\left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)$. Hence,

$$
\begin{aligned}
r_i &\leq \delta(x, y) \cdot \left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \leq \frac{\delta(u, u')}{1 + 5^{-3}\epsilon} \cdot \left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \quad \text{(by It. 2 of Prop. 3)} \\
&\leq \frac{\mathbf{dist}_G(u, u')}{1 + 5^{-3}\epsilon} \cdot \left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \leq \frac{\mathbf{dist}_G(u, v)}{1 + 5^{-3}\epsilon} \cdot \left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \quad \text{(since } P \text{ contains } u') \\
&\leq \frac{(1 + \epsilon)\delta(u, v)}{1 + 5^{-3}\epsilon}\left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \leq \frac{(1 + \epsilon)\delta(u, v)}{1 + 5^{-3}\epsilon}\left(\frac{\lambda}{6 \cdot 5^{10}} - 4\right)^{-1} \quad \text{(since } \kappa \leq 10) \\
&= \frac{1 + \epsilon}{1 + 5^{-3}\epsilon} \cdot \frac{6 \cdot 5^{10}}{5^{20}(1 + \epsilon^{-1}) - 24 \cdot 5^{10}}\delta(u, v) \leq 5^{-8}\epsilon\delta(u, v)
\end{aligned}
\tag{27}
$$

The first equation of Equation (27) also gives us:

$$
\begin{aligned}
r_i &\leq \delta(x, y)\left(\frac{\lambda}{6 \cdot 5^\kappa} - 4\right)^{-1} \leq \delta(x, y) \cdot \left(\frac{\lambda}{6 \cdot 5^{10}} - 4\right)^{-1} \quad \text{(since } \kappa \leq 10) \\
&= \frac{6 \cdot 5^{10}}{5^{20}(1 + \epsilon^{-1}) - 24 \cdot 5^{10}}\delta(x, y) \leq 5^{-8}\epsilon\delta(x, y),
\end{aligned}
\tag{28}
$$

as claimed. $\qquad \square$

We have the following lemma:

Lemma 15. *For every pair of points (u, v) in X, let P be any shortest path from u to v in G. Then, there exists a P-detour in E^*.*

We now show that Lemma 14 follows from Lemma 15.

Proof of Lemma 14. Let u and v be two points of $X \setminus F$ and $P = (u = u_1, u_2, \ldots u_l = v)$ be a shortest path from u to v in G. We induct on the length $\delta(u, v)$. If $\delta(u, v) \le \lambda$ (the minimum distance between two points is 64), we claim that the cross edge between $(u, 0)$ and $(v, 0)$ is in E^*. Observe that both $(u, 0)$ and $(v, 0)$ are incomplete since each of them has one leaf with degree 0 before iteration 0. Hence, by line 12–16, the cross edge between $(u, 0)$ and $(v, 0)$ is in E^* since $\delta(u, v) \ge 64 \ge 64 r_0$. By line 22, $M(S(u, 0), S(v, 0))$ is added to H. By line 2 of Algorithm 2 $S(u, 0) = \{u\}$ and $S(v, 0) = \{v\}$. Thus, $(u, v) \in E(H)$.

Assume that for any two points in $X \setminus F$ with distance less than $\delta(u, v)$, there is an $(1 + 5\epsilon)$-spanner path between them in $H - F$. By Lemma 15, there exist a P-detour (x_1, y_1). Let $i_1 = \text{lvl}(x_1)$. We consider two cases:

Case 1: (x_1, y_1) is a good cross edge of (u, v). We claim that there is a path from u to v with total weight less than $(1 + 5\epsilon)\delta(u, v)$. Let u', v' be two points in $\mathbf{B}(x_1, 16 r_{i_1})$ and $\mathbf{B}(y_1, 16 r_{i_1})$ such that $(u', v') \in E(H - F)$. u' and v' exist by Item 1 of Property 4.

Since $(u, 0)$ is a leaf of x_1, $\delta(u, x_1) \le 2 r_{i_1}$ by Claim 1. By the triangle inequality,

$$\delta(u', u) \le \delta(u, x_1) + \delta(x_1, u') \le 2 r_{i_1} + 16 r_{i_1} = 18 r_{i_1} \tag{29}$$

By Item 2, $r_{i_1} \le 5^{-8} \epsilon \delta(u, v)$. Plugging in Equation (29), we obtain $\delta(u', u) \le 18 \cdot 5^{-8} \epsilon \delta(u, v) < \delta(u, v)$. By our induction hypothesis, $\mathbf{dist}_{H-F}(u', u) \le (1 + 5\epsilon)\delta(u, u')$. Using the same argument, we get $\delta(v, v') \le 18 r_{i_1}$ and $\mathbf{dist}_{H-F}(v, v') \le (1 + 5\epsilon)\delta(v, v')$. By the triangle inequality,

$$\begin{aligned}
\mathbf{dist}_{H-F}(u, v) &\le \mathbf{dist}_{H-F}(u, u') + \delta(u', v') + \mathbf{dist}_{H-F}(v', v) \\
&\le (1 + 5\epsilon)\underbrace{\delta(u, u')}_{\le 18 r_{i_1}} + \delta(u', v') + (1 + 5\epsilon)\underbrace{\delta(v', v)}_{\le 18 r_{i_1}} \\
&\le (1 + 5\epsilon)36 r_{i_1} + \delta(u', v') \le (1 + 5\epsilon)36 r_{i_1} + (1 + 5^{-3}\epsilon)\delta(u, v) \quad \text{(by It. 2 of Prop. 3)} \\
&\le (1 + 5\epsilon)36 \cdot 5^{-8} \epsilon \delta(u, v) + (1 + 5^{-3}\epsilon)\delta(u, v) \quad \text{(by It. 2 of Prop. 4)} \\
&\le (1 + 2\epsilon)\delta(u, v) \le (1 + 5\epsilon)\delta(u, v) \quad \text{(since } \epsilon \le 1/20),
\end{aligned} \tag{30}$$

as claimed.

Case 2: (x_1, y_1) is a good cross edge of (u_1, u_{a_1}) for some $a_1 \in (1, l)$. For each $j \in [1, l]$, let P_j be the subpath $(u_j, u_{j+1}, \ldots u_l)$. By Lemma 15, there exists a P_{a_1}-detour, denoted by (x_2, y_2), such that (x_2, y_2) is a good cross edge of (u_{a_1}, u_{a_2}) for some $a_2 > a_1$. Recursively, for each $k > 0$, we find the P_{a_k}-detour, denoted by (x_{k+1}, y_{k+1}), such that (x_{k+1}, y_{k+1}) is a good cross edge of $(u_{a_k}, u_{a_{k+1}})$ until $a_{k+1} = l$. Assume that we have $s + 1$ of such detours with $s > 0$. Let $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_s, y_s), (x_{s+1}, y_{s+1})\}$.

Observation 4. *We have the following properties:*

1. *For every $k \in [1, s + 1]$, (x_k, y_k) is a good cross edge of $(u_{a_k}, u_{a_{k+1}})$.*

2. *For every $k \in [1, s]$, x_k and y_k are complete.*

31

3. *For every $k \in [1,s]$, x_{k+1} and y_k are ancestors of $(u_{a_k}, 0)$. Hence, either x_{k+1} or y_k is the ancestor of the other.*

4. $a_1 < a_2 < a_3 < \ldots < a_{s+1} = l$.

For every $k \in [1, s+1]$, let $i_k = \mathrm{lvl}(x_k)$. For $k \in [1,s]$, let w_k and z_k be two points in $\mathbf{B}(x_k, 16 r_{i_k})$ and $\mathbf{B}(y_k, 16 r_{i_k})$, respectively, such that $(w_k, z_k) \in E(H)$. w_k and z_k exist by Item 1 of Property 4. See Figure 3 for an illustration.
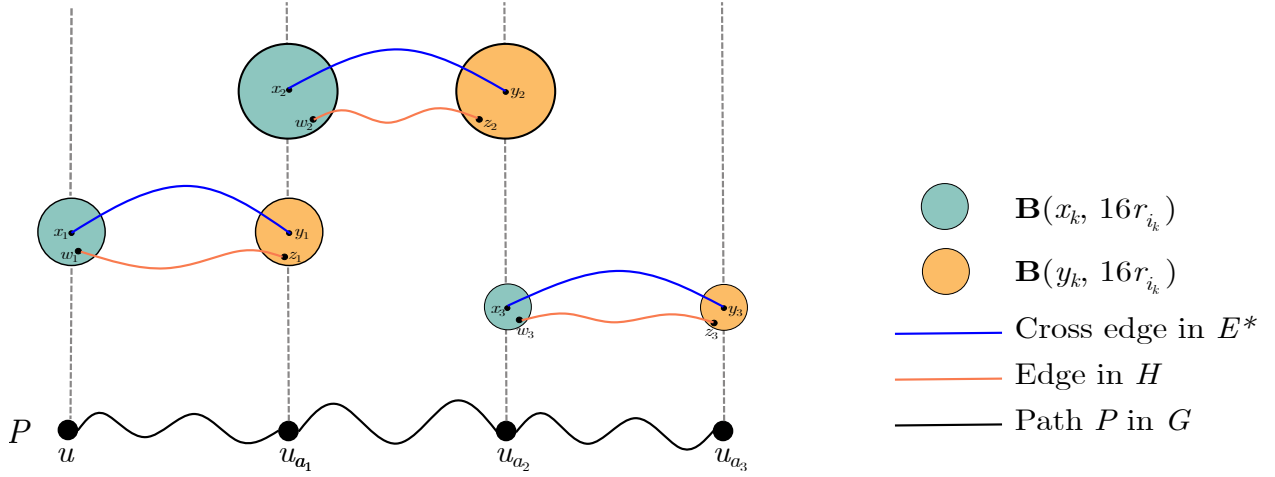

Figure 3: An illustration of $(x_k, y_k), (w_k, z_k)$ for $i \in [1, 3]$. Here, it might be misleading that u_{a_k} is far from x_{k+1} and y_k. In fact, u_{a_k} is in both $\mathbf{B}(x_{k+1}, 16 r_{k+1})$ (the blue ball) and $\mathbf{B}(y_k, 16 r_k)$ (the yellow ball). Furthermore, the higher ball in $\{\mathbf{B}(x_{k+1}, 16 r_{k+1}), \mathbf{B}(y_k, 16 r_k)\}$ contains the other.

Let $a_0 = 1$. Using the triangle inequality, we have:

$$
\begin{aligned}
\mathbf{dist}_{H-F}(u, z_s) &\leq \mathbf{dist}_{H-F}(u, w_1) + \sum_{k=1}^{s} \mathbf{dist}_{H-F}(w_k, z_k) + \sum_{k=1}^{s-1} \mathbf{dist}_{H-F}(z_k, w_{k+1}) \\
&= \mathbf{dist}_{H-F}(u, w_1) + \sum_{k=1}^{s} \delta(w_k, z_k) + \sum_{k=1}^{s-1} \mathbf{dist}_{H-F}(z_k, w_{k+1})
\end{aligned}
\tag{31}
$$

By Item 2 of Property 3, $\delta(w_k, z_k) \leq (1 + 5^{-3}\epsilon)\delta(u_{a_{k-1}}, u_{a_k})$ for every $k \in [1, s]$. Hence,

$$
\begin{aligned}
\sum_{k=1}^{s} \delta(w_k, z_k) &\leq (1 + 5^{-3}) \sum_{k=1}^{s} \delta(u_{a_{k-1}}, u_{a_k}) \leq (1 + 5^{-3}) \sum_{k=1}^{s} \mathbf{dist}_G(u_{a_{k-1}}, u_{a_k}) \\
&\leq (1 + 5^{-3}\epsilon) \mathbf{dist}_G(u, u_{a_s})
\end{aligned}
\tag{32}
$$

The last equation holds since $u_{a_0} = u_1 = u$ and the shortest path from u to u_{a_s} in G, which is P, contains $u_{a_1}, u_{a_2} \ldots u_{a_{s-1}}$.

We claim that $\delta(u, w_1)$ and $\delta(z_k, w_{k+1})$ for $k \in [1, s-1]$ are less than $\delta(u, v)$. Hence, $\mathbf{dist}_{H-F}(u, w_1)$ and $\mathbf{dist}_{H-F}(z_k, w_{k+1})$ are approximate their distances in (X, δ). Since $\delta(u, x_1) \leq 2 r_{i_1}$ by Claim 1 and $w_1 \in \mathbf{B}(x, 16 r_{i_1})$, using the triangle inequality, we have:

$$\delta(u, w_1) \le \delta(u, x_1) + \delta(x_1, w_1) \le 18r_{i_1} < 32r_{i_1}. \tag{33}$$

For every $k \in [1, s-1]$, $z_k \in \mathbf{B}(y_k, 16r_{i_k})$ and $w_{k+1} \in \mathbf{B}(x_{k+1}, 16r_{i_{k+1}})$. By Item 3 of Observation 4, either y_k or x_{k+1} is the ancestor of the other. If y_k is an ancestor of x_{k+1}, then $\mathbf{B}(x_{k+1}, 16r_{i_{k+1}}) \subseteq \mathbf{B}(y_k, 16r_{i_k})$ by Item 1 of Property 2, implying that $w_{k+1} \in \mathbf{B}(y_k, 16r_{i_k})$. Hence, $\delta(z_k, w_{k+1}) \le 16r_{i_k} + 16r_{i_k} = 32r_{i_k}$. Otherwise, x_{k+1} is an ancestor of y_k. Using similar argument, we get $\delta(z_k, w_{k+1}) \le 32r_{i_{k+1}}$. Therefore, we obtain:

$$\delta(z_k, w_{k+1}) \le 32 \max\{r_{i_k}, r_{i_{k+1}}\}. \tag{34}$$

We then prove that $\delta(u, v) > 32r_k$ for every $k \in [1, s]$. Since G is a $(1 + \epsilon)$-spanner of X, for every $k \in [1, s + 1]$, we have:

$$
\begin{aligned}
\delta(u, v) &\ge \frac{\mathbf{dist}_G(u, v)}{1 + \epsilon} \ge \frac{\delta(u_{a_k}, u_{a_{k+1}})}{1 + \epsilon} &&\text{(since } P \text{ contains } (u_{a_k}, u_{a_{k+1}})) \\
&\ge \frac{\delta(x_k, y_k)}{(1 + 5^{-3}\epsilon)(1 + \epsilon)} &&\text{(by It. 2 of Prop. 3)} \\
&\ge \frac{5^8 \epsilon^{-1} r_{i_k}}{(1 + \epsilon)^2}. &&\text{(by It. 2 of Prop. 4),}
\end{aligned}
\tag{35}
$$

implying that $\delta(u, v) > 32r_{i_k}$. By Equation (33), Equation (34), Equation (35), we get $\delta(u, w_1)$ and $\delta(z_k, w_{k+1})$ are less than $\delta(u, v)$. Hence, by the induction hypothesis, $\mathbf{dist}_{H-F}(u, w_1) \le (1 + 5\epsilon)\delta(u, w_1)$ and $\mathbf{dist}_{H-F}(z_k, w_{k+1}) \le (1 + 5\epsilon)\delta(z_k, w_{k+1})$. Then, Equation (33) implies that:

$$\mathbf{dist}_{H-F}(u, w_1) \le (1 + 5\epsilon)\delta(u, w_1) \le (1 + 5\epsilon)32r_{i_1} \le (1 + 5\epsilon)32\sum_{k=1}^{s} r_{i_k}. \tag{36}$$

Similarly, Equation (34) implies that:

$$\sum_{k=1}^{s-1} \mathbf{dist}_{H-F}(z_k, w_{k+1}) \le (1 + 5\epsilon)\sum_{k=1}^{s-1} \delta(z_k, w_{k+1}) \le (1 + 5\epsilon)32\sum_{k=1}^{s-1} \max\{r_{i_k}, r_{i_{k+1}}\} \le (1 + 5\epsilon)64\sum_{i=1}^{k} r_{i_k}. \tag{37}$$

We then bound $\sum_{k=1}^{s} r_{i_k}$. For every index $k \in [1, s]$, by Item 2 of Property 4, $r_{i_k} \le 5^{-8}\epsilon\delta(x_k, y_k)$. Since (x_k, y_k) is a good cross edge of $(u_{a_{k-1}}, u_k)$, $r_{i_k} \le 5^{-8}\epsilon\delta(x_k, y_k) \le 5^{-8}\epsilon(1 + \epsilon)\delta(u_{a_{k-1}}, u_k)$ by Item 1 of Property 3. Hence,

$$\sum_{k=1}^{s} r_{i_k} \le 5^{-8}\epsilon(1 + \epsilon)\sum_{k=1}^{s} \delta(u_{a_{k-1}}, u_{a_k}) \le 5^{-8}\epsilon(1 + \epsilon)\mathbf{dist}_G(u, u_{a_s}) \quad , \tag{38}$$

since the shortest path between u to u_{a_s} in G contains $u_{a_1}, u_{a_2}, \ldots u_{a_{s-1}}$. Plugging Equation (32), Equation (36) and Equation (37) in Equation (31), we get:

$$
\begin{aligned}
\mathbf{dist}_{H-F}(u, z_s) &\le (1 + 5^{-3}\epsilon)\mathbf{dist}_G(u, u_{a_s}) + (1 + 5\epsilon) \cdot 96 \sum_{k=1}^{s} r_{i_k} \\
&\le (1 + 5^{-3}\epsilon)\mathbf{dist}_G(u, u_{a_s}) + \underbrace{(1 + 5\epsilon)96 \cdot 5^{-8}}_{\le 1/2 \ \text{ as } \epsilon \le 1/20} \epsilon(1 + \epsilon)\mathbf{dist}_G(u, u_{a_s}) \text{ (by Eq. 38)} \\
&\le (1 + 3\epsilon/4)\mathbf{dist}_G(u, u_{a_s}) \le (1 + 3\epsilon/4)(1 + \epsilon)\delta(u, u_{a_s}) \text{ (since } G \text{ is a } (1 + \epsilon)\text{-spanner of } X) \\
&\le (1 + 2\epsilon)\delta(u, u_{a_s})
\end{aligned}
\tag{39}
$$

33

The proof of Equation (39) contains all the key ideas of this lemma. The remaining part of the proof focuses on how to deal with a tricky corner case when x_{s+1} is incomplete.

We consider (x_{s+1}, y_{s+1}). If x_{s+1} is complete, let w_{s+1} and z_{s+1} be two points in $\mathbf{B}(x_{s+1}, 4r_{i_{s+1}})$ and $\mathbf{B}(y_{s+1}, r_{i_{s+1}})$ such that $(w_{s+1}, z_{s+1}) \in E(H - F)$. w_{s+1} and z_{s+1} exist by Item 1 of Property 4. Using the same argument in Equation (39), we have $\mathbf{dist}_{H-F}(u, z_{s+1}) \le (1+2\epsilon)\delta(u, v)$. By Claim 1, $\delta(v, y_{s+1}) \le 2r_{i_{s+1}}$. Using the triangle inequality, we get:

$$\begin{aligned}
\delta(v, z_{s+1}) &\le \delta(v, y_{s+1}) + \delta(y_{s+1}, z_{s+1}) \le 2r_{i_{s+1}} + 16r_{i_{s+1}} \\
&= 18r_{i_{s+1}} \le 18(1+\epsilon)^2 5^{-8}\epsilon\delta(u, v) \qquad \text{(by Eq. 35)} \\
&\le \epsilon\delta(u, v)/2 \qquad ,
\end{aligned} \tag{40}$$

implying that $\delta(v, z_{s+1}) < \delta(u, v)$. By our induction hypothesis, $\mathbf{dist}_{H-F}(v, z_{s+1}) \le (1+5\epsilon)\delta(v, z_{s+1}) \le (1+5\epsilon)\epsilon\delta(u, v)/2$. Thus,

$$\begin{aligned}
\mathbf{dist}_{H-F}(u, v) &\le \mathbf{dist}_{H-F}(u, z_{s+1}) + \mathbf{dist}_{H-F}(z_{s+1}, v) \\
&\le (1+2\epsilon)\delta(u, v) + (1+5\epsilon)\epsilon\delta(u, v)/2 \le (1+3\epsilon)\delta(u, v) \qquad ,
\end{aligned} \tag{41}$$

as $\epsilon \le 1/20$.

The last case is when x_{s+1} is incomplete. Let $t = \mathrm{LCA}(u_{a_s})$ and $i = \mathrm{lvl}(z)$ and $\tilde{v}$ be the ancestor of $(v, 0)$ at level i. Since $(t, \tilde{v})$ is an ancestor of a cross edge, $\delta(t, \tilde{v}) \le \lambda r_i$ by Item 5 of Property 3. Consider two cases:
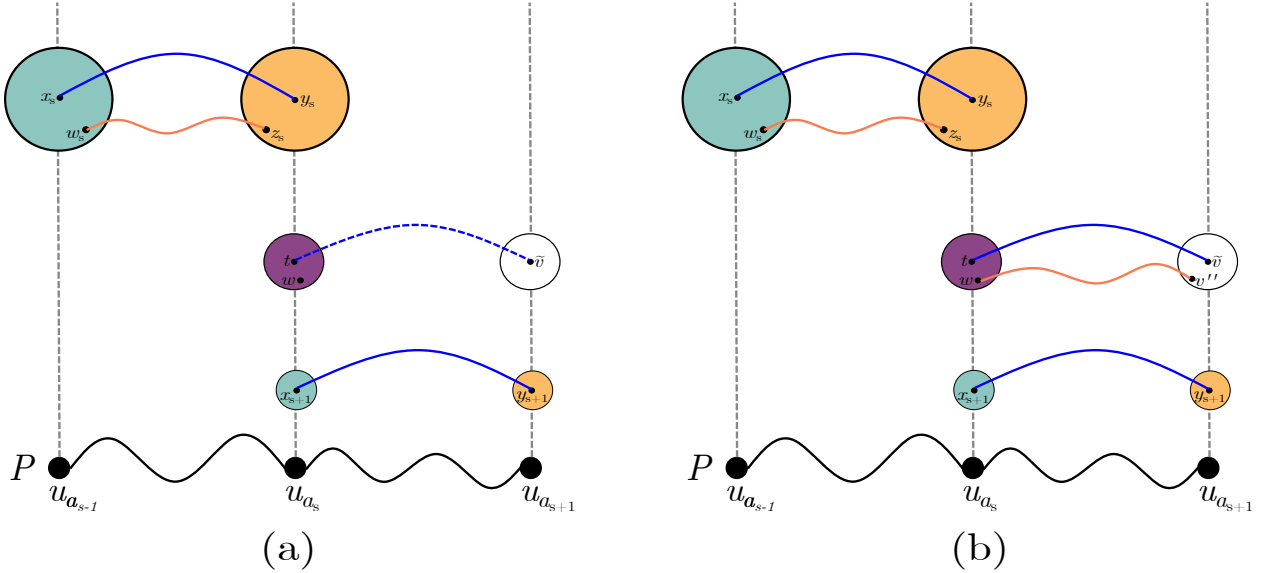

Figure 4: An illustration of two cases. The blue, orange edges and the cyan and yellow balls have the same meaning as those in Figure 3. The purple node is the LCA of u_{a_s} and the white node $\tilde{v}$ is the ancestor of $v = u_{a_{s+1}}$ at the same level. The cross edge between t and $\tilde{v}$ is not in E^* for the first case (a). In figure (b), $(t, \tilde{v}) \in E^*$ and hence there is an edge of $H - F$ in the bipartite connection between $S(t)$ and $S(\tilde{v})$.

Case 1: $\delta(t, \tilde{v}) < 64r_i$. Let w be a point in $\mathbf{B}(t, 16r_i) \smallsetminus F$. w exists since by Lemma 11, $\mathbf{B}(t, 4r_i)$ contains at least $4f + 4$ points. See Figure 4 (a) for an illustration. Since $i \le i_s$, we get y_s is an ancestor of z, implying that $\mathbf{B}(t, 16r_i) \subseteq \mathbf{B}(y_s, 16r_{i_s})$ by Item 1 of Property 2. Hence, $w \in \mathbf{B}(y_s, 16r_{i_s})$. Both z_s and w are in $\mathbf{B}(y_s, 16r_{i_s})$, $\delta(z_s, w) \le 32r_{i_s} < \delta(u, v)$ by Equation (35), which implies:

$$\mathbf{dist}_{H-F}(z_s, w) \le (1 + 5\epsilon)\delta(z_s, w) \le (1 + 5\epsilon)32r_{i_s} \tag{42}$$

by the induction hypothesis. On the other hand,

$$\begin{aligned}
\delta(w, v) &\le \delta(w, t) + \delta(t, \tilde{v}) + \delta(\tilde{v}, v) \\
&\le \delta(w, t) + 64r_i + \delta(\tilde{v}, v) \qquad \text{(by our assumption)} \\
&\le 16r_i + 64r_i + \delta(\tilde{v}, v) \le 80r_i + \delta(\tilde{v}, v) \qquad \text{(since } w \in \mathbf{B}(t, 16r_i)) \\
&\le 96r_i \le 96r_{i_s} \qquad \text{(since } \delta(\tilde{v}, v) \le 2r_i \text{ by Claim 1).}
\end{aligned} \tag{43}$$

Since $r_{i_s} \le 2^{-8}\epsilon(1 + \epsilon)^2\delta(u, v)$ (Equation (35)), we get $\delta(w, v) \le 96 \cdot 2^{-8}\epsilon(1 + \epsilon)^2\delta(u, v) < \delta(u, v)$ as $\epsilon \le 1/20$. By the induction hypothesis,

$$\mathbf{dist}_{H-F}(w, v) \le (1 + 5\epsilon)\delta(w, v) \le (1 + 5\epsilon)96r_{i_s} \tag{44}$$

Hence,

$$\begin{aligned}
\mathbf{dist}_{H-F}(u, v) &\le \mathbf{dist}_{H-F}(u, z_s) + \mathbf{dist}_{H-F}(z_s, w) + \mathbf{dist}_{H-F}(w, v) \\
&\le (1 + 2\epsilon) \underbrace{\delta(u, u_{a_s})}_{\le \mathbf{w}(P) \le (1+\epsilon)\delta(u,v)} + \mathbf{dist}_{H-F}(z_s, w) + \mathbf{dist}_{H-F}(w, v) \qquad \text{(by Eq. 39)} \\
&\le (1 + 2\epsilon)(1 + \epsilon)\delta(u, v) + (1 + 5\epsilon)32r_{i_s} + (1 + 5\epsilon)96r_{i_s} \qquad \text{(by Eq. 42 and Eq. 44)} \\
&\le (1 + 4\epsilon)\delta(u, v) + 128(1 + 5\epsilon)r_{i_s} \\
&\le (1 + 4\epsilon)\delta(u, v) + 128(1 + 5\epsilon) \cdot 5^{-8}\epsilon(1 + \epsilon)^2\delta(u, v) \qquad \text{(by Eq. 35)} \\
&\le (1 + 4\epsilon)\delta(u, v) + \epsilon\delta(u, v) = (1 + 5\epsilon)\delta(u, v) \qquad \text{(since } \epsilon \le 1/20),
\end{aligned} \tag{45}$$

as claimed.

Case 2: $\delta(t, \tilde{v}) \ge 64r_i$. Then, $(t, \tilde{v}) \in E^*$ since $\delta(t, \tilde{v}) \le \lambda r_{i_s}$ and $\mathrm{Aug}_i(t) \subseteq E^*$ as z has at least one incomplete child (line 16). Let w and v'' be two points in $S(t)$ and $S(\tilde{v})$ such that $(w, v'') \in E(H - F)$. w and v'' exist since $S(t)$ contains $f + 1$ points and $S(\tilde{v})$ contains either $f + 1$ points or v. See Figure 4 (b) for an illustration.

Since both w and u_{a_s} are in $\mathbf{B}(t, 16r_i)$ ($\delta(u_{a_s}, t) \le 2r_i$ by Claim 1), we have $\delta(w, u_{a_s}) \le 32r_i \le 32r_{i_s}$. Similarly, $\delta(z_s, w), \delta(v'', v) \le 32r_{i_s}$. Since $32r_{i_s} < \delta(u, v)$ by Equation (35), using the induction hypothesis, we obtain:

$$\mathbf{dist}_{H-F}(z_s, w) \le (1 + 5\epsilon)\delta(z_s, w) \le (1 + 5\epsilon)32r_{i_s} \qquad .$$

Similarly, $\mathbf{dist}_{H-F}(v'', v) \le (1 + 5\epsilon)32r_{i_s}$. Using the triangle inequality, we have:

$$\mathbf{dist}_{H-F}(z_s, v) \le \mathbf{dist}_{H-F}(z_s, w) + \mathbf{dist}_{H-F}(w, v'') + \mathbf{dist}_{H-F}(v'', v)$$

$$\le (1 + 5\epsilon)32r_{i_s} + \delta(w, v'') + (1 + 5\epsilon)32r_{i_s} \qquad \text{(since } (w, v'') \text{ is in } H - F)$$

$$\le (1 + 5\epsilon)64r_{i_s} + \delta(w, u_{a_s}) + \delta(u_{a_s}, v) + \delta(v, v'')$$

$$\le (1 + 5\epsilon)64r_{i_s} + 32r_{i_s} + \delta(u_{a_s}, v) + 32r_{i_s} \qquad \text{(since } w, u_{a_s} \in \mathbf{B}(y_s, 16r_{i_2}) \text{ and } v, v'' \in \mathbf{B}(\tilde{v}, 16r_i))$$

$$\le (1 + 5\epsilon)128r_{i_s} + \delta(u_{a_s}, v).$$

$$(46)$$

Consider a path from u to v passing through z_s, we have the following bound based on the triangle inequality:

$$\mathbf{dist}_{H-F}(u, v) \le \mathbf{dist}_{H-F}(u, z_s) + \mathbf{dist}_{H-F}(z_s, v)$$

$$\le (1 + 2\epsilon)\delta(u, u_{a_s}) + (1 + 5\epsilon)128r_{i_s} + \delta(u_{a_s}, v) \qquad \text{(by Eq. 31 and Eq. 46)}$$

$$\le (1 + 2\epsilon)(\delta(u, u_{a_s}) + \delta(u_{a_s}, v)) + (1 + 5\epsilon)128r_{i_s} \qquad (47)$$

$$\le (1 + 2\epsilon)(1 + \epsilon)\delta(u, v) + (1 + 5\epsilon)128 \cdot 5^{-8}\epsilon(1 + \epsilon)^2\delta(u, v) \qquad \text{(by Eq. 35)}$$

$$\le (1 + 5\epsilon)\delta(u, v) \qquad \text{(since } \epsilon \le 1/20),$$

as desired. $\qquad \square$

In the rest of this section, we focus on proving Lemma 15. We first define some notation. Given a path $P = (u_1, u_2, \ldots u_l)$ in G. A cross edge (x, y) is a *P-jump* if:

1. x is a complete ancestor of $(u_1, 0)$ and y is an ancestor of $(u_j, 0)$ for some $j \in (1, l]$.

2. (x, y) is a κ-cross edge of (u_1, u_j) with $0 \le \kappa \le 2$.

A P-jump is different from a P-detour. In general, a P-detour has two complete end nodes while the definition of P-jump only guarantees one end to be complete. A P-jump (x, y) with incomplete y is a P-detour if and only if y is an ancestor of $(u_l, 0)$.

For every set of cross edges A, let $\mathrm{Pa}(A)$ be the set of all cross edges (x', y') such that there exists a child (x, y) of (x', y') in A. We call $\mathrm{Pa}(A)$ the parent set of A. Given a positive integer i, let $\mathrm{Pa}^1(A) = \mathrm{Pa}(A)$ and $\mathrm{Pa}^i(A) = \mathrm{Pa}(\mathrm{Pa}^{i-1}(A))$. For each node x and a nonnegative integer i, we also use the notation $\mathrm{Pa}^i(x)$ for the ancestor of x at level $\mathrm{lvl}(x) + i$. We prove the following lemma:

Lemma 16. *Let u and v be two points in X, $P = (u = u_1, u_2, \ldots u_l = v)$ be the shortest path between u and v in G, i_1 be the level of $\mathrm{LCA}(u_1)$ and v_1 be the ancestor v at level i_1. If $\delta(\mathrm{LCA}(u_1), v_1) > \lambda r_{i_1}$, then there exists a P-jump (x_0, y_0) in E^*. Furthermore, $\mathrm{Pa}^2(\mathrm{Aug}(x_0, 0, \log \lambda))$ and $\mathrm{Pa}^2(\mathrm{Aug}(y_0, 0, \log \lambda))$ are subsets of E^*.*

To prove Lemma 16, we first show that if a cross edge (y, z) is an augmented cross edge of a node x, then some particular augmmented cross edges of y and z are also augmented cross edges of x.

Claim 5. *Let x be a node in T, γ_1 and γ_2 be two non-negative integers such that $\gamma_1 < \gamma_2 - 2$ and (y, z) is a cross edge in $\mathrm{Aug}(x, 0, \gamma_1)$. Then, $\mathrm{Pa}^2(\mathrm{Aug}(y, 0, \gamma_2 - \gamma_1 - 2))$ and $\mathrm{Pa}^2(\mathrm{Aug}(z, 0, \gamma_2 - \gamma_1 - 2))$ are subsets of $\mathrm{Aug}(x, 0, \gamma_2)$.*

Proof. By symmetric property, we only need to show $\mathrm{Pa}^2(\mathrm{Aug}(y,0,\gamma_2-\gamma_1-2)) \subseteq \mathrm{Aug}(x,0,\gamma_2)$. Let i_x and i_y be the levels of x and y, $i_x \le i_y \le i_x+\gamma_1$. For each cross edge $(z,w) \in \mathrm{Aug}(y,0,\gamma_2-\gamma_1-2)$, let i_{zw} be the level of z and w ($i_y \le i_{zw} \le i_y + \gamma_2 - \gamma_1 - 2$), $z' = \mathrm{Pa}^2(z)$ and $w' = \mathrm{Pa}^2(w)$. We prove that $(z',w') \in \mathrm{Aug}(x,0,\gamma_2)$.

Since (z,w) is a cross edge, its ancestor (z',w') is also a cross edge by Item 5 of Property 3. Observe that $\mathrm{lvl}(z') = i_{zw}+2 \le i_y+\gamma_2-\gamma_1 \le i_x+\gamma_2$. Then, to finish the proof of $(z',w') \in \mathrm{Aug}(x,0,\gamma_2)$, we only need to show that both z' and w' are in the cross neighborhood of the ancestor of x at level $\mathrm{lvl}(z')$.

Let x' and y' be the ancestors of x and y at level $\mathrm{lvl}(z') = i_{zw} + 2$. By Item 3 of Property 3,

$$\delta(x,y) \le (\lambda + 2)r_{i_y} \qquad \text{and} \qquad \delta(y,z) \le (\lambda + 2)r_{i_{zw}} \tag{48}$$

We show that $\delta(x',z') \le \lambda r_{i_{zw}+2}$ and hence $z' \in NC[x']$. By the triangle inequality, we have:

$$
\begin{aligned}
\delta(x',z') &\le \underbrace{\delta(x',x)}_{\le 2r_{\mathrm{lvl}(x')}} + \delta(x,z) + \underbrace{\delta(z,z')}_{\le 2r_{\mathrm{lvl}(z')}} \le 4r_{i_{zw}+2} + \delta(x,z) \qquad (\text{since } \mathrm{lvl}(x') = \mathrm{lvl}(z') = i_{zw} + 2) \\
&\le 4r_{i_{zw}+2} + \delta(x,y) + \delta(y,z) \qquad (\text{by triangle inequality}) \\
&\le 4r_{i_{zw}+2} + (\lambda + 2)r_{i_y} + (\lambda + 2)r_{i_{zw}} \qquad (\text{by Eq. Equation (48)}) \\
&\le \left(4 + \frac{\lambda + 2}{25} + \frac{\lambda + 2}{25}\right) r_{i_{zw}+2} \le \lambda r_{i_{zw}+2} \qquad (\text{since } i_y \le i_{zw} \text{ and } \lambda = 5^{20}(1 + \epsilon^{-1})),
\end{aligned}
\tag{49}
$$

as claimed. Similarly, $w' \in NC[x']$. Therefore, $(z',w') \in \mathrm{Aug}(x,0,\gamma_2)$.

$\square$

Proof of Lemma 16. We find a P-jump from the set of cross edges of $E(G)$ if there is a long edge in some prefix of P. Otherwise, we claim that there exist P-jump from the set of argumented cross edges of nodes in LNF.

Let $x_1 = \mathrm{LCA}(u_1)$ and $x_2, x_3, \ldots x_l$ be the ancestors of $u_2, u_3, \ldots u_l$ at level i_1, respectively. Let j be the largest integer in $[1,l]$ such that $\delta(x_1,x_j) \le \lambda r_{i_1}$. Since $\delta(\mathrm{LCA}(u_1),v_1) > \lambda r_{i_1}$, $j < l$. Let $(\hat{u}_1, \hat{u}_{j+1})$ be the original cross edge of (u_1, u_{j+1}). Because of the maximality of j, $\mathrm{lvl}(\hat{u}_1) = \mathrm{lvl}(\hat{u}_{j+1}) > i_1$. Then, $(\hat{u}_1, \hat{u}_{j+1})$ is a P-jump.

If $\mathrm{lvl}(\hat{u}_1) \le i_1 + 3\log\lambda$ then $(\hat{u}_1, \hat{u}_{j+1}) \in E^*$ by line $3 - 9$ in Algorithm 1. Using Claim 5 with $\gamma_1 = 3\log\lambda$ and $\gamma_2 = 5\log\lambda$, we obtain $\mathrm{Pa}^2(\mathrm{Aug}(\hat{u}_1,0,\gamma_2 - \gamma_1 - 2))$ is a subset of $\mathrm{Aug}(x_1,0,5\log\lambda)$. Since $\gamma_2 - \gamma_1 - 2 = 5\log\lambda - 3\log\lambda - 2 \ge \log\lambda$, $\mathrm{Pa}^2(\mathrm{Aug}(\hat{u}_1,0,\log\lambda))$ is a subset of $\mathrm{Aug}(x_1,0,5\log\lambda)$ and hence is in E^*. Similarly, $\mathrm{Pa}^2(\mathrm{Aug}(\hat{u}_{j+1},0,\log\lambda))$ is also in E^*.

The last case is when $\mathrm{lvl}(\hat{u}_1) > i_1 + 3\log\lambda$. Let $(\tilde{u}_j, \tilde{u}_{j+1})$ be the original cross edge of (u_j, u_{j+1}). By line $3 - 9$, $(\tilde{u}_j, \tilde{u}_{j+1}) \in E^*$. Let $\hat{i}$ and $\tilde{i}$ be the levels of $\hat{u}_{j+1}$ and $\tilde{u}_{j+1}$, respectively. We claim that $\tilde{i} - 2 \le \hat{i} \le \tilde{i} + 2$, and therefore, there exists a good cross edge of (u_1, u_{j+1}) in E^*. To do that, we show the following claim:

Claim 6. $\frac{1}{2} \le \frac{\delta(\hat{u}_1,\hat{u}_{j+1})}{\delta(\tilde{u}_j,\tilde{u}_{j+1})} \le 2$.

Proof. We first prove that $\delta(\hat{u}_1, \hat{u}_{j+1}) \le 2\delta(\tilde{u}_j, \tilde{u}_{j+1})$. Since $\mathrm{lvl}(\hat{u}_1) > i_1 + 3\log\lambda$, $\delta(\hat{u}_1, \hat{u}_{j+1}) \ge \lambda^3 r_{i_1}$. By triangle inequality, $\delta(u_1, u_j) \le (\lambda + 32)r_{i_1} \le \frac{(\lambda+32)\delta(\hat{u}_1,\hat{u}_{j+1})}{\lambda^3} \le \frac{\delta(\hat{u}_1,\hat{u}_{j+1})}{\lambda}$ because $\lambda^2 \ge 2\lambda \ge \lambda + 32$. By Item 2 of Property 3, $\delta(\hat{u}_1, \hat{u}_{j+1}) \le \frac{\delta(u_1,u_{j+1})}{1-5^{-3}\epsilon}$. Using the triangle inequality,

$$\delta(u_1, u_j) \le (\lambda + 32)r_{i_1} \le \frac{\delta(\hat{u}_1, \hat{u}_{j+1})}{\lambda} \le \frac{\delta(u_1, u_{j+1})}{\lambda \cdot (1 - 5^{-3}\epsilon)} \le \frac{\delta(u_1, u_j) + \delta(u_j, u_{j+1})}{\lambda \cdot (1 - 5^{-3}\epsilon)} \qquad , \qquad (50)$$

implying that $\delta(u_1, u_j) \le \frac{\delta(u_j, u_{j+1})}{\lambda \cdot (1 - 5^{-3}\epsilon) - 1} \le \epsilon\delta(u_j, u_{j+1})$. By triangle inequality, $\delta(u_1, u_{j+1}) \le \delta(u_j, u_{j+1}) + \delta(u_1, u_j) \le (1 + \epsilon)\delta(u_j, u_{j+1})$. Then, we have:

$$
\begin{aligned}
\delta(\hat{u}_1, \hat{u}_{j+1}) &\le (1 + 5^{-3}\epsilon)\delta(u_1, u_{j+1}) && \text{(by It. 2 of Prop. 3)} \\
&\le (1 + 5^{-3}\epsilon)(1 + \epsilon)\delta(u_j, u_{j+1}) \le (1 + \epsilon)^2\delta(u_j, u_{j+1}) \\
&\le \frac{(1 + \epsilon)^2}{1 - 5^{-3}\epsilon}\delta(\tilde{u}_j, \tilde{u}_{j+1}) && \text{(by It. 2 of Prop. 3)} \\
&\le 2\delta(\tilde{u}_j, \tilde{u}_{j+1}) \qquad ,
\end{aligned}
\tag{51}
$$

as claimed. Similarly, $\delta(\hat{u}_1, \hat{u}_{j+1}) \ge 1/2 \cdot \delta(\tilde{u}_j, \tilde{u}_{j+1})$. $\qquad\square$

Hence, from Item 4 of Property 3, we have $\hat{i} \le \log \frac{\delta(\hat{u}_1, \hat{u}_{j+1})}{\lambda} + 2$ and $\tilde{i} \ge \log \frac{\delta(\tilde{u}_j, \tilde{u}_{j+1})}{\lambda}$. Therefore,

$$\hat{i} - \tilde{i} \le \log \frac{\delta(\hat{u}_1, \hat{u}_{j+1})}{\lambda} + 2 - \log \frac{\delta(\tilde{u}_j, \tilde{u}_{j+1})}{\lambda} = \log \frac{\delta(\hat{u}_1, \hat{u}_{j+1})}{\delta(\tilde{u}_j, \tilde{u}_{j+1})} + 2 < 3 \qquad , \qquad (52)$$

implying that $\hat{i} \le \tilde{i} + 2$ since both $\hat{i}$ and $\tilde{i}$ are integers. Similarly, $\hat{i} \ge \tilde{i} - 2$.

Let (w_1, w_{j+1}) be the 2-cross edge of (u_1, u_{j+1}), i.e., the ancestor at level $\hat{i} + 2$ of $(\hat{u}_1, \hat{u}_{j+1})$. (w_1, w_{j+1}) is a P-jump by definition. We complete our proof by showing that $(w_1, w_{j+1}) \in E^*$. By line $3 - 9$, $\text{Aug}(\tilde{u}_{j+1}, 0, 5\log\lambda) \subseteq E^*$. Observe that $\text{lvl}(w_{j+1}) = \hat{i} + 2 \in [\tilde{i}, \tilde{i} + 4]$, implying that w_{j+1} is an ancester of $\tilde{u}_{j+1}$ at a level lower than or equal to $\tilde{i} + 4$. Hence, $(w_1, w_{j+1}) \in \text{Aug}(\tilde{u}_{j+1}, 0, 5\log\lambda)$. Furthermore, by Claim 5, we also obtain $\text{Pa}^2(\text{Aug}(w_1, 0, \log\lambda))$ and $\text{Pa}^2(\text{Aug}(w_{j+1}, 0, \log\lambda))$ are in E^*. $\qquad\square$

To construct a P-detour, we keep finding a jump recursively on subpaths of P until we reach a complete end. We later prove that the set of those jumps has a structure called P-*stair-jump*.

An oriented cross edge $(x \to y)$ is a cross edge with direction from x to y. We formally define a P-stair-jump.

Definition 5. *Given a path $P = \{u_1, u_2, \dots u_l\}$ of G, an ordered set of oriented cross edges $\overrightarrow{D} = \{(x_1 \to y_1), (x_2 \to y_2), \dots (x_h \to y_h)\}$ is a P-stair-jump if:*

- *For any k, (x_k, y_k) is a good cross edge of (u_{a_k}, u_{b_k}), $a_k < b_k$, $a_1 = 1$ and $a_{k+1} = b_k$ for any $k < |\overrightarrow{D}|$.*

- *(x_k, y_k) is a P_{a_k}-jump with $P_j = \{(u_j, u_{j+1}), (u_{j+1}, u_{j+2}), \dots (u_{l-1}, u_l)\}$ for every $j \in [1, l]$.*

- *$\text{lvl}(x_1) < \text{lvl}(x_2) < \dots < \text{lvl}(x_h)$.*

See Figure 5 for an illustration. The set D containing undirected versions of cross edges in $\overrightarrow{D}$ is an *undirected P-stair-jump*. The node y_h is the *tail* of D and $\overrightarrow{D}$.
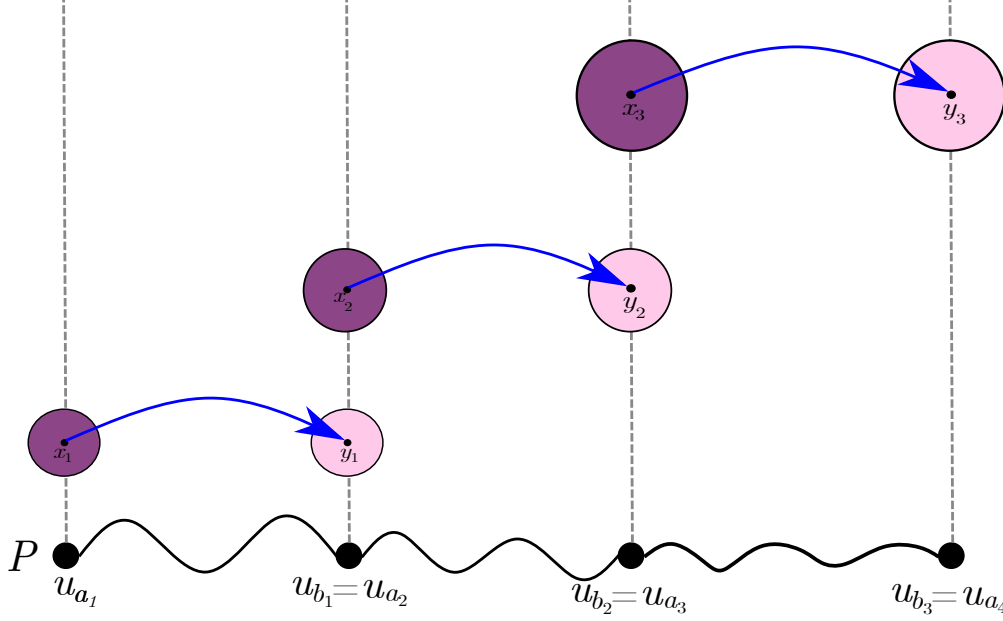
Figure 5: An example of a P-stair-jump with three cross edges. Note that $u_{a_1} = u$. Each pink node y_k is a descendant of a purple node x_{k+1}. When we construct a P-stair-jump later in the proof of Lemma 15, every pink node is incomplete and every purple node is complete.

We prove that the total weight of $\vec{D}$ is approximate the weight of the highest cross edge.

Claim 7. *Let u and v be two points in X and $P = (u = u_1, u_2, \ldots u_l = v)$ be the shortest path from u to v in G. For any P-stair-jump $\vec{D} = \{(x_1 \to y_1), (x_2 \to y_2), \ldots (x_h \to y_h)\}$ with the last point u',*
$$\frac{\lambda}{250(1+\epsilon)^2} r_{\text{lvl}(x_h)} \le \delta(u_1, u') \le \frac{2\lambda}{1-\epsilon} r_{\text{lvl}(x_h)}.$$

Proof. By Definition 5, we have $\text{lvl}(x_1) < \text{lvl}(x_2) < \ldots \text{lvl}(x_h)$. By Definition 5, there exists $\{a_i, b_i\}_{i \in [1,h]}$ such that $a_1 = 1, a_{k+1} = b_k$ for every $k \in [1, h-1]$ and (x_k, y_k) is a good cross edge of (u_{a_k}, u_{b_k}) for every $k \in [1, h]$. Note that $u' = u_{b_h}$. For every $k \in [1, h]$, since (x_k, y_k) is a cross edge, $\delta(x_k, y_k) \le \lambda r_{\text{lvl}(x_t)}$. Hence,

$$\sum_{k=1}^{h} \delta(x_k, y_k) \le \lambda \cdot \sum_{k=1}^{h} r_{\text{lvl}(x_k)} \le 2\lambda \cdot r_{\text{lvl}(x_h)} \tag{53}$$

by a geometric sum. By Item 2 of Property 3, $1 - 5^{-3}\epsilon \le \frac{\delta(x_k, y_k)}{\delta(u_{a_k}, u_{b_k})} \le 1 + 5^{-3}\epsilon$ for any k. Then, using the triangle inequality, we obtain:

$$\delta(u_1, u_{b_h}) \le \sum_{k=1}^{h} \delta(u_{a_k}, u_{b_k}) \le \frac{1}{1 - 5^{-3}\epsilon} \sum_{k=1}^{h} \delta(x_k, y_k) \le \frac{2\lambda r_{\text{lvl}(x_h)}}{1 - 5^{-3}\epsilon} \le \frac{2\lambda r_{\text{lvl}(x_h)}}{1 - \epsilon}. \tag{54}$$

39

For the lower bound, since G is an $(1 + \epsilon)$-spanner of X, we have:

$$\delta(u_1, u_{b_h}) \geq \frac{\mathbf{dist}_G(u_1, u_{b_h})}{1 + \epsilon} = \frac{1}{1 + \epsilon} \sum_{k=1}^{h} \mathbf{dist}_G(u_{a_k}, u_{b_k})$$

$$\geq \frac{1}{1 + \epsilon} \sum_{k=1}^{h} \delta(u_{a_k}, u_{b_k}) \geq \frac{1}{(1 + \epsilon)(1 + 5^{-3}\epsilon)} \sum_{k=1}^{h} \delta(x_k, y_k) \qquad \text{(by It. 2 of Prop. 3)}. \tag{55}$$

Since (x_k, y_k) is a κ-cross edge of (u_{a_k}, u_{b_k}) with $0 \leq \kappa \leq 2$, $\delta(x_k, y_k) \geq \left(\frac{\lambda}{6 \cdot 5^2} - 4\right) r_{\mathrm{lvl}(x_k)} \geq \frac{\lambda}{250} \cdot r_{\mathrm{lvl}(x_k)}$ by Item 1 of Property 3. Thus, we have:

$$\delta(u_1, u_{b_h}) \geq \frac{1}{(1 + \epsilon)(1 + 5^{-3}\epsilon)} \sum_{k=1}^{h} \frac{\lambda}{250} \cdot r_{\mathrm{lvl}(x_k)} \geq \frac{1}{(1 + \epsilon)^2} \frac{\lambda}{250} \cdot r_{\mathrm{lvl}(x_k)} \qquad , \tag{56}$$

as claimed. $\qquad\square$

In the next lemma, we show that the highest cross edge of a P-stair-jump $\overrightarrow{D}$ is approximately equal to the total distance from u to the last point of $\overrightarrow{D}$, meaning that there is a P-detour whose level close to the highest cross edge in $\overrightarrow{D}$.

Lemma 17. *Let u and v be two points in X and $P = (u = u_1, u_2, \ldots u_l = v)$ be the shortest path from u to v in G. For any P-stair-jump $\overrightarrow{D} = \{(x_1 \to y_1), (x_2 \to y_2), \ldots (x_h \to y_h)\}$ with the last point u', if y_h is complete, there exists a good cross edge (x, y) of (u_1, u') in $\mathrm{Pa}^2(\mathrm{Aug}(x_h, 0, \lambda))$ such that x and y are complete.*

Proof. Let $\{(a_k, b_k)\}_{k \in [1,h]}$ be the notation as in Definition 5. We claim that the original cross edge of (u, u_{b_h}) locates at a level not too far from $\mathrm{lvl}(x_h)$.

Claim 8. *For every integer $k \in [1, h]$, let i_k be the level of the original cross edge of (u, u_{b_k}). Then, $\mathrm{lvl}(x_k) - 3 \leq i_k \leq \mathrm{lvl}(x_k) + 3$.*

Proof. Let $e = (u, u_{b_k})$ and $(\check{u}, \check{u}_{b_k})$ be the original cross edge of e. From Claim 7, $\frac{\lambda r_{\mathrm{lvl}(x_k)}}{250(1+\epsilon)^2} \leq \delta(u_1, u_{b_k}) \leq \frac{2\lambda r_{\mathrm{lvl}(x_k)}}{1-\epsilon}$. Combining with Item 2 of Property 3, we obtain:

$$\delta(\check{u}, \check{u}_{b_k}) \leq (1 + 5^{-3}\epsilon)\delta(u_1, u_{b_k}) \leq \frac{2\lambda r_{\mathrm{lvl}(x_k)} \cdot (1 + 5^{-3}\epsilon)}{1 - \epsilon} \qquad . \tag{57}$$

Hence, by Item 4 of Property 3, we have

$$\mathrm{lvl}(\check{u}) \leq \log \frac{\delta(\check{u}, \check{u}_{b_k})}{\lambda} + 2 \leq \log \frac{2\lambda r_{\mathrm{lvl}(x_k)} \cdot (1 + 5^{-3}\epsilon)}{(1 - \epsilon)\lambda} + 2 \leq \mathrm{lvl}(x_k) + 3 , \tag{58}$$

implying that $i_k \leq \mathrm{lvl}(x_k)$ since $i_k = \mathrm{lvl}(\check{u})$. On the other hand:

$$\delta(\check{u}, \check{u}_{b_k}) \geq (1 - 5^{-3}\epsilon)\delta(u_1, u_{b_k}) \geq (1 - 5^{-3}\epsilon)\frac{\lambda r_{\mathrm{lvl}(x_k)}}{250(1 + \epsilon)^2} \cdot \qquad \text{(by Claim 7).} \tag{59}$$

40

Thus, by Item 4 of Property 3, we have:

$$\mathrm{lvl}(\check{u}) \geq \log \frac{\delta(\check{u}, \check{u}_{b_k})}{\lambda} \geq \log\left(\frac{(1 - 5^{-3}\epsilon)r_{\mathrm{lvl}(x_k)}}{250(1 + \epsilon)^2}\right)$$

$$\geq \mathrm{lvl}(x_k) + \log\left(\frac{1 - 5^{-3}\epsilon}{250(1 + \epsilon)^2}\right) \geq \mathrm{lvl}(x_k) - 3 , \tag{60}$$

as $\epsilon \leq 1/20$. Therefore, we get $\mathrm{lvl}(\check{u}) \geq \mathrm{lvl}(x_k) - 3$, or $i_k \geq \mathrm{lvl}(x_k) - 3$ as claimed. $\qquad\square$

Let $(\hat{u}, \hat{u}_{b_h})$ be the original cross edge of (u, u_{b_h}). By Claim 8, $\mathrm{lvl}(x_h) - 3 \leq \mathrm{lvl}(\hat{u}) \leq \mathrm{lvl}(x_h) + 3$. Let (z_5, t_5) be the 5-cross edge of (u, u_{b_h}), we prove that (z_5, t_5) is in $\mathrm{Aug}(x_h, 0, \lambda)$, implying that the 7-cross edge of (u, u_{b_h}) is in $\mathrm{Pa}^2(\mathrm{Aug}(x_h, 0, \log \lambda))$. Let x'_h be the ancestor of x_h at level $\mathrm{lvl}(z_5)$. x'_h exists since

$$\mathrm{lvl}(x'_h) = \mathrm{lvl}(z_5) = \mathrm{lvl}(\hat{u}) + 5 \geq \mathrm{lvl}(x_h) + 2 > \mathrm{lvl}(x_h). \tag{61}$$

The set of cross edges between nodes in $NC[x'_h]$ is a subset of $\mathrm{Aug}(x_h, 0, \log \lambda)$ because $\mathrm{lvl}(x'_h) = \mathrm{lvl}(\hat{u}) + 5 \leq \mathrm{lvl}(x_h) + 8 \leq \mathrm{lvl}(x_h) + \log \lambda$. Since (x_h, y_h) is a κ-cross edge of (u_{a_h}, u_{b_h}) with $0 \leq \kappa \leq 2$ and (x'_h, t_5) is an ancestor of (x_h, y_h), $t_5 \in NC[x'_h]$ by Item 5 of Property 3. We claim that z_5 is also in $NC[x'_h]$. Let $(\tilde{u}, \tilde{u}_{b_{h-1}})$ be the originial cross edge of $(u, u_{b_{h-1}})$. By Claim 8, $\mathrm{lvl}(\tilde{u}) \leq \mathrm{lvl}(x_{h-1}) + 3 \leq \mathrm{lvl}(x_h) + 2$, which is at most $\mathrm{lvl}(x'_h)$ by Equation (61). Hence, (z_5, x'_h) is an ancestor of $(\tilde{u}, \tilde{u}_{b_{h-1}})$ and therefore, $z_5 \in NC[x'_h]$ by Item 5 of Property 3.

We finish our proof by showing that both ends of the 7-cross edge of (u, u_{b_h}) are complete. Since the 7-cross edge of (u, u_{b_h}) is an ancestor of (z_5, t_5), it is enough to show that both z_5 and t_5 are ancestors of some complete ancestors of u and u_{b_h} by Claim 2. Since $\mathrm{lvl}(z_5) > \mathrm{lvl}(x_h) \geq \mathrm{lvl}(x_1)$ by Equation (61) and x_1 is a complete ancestor of $(u, 0)$, z_5 is an complete ancestor of $(u, 0)$. Similarly, $\mathrm{lvl}(t_5) = \mathrm{lvl}(z_5) \geq \mathrm{lvl}(y_h)$. Hence, t_5 is an ancestor of a complete ancestor y_h of (u_{b_h}). $\qquad\square$

We now prove for any shortest path from u to v in G, there is a P-detour set in E^*. For the case that two points u and v are close, the next claim shows that if the level of the original cross edge of (u, v) is less than the level of the lowest complete ancestor of any point in the shortest path between u and v, there exists a good cross edge of (u, v) in E^* (which is also a P-detour).

Claim 9. *Let u and v be two points in X, i be the level of the original cross edge of (u, v) and $P = (u = u_1, u_2, \ldots u_l = v)$ be a shortest path from u to v in G. For any point $u_j \in V(P)$ $(1 \leq j \leq l)$, if $i \leq \mathrm{lvl}(\mathrm{LCA}(u_j))$, there is a good cross edge of e in E^*.*

Proof. Let i_l, i_r be the level of the original cross edges of (u, u_j) and (u_j, v), respectively. We claim that $i_l, i_r \leq i + 2$. Let $(\hat{u}, \hat{v})$ and $(\tilde{u}, \tilde{u}_j)$ be (u, v)'s and (u, u_j)'s original cross edges, respectively. By Item 2 of Property 3, we have:

$$\delta(\tilde{u}, \tilde{u}_j) \leq (1 + 5^{-3}\epsilon)\delta(u, u_j) \leq (1 + 5^{-3}\epsilon)\mathbf{dist}_G(u, u_j)$$

$$\leq (1 + 5^{-3}\epsilon)\mathbf{dist}_G(u, v) \quad \text{(since P contains u_j and $\mathbf{dist}_G(u, v) = \mathbf{w}(P)$)}$$

$$\leq (1 + 5^{-3}\epsilon)(1 + \epsilon)\delta(u, v) \quad \text{(since G is a $(1 + \epsilon)$-spanner of X)} \tag{62}$$

$$\leq (1 + \epsilon)\frac{1 + 5^{-3}\epsilon}{1 - 5^{-3}\epsilon}\delta(\hat{u}, \hat{v}) \leq 2\delta(\hat{u}, \hat{v}) \quad \text{(by It. 2 of Prop. 3)}.$$

By Item 4 of Property 3, $i_l \leq \log \frac{\delta(\tilde{u}, \tilde{u}_j)}{\lambda} + 2 \leq \log \frac{2\delta(\hat{u}, \hat{v})}{\lambda} + 2 < i + 3$, implying that $i_l \leq i + 2$ since both i_l and i are integers. Similarly, $i_r \leq i + 2$. Let $\check{u}, \check{u}_j$ and $\check{v}$ be the ancestor of $(u, 0), (u_j, 0)$ and $(v, 0)$

at level $i + 3$. Since $i_l, i_r \leq i + 3$, $(\check{u}, \check{u}_j)$ and $(\check{u}_j, \check{v})$ are ancestors of (u, u_j)'s and (u_j, v)'s orginal cross edges, respectively. Hence, $\check{u}, \check{v} \in NC[\check{u}_j]$ by Item 5 of Property 3. Furthermore, $(\check{u}, \check{v})$ is a good cross edge of (u, v) by definition. If $\check{u}_j$ is incomplete, then all cross edges between nodes in $NC[\check{u}_j]$ is in E^* by line 16 of Algorithm 1, implying that $(\check{u}, \check{v}) \in E^*$. If $\check{u}_j$ is complete, then $\check{u}_j$ has an incomplete ancestor at level $\text{lvl}(\text{LCA}(u_j)) - 1$ by the definition of lowest complete ancestor. Since $\text{lvl}(\text{LCA}(u_j)) - 1 \geq i - 1 \geq \text{lvl}(\check{u}_j) - 5 \log \lambda$, by line 16, all (long enough) cross edges in $NC[\check{u}_j]$ are also in E^*, implying that $(\check{u}, \check{v}) \in E^*$.

$\qed$

We now ready to prove there is always a P-detour for every shortest path P in G.

Proof of Lemma 15. The idea is to find a P-stair-jump $\overrightarrow{D}$. If the tail y of $\overrightarrow{D}$ is complete, there exists a P-detour in E^* by Lemma 17. Otherwise, if y is incomplete, we keep adding another jump to $\overrightarrow{D}$. If we cannot find any jump, then the subtree containing y, denoted by T_y, in the LNF is close to both u and v, implying that we can find a detour directly from an ancestor of $(u, 0)$ to an ancestor of $(v, 0)$ in the set of augmented cross edges of nodes in T_y.

Let $P = (u = u_1, u_2, \ldots u_l = v)$, i_1 be the level of $\text{LCA}(u_1)$ and v_1 be the ancestor of v at level i_1. We denote by i the level of the original cross edge of (u, v). We claim that there is either a good cross edge of (u, v) or a P-stair-jump in E^*.

Let $\overrightarrow{D}$ be a list of oriented cross edges. At the beginning, $\overrightarrow{D} = \varnothing$ and we consider P_1. If $\delta(\text{LCA}(u_1), v_1) \leq \lambda r_{i_1}$, then the original cross edge of (u, v) has level at most i_1. By Claim 9, there exists a good cross edge of (u, v) in E^*. Otherwise, $\delta(\text{LCA}(u_1), v_1) > \lambda r_{i_1}$, by Lemma 16, E^* contains a good cross edge (x_1, y_1) of (u_1, u_{b_1}) for some $b_1 > 1$. Let $a_1 = 1$. Furthermore, we have $\text{Pa}^2(\text{Aug}(x_1, 0, \log \lambda))$ and $\text{Pa}^2(\text{Aug}(y_1, 0, \log \lambda))$ are also in E^*. If y_1 is complete, then we found a P-stair-jump with one element. Otherwise, we keep doing this process with the path P_{b_1}. Assuming that at current step, $\overrightarrow{D} = \{(x_1, y_1), (x_2, y_2), \ldots (x_h, y_h)\}$ with $h > 0$ such that:

- For every $k \in [1, h]$, (x_k, y_k) is a good cross edge of (u_{a_k}, u_{b_k}) with $a_k < b_k$, $a_1 = 1$ and $a_{k+1} = b_k$ if $k < h$.

- y_k is incomplete for every $k \in [1, h]$.

- $\text{Pa}^2(\text{Aug}(x_k, 0, \log \lambda))$ and $\text{Pa}^2(\text{Aug}(y_k, 0, \log \lambda))$ are in E^* for every $k \in [1, h]$.

See Figure 5 for an illustration. For every $k \in [1, h]$, let $i_k = \text{lvl}(x_k)$ ($= \text{lvl}(y_k)$). Let $j = b_h$, $g_j = \text{lvl}(\text{LCA}(u_j))$ and v_j be the ancestor of $(v, 0)$ at level g_j. If $\delta(\text{LCA}(u_j), v_j) \leq \lambda r_{g_j}$, we claim that there is an good cross edge between u and v in E^*.

Claim 10. *If $\delta(\text{LCA}(u_j), v_j) \leq \lambda r_{g_j}$, then there exists a good cross edge (x, y) of (u, v) in E^*.*

Proof. By Claim 7, $\frac{\lambda}{250(1+\epsilon)^2} r_{\text{lvl}(x_h)} \leq \delta(u_1, u_j) \leq \frac{2\lambda r_{\text{lvl}(x_h)}}{1-\epsilon}$. Let $(\hat{u}, \hat{v})$ be the original cross edge of (u, v) and i is the level of $\hat{u}$. We have $\delta(\hat{u}, \hat{v}) \leq \lambda r_i$. We show that $i \leq g_j + 2$. Observe that:

$$\delta(\hat{u}, \hat{v}) \leq (1 + 5^{-3}\epsilon)\delta(u, v) \leq (1 + 5^{-3}\epsilon)(\delta(u, u_j) + \delta(u_j, v)) \qquad \text{(by Eq. 2 of Prop. 3)}$$
$$\leq (1 + 5^{-3}\epsilon)\left(\frac{2\lambda r_{i_h}}{1 - \epsilon} + \delta(u_j, v)\right) \qquad \text{(by Claim 7)}$$

(63)

Since $\delta(\mathrm{LCA}(u_j), v_j) \le \lambda r_{i_j}$, $(\mathrm{LCA}(u_j), v_j)$ is an ancestor of the original cross edge of (u_j, v), denoted by $(\tilde{u}_j, \tilde{v})$. Let $\tilde{i} = \mathrm{lvl}(\tilde{v})$, then $\delta(\tilde{u}, \tilde{v}) \le \lambda r_{\tilde{i}} \le \lambda r_{g_j}$ by the minimality of the level of the original cross edge. Using Item 2 of Property 3, we get:

$$\delta(u_j, v) \le \frac{\delta(\tilde{u}_j, \tilde{v})}{1 - 5^{-3}\epsilon} \le \frac{\lambda r_{g_j}}{1 - 5^{-3}\epsilon} \le \frac{\lambda r_{g_j}}{1 - \epsilon} \qquad . \tag{64}$$

Plugging in Equation (63), we have:

$$\delta(\hat{u}, \hat{v}) \le \frac{\lambda(1 + 5^{-3}\epsilon)}{1 - \epsilon}(2 r_{i_h} + r_{g_j}) \tag{65}$$

Since y_h is incomplete, we have $i_h = \mathrm{lvl}(x_h) = \mathrm{lvl}(y_h) \le g_j - 1$, implying that $2 r_{i_h} \le r_{g_j}$. Then, by Equation (65), $\delta(\hat{u}, \hat{v}) \le \frac{2(1 + 5^{-3}\epsilon)\lambda r_{g_j}}{1 - \epsilon}$. From Item 4 of Property 3,

$$i \le \log \frac{\delta(\hat{u}, \hat{v})}{\lambda} + 2 \le \log \frac{2(1 + 5^{-3}\epsilon) r_{g_j}}{1 - \epsilon} + 2 \le g_j + 3 \qquad (\text{as } \epsilon \le 1/20) \tag{66}$$

If $i \le g_j$, by Claim 9, there is a good cross edge of (u, v) in E^*. Otherwise, $i > g_j$. Let $(\hat{u}_3, \hat{v}_3)$ be the ancestor of $(\hat{u}, \hat{v})$ at level $i + 3$. Let $\hat{u}_j$ be the ancestor of u_j at level $i + 3$. Since $i + 3 \le g_j + 6 \le g_j + 5\lambda$, $\mathrm{Aug}_{i+3}(\mathrm{LCA}(u_j)) \subseteq \mathrm{Aug}(\mathrm{LCA}(u_j), 0, 5\log\lambda) \subseteq E^*$ by line 14 − 16 of Algorithm 1. In other words, all cross edges between nodes in $NC[\hat{u}_j]$ are in E^*. From Claim 8, the level of the original cross edge of (u, u_j) is at most $\mathrm{lvl}(x_h) + 3 = i_h + 3 \le g_j + 2 \le i + 2$, then $(\hat{u}_3, \hat{u}_j)$ is an ancestor of the original cross edge of (u, u_j). By Item 5 of Property 3, $\hat{u}_3 \in NC[\hat{u}_j]$. Since $\delta(\mathrm{LCA}(u_j), v_j) \le \lambda r_{g_j}$ and $(\hat{u}_j, \hat{v}_3)$ is an ancestor of $(\mathrm{LCA}(u_j), v_j)$, $\hat{v}_3 \in NC[\hat{u}_j]$. Since $(\hat{u}_3, \hat{v}_3)$ is a good cross edge of (u, v) and $\hat{u}_3, \hat{v}_3 \in NC[\hat{u}_j]$, $(\hat{u}_3, \hat{v}_3) \in E^*$.

$\square$

By Claim 10, if $\delta(\mathrm{LCA}(u_j), v_j) \le \lambda r_{i_j}$, there exists a good cross edge of (u, v) in E^*, which gives us the lemma. Otherwise, by Lemma 16, there exists a P_j-jump (x_{h+1}, y_{h+1}) with x_h is a complete ancestor of $(u_j, 0)$ and y_h is an ancestor of $(u_{j'}, 0)$ for some $j' > j$. Set $a_{h+1} = j$ and $b_{h+1} = j'$. We append $(x_{h+1} \to y_{h+1})$ to $\vec{D}$. If y_{h+1} is incomplete, we continue the process recursively with $P_{j'}$. Otherwise $\vec{D}$ is a P-detour-path with complete tail. From Lemma 17, there exists a good cross edge (x, y) of $(u, u_{j'})$ such that both x and y are complete. Hence, (x, y) is a P-detour by definition. $\square$

7 Fast Implementation

In this section, we describe an implementation of Algorithm 1 in time $O(nf + n \log n)$, which is asymptotically optimal. We consider the running time of each step in Algorithm 1:

- Line 1: Constructing a light spanner can be done in $O(n \log n)$ time [FS16]. A net tree T is constructed in $O(n \log n)$ [HPM06].

- Line 3–9: The total number of cross edges in T is $O_{\epsilon, d}(n)$. Hence, finding all the original cross edges and adding them to E^* cost $O_{\epsilon, d}(n)$ time. For each original cross edge $(\hat{u}, \hat{v})$, we add (long enough) cross edges in $\mathrm{Aug}(\hat{u}, 0, 5\log\lambda)$ and $\mathrm{Aug}(\hat{v}, 0, \log\lambda)$ to E^*. Since each of these sets has a constant size and can be found in constant time, the for loop in line 3–9 can be implemented in $O(n)$ time.

- Line 11–22: In this (nested) for loop, we have two for loops. We indeed do not perform ζ loops since there are many levels with no cross edge. Since the number of cross edges is at most $O(n)$, we actually only need to consider the running time for $O(n)$ loops.

    - Line 12–16: For each node x, finding whether x has a small descendant w such that w has at most f leaves (recall that such w is called incomplete) needs constant time. Adding cross edges between nodes in $NC[x]$ to E^* also requires only constant time since the size of $NC[x]$ is constant due to the packing bound (Lemma 1).
    - Line 19–22: Since H has at most $O(nf)$ edges, adding egdes to H in line 22 only need $O(nf)$ time in total ($O(f)$ on average). Let U be the upper bound time complexity for Algorithm 2, the amount of time needed to run this for loop is $2U + O(f)$.

In total, the time complexity of Algorithm 1 is $O(nf + n \log n + nU)$. The running time of Algorithm 2 is therefore crucial if we want to achieve optimal time. We then focus on how to implement Algorithm 2 such that its amortized cost is $O(f + \log n)$, meaning the total running time of all calls to Algorithm 2 is $O(n(\log n + f))$.

We generalize the set $\mathbf{B}(u, 4r_i)$ as the *pool set* of (u, i), denoted by $P(u, i)$, and $\mathbf{B}(u, 16r_i)$ as the *extended pool set* of (u, i), denoted by $P^+(u, i)$. We re-write Algorithm 2 as Algorithm 3.

Algorithm 3: Implement-SelectSurrogate

   Input : A node $x \in V(T)$
   Output: A surrogate set $S(x)$

1 **if** x *is small* **then**
2     $S(x) \leftarrow$ arbitrary $f + 1$ leaves of $T(x)$ if $T(x)$ has at least $f + 1$ leaves; otherwise
      $S(x) \leftarrow$ all leaves of $T(x)$;
3 **else**
4     $S' \leftarrow \{u \in X \cap P^+(x) : \deg_H(u) \geq c_2 \cdot f$ and u is not saturated$\}$;
5     $S'' \leftarrow \{u \in X \cap P(x) : \deg_H(u) < c_2 \cdot f\}$;
6   **if** $|S'| \geq f + 1$ **then**
7       $S(x) \leftarrow$ arbitrary $f + 1$ vertices in S';
8   **else**
9       $S(x) \leftarrow S' \cup \{$arbitrary $f + 1 - |S'|$ vertices in $S''\}$;

10 **return** $S(x)$;

Given a node $x = (u, i)$, we choose the set $P(x)$ and $P^+(x)$ such that Algorithm 1 with the call to Algorithm 3 in line 20 and line 21 still returns a bounded degree, bounded lightness $(1 + \epsilon)$-VFT spanner. We keep the same setting for the pool set as in Algorithm 2, i.e., $P(u, i) = \mathbf{B}(u, 4r_i)$. To maintain a set of clean and semi-saturated points, we use two lists attached to each node in the tree. For each node (u, i), we store a list Clean(u, i) of at most $4f + 4$ clean points in $P(x)$. We maintain this set by following exactly the proof of Lemma 11: when a point in Clean(u, i) becomes semi-saturated, we look down to the path of T from (u, i) to one of its leaf with a high current degree, call v. We denote this path by $P = \{x_i, x_{i-1}, \ldots x_0\}$ with x_k is the ancestor of $(v, 0)$ at level k ($x_i = (u, i)$). Then, we find the clean points near that branch to add to $\mathbf{B}(u, 4r_i)$.

A problem with this approach is that we may have to search at some level much lower than i to find enough clean points. This problem is due to the fact that for many level j, there might be

no level-j edge incident to some point in $P(x_j)$. However, we do not have to search over all the levels until getting enough $4f + 4$ clean points. For each node x, we store a pointer pointing to its descendant x' such that the subtree rooted x' also contains x's highest-degree leaf and there exists a level-j edge incident to some point in $P(x)$. Hence, at the time a point in $P(x)$ turns from clean to semi-saturated, we only need to look down at most $4f + 4$ nodes to find a substitution in $P(x)$. Since each point only turns from clean to semi-saturated once, we only need $O_{\epsilon,d}(nf)$ time in total for updating from clean to semi-saturated.

The harder task is to maintain the list of semi-saturated points in $P^+(x)$. We do not have any information about how many semi-saturated points are in $P^+(x)$. Since semi-saturated points in $P^+(x)$ are prioritized over the clean points in $P(x)$, we need to make sure that all up to $f + 1$ semi-saturated points in $P^+(x)$ can be found in an efficient way. However, there is no information about where should we look for semi-saturated points in $P^+(x)$ if we keep setting $P^+(x) = \mathbf{B}(u, 16r_i)$. Observe that it is easier if we can compute the set $P^+(x)$ recursively from the extended pool of some below nodes. Then, we want to modify $P^+(x)$ such that it still preserves the crucial properties needed to prove the maximum degree, bounded lightness, and fault-tolerant property but can be computed recursively.

From Remark 1, to ensure the correctness of Algorithm 1, it is sufficient that $\mathbf{B}(x, 12r_i) \subseteq P^+(x) \subseteq \mathbf{B}(x, 16r_i)$. Then, we define $P^+(x)$ recursively as follow:

- If $i \le 3$, $P^+(x) = \mathbf{B}(x, 16r_i)$

- If $i > 3$, let $D_{i-2}(x)$ be the set of node y at level $i - 2$ such that $\mathbf{B}(y, 12r_{i-2}) \cap \mathbf{B}(x, 12r_i) \neq \varnothing$. Then, $P^+(x) = \bigcup_{y \in D_{i-2}(x)} P^+(y)$.

We prove $P^+(x)$ possesses the required property in Remark 1.

Claim 11. *For every node x at level i, $\mathbf{B}(x, 12r_i) \subseteq P^+(x) \subseteq \mathbf{B}(x, 16r_i)$.*

Proof. If $i \le 3$, $P^+(x) = \mathbf{B}(x, 16r_i)$. Claim 11 holds trivially. Assume that Claim 11 holds for every node at any level lower than i. Let $C^+(x)$ be the set of nodes at level $i - 2$ such that for every $y \in C^+(x)$, $\mathbf{B}(y, 12r_{i-2})$ intersects $\mathbf{B}(x, 12r_i)$. By Claim 1, for every point w in X, the distance between w and its ancestor at level $i-2$ is at most $5r_{i-2}/4$. Hence, the union of $\mathbf{B}(y, 12r_{i-2})$ over all node y at level $i - 2$ covers the whole space X, implying that the union of $\mathbf{B}(y, 12r_{i-2}) \cap \mathbf{B}(x, 12r_i)$ over all $y \in C^+(x)$ covers $\mathbf{B}(x, 12r_i)$. Thus, $P^+(x)$ contains $\mathbf{B}(x, 12r_i)$.

We complete our proof by showing $\delta(x, w) \le 16r_i$ for every $w \in \bigcup_{y \in C^+(x)} P^+(y)$. For every $w \in P^+(x)$, let y be the node in $C^+(x)$ that $w \in P^+(y)$. By our induction hypothesis, $w \in \mathbf{B}(y, 16r_{i-2})$. Since $\mathbf{B}(y, 12r_{i-2}) \cap \mathbf{B}(x, 12r_i) \neq \varnothing$, $\delta(x, y) \le 12r_{i-2} + 12r_i$. By the triangle inequality,

$$\delta(x, w) \le \delta(x, y) + \delta(y, w) \le 12r_{i-2} + 12r_i + 16r_{i-2} \le r_i(12 + 12/25 + 16/25) \le 16r_i \quad,$$

implying that $w \in \mathbf{B}(x, 16r_i)$. Hence, $P^+(x) \subseteq \mathbf{B}(x, 16r_i)$. $\square$

Now, we describe how to maintain $P^+(x)$. For each node x, we store a list $C^+(x)$ of nodes at level $i - 2$ such that for every $y \in C^+(x)$, $\mathbf{B}(x, 12r_i) \cap \mathbf{B}(y, 12r_{i-2}) \neq \varnothing$. Each node in $C^+(x)$ is called an *extended child* of x and x is an *extended parent* of every node in $C^+(x)$ (a node might have multiple extended parents). Similarly, a node x is called an *extended ancestor* of y if x is the extended parent of either y or of some extended ancestor of y. The node y is then called an *extended descendant* of x. Observe that x does not have any extended descendant at a level j if

$i - j$ is odd. Similarly, there is no extended ancestor of x at level k if $k - i$ is odd. We construct an extended graph T^+ whose nodes are $V(T)$ plus $O(n)$ additional nodes that we will define later.

For any two nodes x and y in $V(T)$, $(x, y) \in E(T^+)$ if and only if x is an extended parent of y. The idea to compute $P^+(x)$ is to merge the lists of semi-saturated points from some extended descendants of x plus the new semi-saturated points. However, storing all the semi-saturated points in $P^+(y)$ for all $y \in V(T)$ will increase the time needed to find semi-saturated points since the extended pool of all extended children might share many points in common. Moreover, merging two lists with multiplicity in sublinear time is impossible. Thus, we only store the list of semi-saturated points in the extended pool of *leaf* nodes of T^+, meaning nodes with no extended child.

Information stored on each node: For each node x, we store a list of cf semi-saturated points in $P^+(x)$, denoted by SSList(x). Here, $c = \epsilon^{-O(d)}$ is the constant in Claim 12. If x is a leaf, we store all semi-saturated points in x. The lists of semi-saturated points of each node must be computed upward. We set SSList(x) to be the union (up-to cf points) of all SSList(y) such that $y \in C^+(x)$. During some iterations, there might be some points in $P^+(x)$ turning to semi-saturated and some turning to saturated. While the former ones only require a local modification on some nodes, the latter ones are more challenging to update.

Update when a point becomes semi-saturated: When some point u turns into semi-saturated, u might belong to some extended pools of nodes at a much lower level. However, it is unnecessarily costly to update every $P^+(y)$ such that $u \in P^+(y)$. In fact, a new semi-saturated point at level i only affects the extended pool sets of nodes at level i and $i + 1$ since for nodes at higher levels, we update their extended pools recursively. Hence, for each node x at level i in $V(T)$, we create an extended child x' of x and add (x, x') to T^+. $P^+(x')$ contains points in $P^+(x)$ which turn into semi-saturated during iteration i or $i - 1$. Observe that x' is a leaf of T^+. We show that the total points in $P^+(x')$ is at most $\epsilon^{-O(d)} f$.

Claim 12. *Let x be a node at level i in $V(T)$. Then, the number of points in $P^+(x)$ becoming semi-saturated during iteration i or $i - 1$ is at most cf where $c = \epsilon^{-O(d)}$.*

Proof. We only consider the number of points in $P^+(x)$ turning into semi-saturated in iteration i, those in iteration $i - 1$ will be counted similarly. A point $u \in P^+(x)$ becomes semi-saturated during iteration i only if it is a surrogate of some node y at level i. Using the triangle inequality, we have:

$$\delta(x, y) \le \delta(x, u) + \delta(u, y) \le 16r_i + 16r_i = 32r_i. \tag{67}$$

Hence, $y \in \mathbf{B}(x, 32r_i)$. By packing bound (Lemma 1), there are at most $2^{O(d)}$ such y. Furthermore, from Lemma 2, for every node y at level i, there are at most $\epsilon^{-O(d)} f$ points which are used as surrogates of y. Hence, the total number of surrogates of y becoming semi-saturated is $\epsilon^{-O(d)}$. Therefore, the total number of points in $P^+(x)$ turning into semi-saturated during iteration i is at most the number of nodes in $\mathbf{B}(x, 32r_i)$ times the maximum number of surrogates of a node turning into semi-saturated, which is $\epsilon^{-O(d)}$. $\square$

Update when a point becomes saturated: A node y is marked as DELETED if all semi-saturated points in all leaves of y have become saturated. For each node x at level i, we add a pointer pointing to a leaf of x, denoted by $\rho(x)$, such that $\rho(x)$ is not deleted. Initially, $\rho(x) = \rho(z)$ for some $z \in C^+(x)$ such that $\rho(z) \ne$ NULL. When a point u becomes saturated after iteration i,

we update SSList(x) for every x at level i and $i-1$ such that SSList(x) contains u. By packing bound (Lemma 1), there are a constant number of such nodes x. We then focus on updating only one node x at level i such that SSList(x) contains u.

We use $\rho(x)$ to find an extended leaf of x that still has more semi-saturated points. Let $z = \rho(x)$. If there exists a semi-saturated point in SSList(z), we update SSList(x) by adding that point. However, if there is no semi-saturated point in SSList(z), a recursive procedure named NodeDeletion(z) is invoked to delete z. The procedure is as follows:

For every node x' at level i or $i-1$ such that $\rho(x') = z$, we update $\rho(x')$. First, we find an extended parent z' of z such that x' is the ancestor of z'. Second, we update $\rho(z')$ by finding an extended child of z' which is not deleted and set $\rho(z')$ to be that child. If all extended children of z' are deleted, we mark z' as deleted and recursively run NodeDeletion(z'). Otherwise, we set $\rho(x') = \rho(z')$. We later argue that each call to procedure NodeDeletion($\cdot$) runs in $O(\log n)$.

Running time analysis of all updates: First, we prove that the total number of nodes in T^+ is $|V(T)| + O_d(n)$.

Claim 13. *The total number of nodes in T^+ is $|V(T)| + O_d(n)$.*

Proof. We prove that the size of the set containing extended leaves added to T^+, call A, is $O(n)$. Recall that for each node x at level i, we add an extended leaf to x at level $i-1$ containing all points in $P^+(x)$ that become semi-saturated during iteration $i-1$ or i. The leaf corresponding to x, call x', is an artificial node that contains all points in $P^+(x)$ turning into semi-saturated during iteration i or $i-1$. We call x' an *artificial leaf*. Since each point $u \in X$ only turns into semi-saturated once, it is easy to verify that there are $2^{O(d)}$ artificial leaves y such that $u \in P^+(y)$. Let A_u be the set of those artificial leaves, we have:

$$|A| \le \sum_{u \in X} |A_u| \le \sum_{u \in X} 2^{-O(d)} = n \cdot 2^{O(d)}. \tag{68}$$

Therefore, $|V(T^+)| = |V(T)| + |A| \le |V(T)| + O(n)$. $\qquad\square$

In our construction, we only focus on *relevant nodes*. A relevant node is either a leaf or a node with at least 2 extended children in T^+. Observe that a non-leaf node x at level i is relevant if there is a node y at level $i-2$ such that $\delta(x, y) \le 12r_{i-2} + 12r_i = O(r_{i-2})$. The number of such pairs (x, y) is bounded by $O(n)$ by using the similar technique as in bounding the number of cross edges (See Theorem 5.3 [CGMZ16]). Thus, the total number of nodes in T^+ is $O(n)$. The update procedure when a point becomes semi-saturated therefore needs $O(n)$ time in total. It remains to prove that NodeDeletion($\cdot$) can be done in $O(nf + n\log n)$ time in total.

Observe that for each node z, NodeDeletion(z) only updates z's extended parents and z's extended ancestors at level i and $i-1$ with i is the current iteration in Line 11 of Algorithm 1. The following observation of the packing bound implies that NodeDeletion(z) only updates a constant number of nodes.

Observation 5. *For every node x at level i of T^+, we have:*

1. *The number of x's extended children is $O(1)$.*

2. *For every level $k > i$, the number of extended ancestors of x at level k is $O(1)$.*

We complete our implementation by showing that for each node, all of its extended ancestors can be found in $O(\log n)$ time.

For each node x at level i, we store the extended ancestors of x in T^+ at level $i+1, i+2, i+4, \ldots i+2^h$ for $h \in [0, \log n]$. Since x has a constant number of ancestors at a certain level, we only need time (and space) $O(\log n)$ for each node x. Therefore, the total time complexity for this step is $O(n \log n)$. For each node x, computing the ancestor of x at level $k > i$ is done in $O(\log n)$ time.

In total, for each node in T^+, we will delete it once, and each deletion procedure takes time $O(\log n)$. The construction of T^+ requires time $O(|V(T^+)|f)$ since each node x in T^+ has at most cf points in its surrogates list SSList(x). Therefore, the total running time of computing all $P^+(\cdot)$ is then $O(n \log n + nf)$.

References

[ADD⁺93] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Computational Geometry*, 9(1):81–100, 1993, `doi:10.1007/BF02189308`. 1

[ADM⁺95] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: Short, thin, and lanky. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 489–498, 1995. 1, 5

[AS94] S. Arya and M. H. M. Smid. Efficient construction of a bounded degree spanner with low weight. In *Proc. of 2nd ESA*, pages 48–59, 1994. 1, 4

[AWY05] P. K. Agarwal, Y. Wang, and P. Yin. Lower bound for sparse Euclidean spanners. In *Proc. of 16th SODA*, pages 670–671, 2005. 1

[BCF⁺10] P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. H. M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010, `doi:10.1007/s00453-009-9293-4`. 2

[BLW19] G. Borradaile, H. Le, and C. Wulff-Nilsen. Greedy spanners are optimal in doubling metrics. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2371–2379, 2019, `doi:10.1137/1.9781611975482.145`. 3

[CDS01] D. Z. Chen, G. Das, and M. H. M. Smid. Lower bounds for computing geometric spanners and approximate shortest paths. *Discrete Applied Mathematics*, 110(2-3):151–167, 2001. 1

[CG06] H. T.-H. Chan and A. Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proc. of 17th SODA*, pages 70–78, 2006. 3, 5

[CGMZ05] T.-H. H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'05, 2005. 3, 4, 5

[CGMZ16] T. H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, 2016. Preliminary version appeared in SODA 2005. 10, 11, 47

[Che86]   L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 169–177, 1986. 1

[Cla87]   K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. of 19th STOC*, pages 56–65, 1987. 1

[CLN12]   T.-H. H. Chan, M. Li, and L. Ning. Sparse fault-tolerant spanners for doubling metrics with bounded hop-diameter or degree. In *ICALP (1)*, pages 182–193, 2012. Full version in Algorithmica'15. 2, 3, 4, 5

[CLNS13]   T. H. Chan, M. Li, L. Ning, and S. Solomon. New doubling spanners: Better and simpler. In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *Proc. 40th ICALP*, volume 7965 of *Lecture Notes in Computer Science*, pages 315–327. Springer, 2013. Full version in SICOMP'15, `doi:10.1007/978-3-642-39206-1\_27`. 2, 3, 4, 5, 6

[CZ03]   A. Czumaj and H. Zhao. Fault-tolerant geometric spanners. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, SoCG'03, pages 1–10, 2003. 1, 2, 4

[DHN93]   G. Das, P. Heffernan, and G. Narasimhan. Optimally sparse spanners in 3-dimensional euclidean space. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, SCG '93, pages 53–62, 1993. 4

[DN94]   G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. In *Proc. of 10th SOCG*, pages 132–139, 1994. 1

[ES15]   M. Elkin and S. Solomon. Optimal euclidean spanners: Really short, thin, and lanky. *J. ACM*, 62(5):35:1–35:45, 2015. 3

[FS16]   A. Filtser and S. Solomon. The greedy spanner is existentially optimal. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 9–17, 2016. to appear in SICOMP 2020, `doi:10.1145/2933057.2933114`. 11, 43

[GGN04]   J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. In *Proc. of 20th SoCG*, pages 190–199, 2004. 3, 4, 5

[GLN02]   J. Gudmundsson, C. Levcopoulos, and G. Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500, 2002. 1

[GR08a]   L. Gottlieb and L. Roditty. Improved algorithms for fully dynamic geometric spanners and geometric routing. In *Proc. of 19th SODA*, pages 591–600, 2008. 3

[GR08b]   L. Gottlieb and L. Roditty. An optimal dynamic spanner for doubling metric spaces. In *Proc. of 16th ESA*, pages 478–489, 2008. Another version of this paper is available via `http://cs.nyu.edu/~adi/spanner2.pdf`. 3, 4, 5

[HPM06]   S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006, `doi:10.1137/s0097539704446281`. 3, 43

[KG92]   J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete and Computational Geometry*, 7(1):13–28, 1992. 1

[KLMS22]   O. Kahalon, H. Le, L. Milenkovic, and S. Solomon. Can't see the forest for the trees: Navigating metric spaces by bounded hop-diameter spanners. In A. Milani and P. Woelfel, editors, *Proc. PODC*, pages 151–162. ACM, 2022, `doi:10.1145/3519270.3538414`. 3

[LNS98]   C. Levcopoulos, G. Narasimhan, and M. Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, STOC'98, pages 186–195, 1998. 1, 2, 3, 4

[LT22]   H. Le and C. Than. Greedy spanners in euclidean spaces admit sublinear separators. In J. S. Naor and N. Buchbinder, editors, *Proc. SODA*, pages 3287–3310. SIAM, 2022, `doi:10.1137/1.9781611977073.130`. 3

[Luk99]   T. Lukovszki. New results on geometric spanners and their applications. *Ph.D. thesis, Department of Computer-Science, University of Paderborn, Paderborn, Germany*, 1999. 2, 4

[NS07]   G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. 1, 3, 4, 5

[Smi09]   M. Smid. The weak gap property in metric spaces of bounded doubling dimension. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, pages 275–289. Springer-Verlag, 2009. 3

[Sol14]   S. Solomon. From hierarchical partitions to hierarchical covers: optimal fault-tolerant spanners for doubling metrics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC'14, pages 363–372, 2014. 2, 3, 4, 5, 6, 10, 11