Proximity Graphs for Similarity Search: Fast Construction, Lower Bounds, and Euclidean Separation

Shangqi Lu
HKUST-Guangzhou
shangqilu@hkust-gz.edu.cn

Yufei Tao
CUHK
taoyf@cse.cuhk.edu.hk

September 8, 2025

Abstract

Proximity graph-based methods have emerged as a leading paradigm for approximate nearest neighbor (ANN) search in the system community. This paper presents fresh insights into the theoretical foundation of these methods. We describe an algorithm to build a proximity graph for $(1+\epsilon)$-ANN search that has $O((1/\epsilon)^\lambda \cdot n \log \Delta)$ edges and guarantees $(1/\epsilon)^\lambda \cdot \text{polylog}\, \Delta$ query time. Here, n and Δ are the size and aspect ratio of the data input, respectively, and $\lambda = O(1)$ is the doubling dimension of the underlying metric space. Our construction time is near-linear to n, improving the $\Omega(n^2)$ bounds of all previous constructions. We complement our algorithm with lower bounds revealing an inherent limitation of proximity graphs: the number of edges needs to be at least $\Omega((1/\epsilon)^\lambda \cdot n + n \log \Delta)$ in the worst case, up to a subpolynomial factor. The hard inputs used in our lower-bound arguments are non-geometric, thus prompting the question of whether improvement is possible in the Euclidean space (a key subclass of metric spaces). We provide an affirmative answer by using geometry to reduce the graph size to $O((1/\epsilon)^\lambda \cdot n)$ while preserving nearly the same query and construction time.

1 Introduction

Approximate nearest neighbor (ANN) search is fundamental to similarity retrieval and plays an imperative role in a wide range of database applications, such as recommendation systems, entity matching, multimedia search, DB for AI, and so on. In the past decade, proximity graph-based approaches (e.g., HNSW [22]) have become a dominant paradigm for ANN search in the system community. Numerous articles [6,7,12–14,16,19,21–24,26,27,29] in venues like SIGMOD, VLDB, NeurIPS, etc. have demonstrated proximity graphs' superior empirical performance on real-world data, even against methods with solid worst-case guarantees. Despite their empirical success, however, the theoretical underpinnings of proximity graphs remain largely unexplored. This raises a critical question:

> Is the performance of proximity graphs driven by specific properties of the datasets evaluated, or do they possess inherent theoretical strengths?

Given the vast popularity of proximity graphs, we believe that there is an urgent need to deepen our understanding of their combinatorial nature, thereby enabling us to analyze and predict their efficacy across diverse contexts.

1.1 Problem Definitions

We consider a metric space $(\mathcal{M}, D)$ where

- $\mathcal{M}$ is a (possibly infinite) set where each element is called a *point*;
- D is a function that, given two points $p_1, p_2 \in \mathcal{M}$, computes in constant time a non-negative real value as their *distance*, denoted as $D(p_1, p_2)$.

The function D satisfies (i) identity of indiscernibles: $D(p_1, p_2) = 0$ if and only if $p_1 = p_2$, (ii) symmetry: $D(p_1, p_2) = D(p_2, p_1)$, and (iii) triangle inequality: $D(p_1, p_2) \leq D(p_1, p_3) + D(p_2, p_3)$.

Let P be a set of $n \geq 2$ points from $\mathcal{M}$, which we refer to as the *data points*. Given a point $q \in \mathcal{M}$, a point $p^* \in P$ is a *nearest neighbor* (NN) of q if $D(p^*, q) \leq D(p, q)$ holds for all $p \in P$. For a value $\epsilon \in (0, 1]$, a point $p \in P$ is called a $(1+\epsilon)$-*approximate nearest neighbor* of q if $D(p, q) \leq (1+\epsilon) \cdot D(p^*, q)$.

Consider a simple directed graph G, where each point of P corresponds to a vertex in G, and vice versa. We call G a $(1+\epsilon)$-*proximity graph* (PG) if, given any query point $q \in \mathcal{M}$ and any data point $p_{\text{start}} \in P$, the following procedure always returns a $(1+\epsilon)$-ANN of q:

greedy(p_{start}, q)
1. $p^\circ \leftarrow p_{\text{start}}$ /* the first hop */
2. **repeat**
3. $\quad p^+_{\text{out}} \leftarrow$ the out-neighbor of p° closest to q /* $p^+_{\text{out}} = $ nil if p° has no out-neighbors */
4. $\quad$ **if** $p^+_{\text{out}} = $ nil **or** $D(p^\circ, q) \leq D(p^+_{\text{out}}, q)$ **then return** p°
5. $\quad p^\circ \leftarrow p^+_{\text{out}}$ /* the next hop */

At each p° — henceforth referred to as a *hop vertex* — the procedure computes $D(p_{\text{out}}, q)$ for every out-neighbor p_{out} of p°. The sequence of hop vertices (a.k.a. data points) visited have strictly descending distances (i.e., $D(p^\circ, q)$) to q.

Although **greedy** always returns a correct answer, it can be slow because it may need to visit a long senquence of vertices before termination. However, a good $(1+\epsilon)$-PG should allow **greedy** to find an $(1+\epsilon)$-ANN after only a small number of distance computations. Formally, we say that G ensures *query time* Q if the following algorithm always returns a $(1+\epsilon)$-ANN of q, regardless of the choice of p_{start} and q:

query(p_{start}, q, Q)
1. run **greedy**(p_{start}, q) until it self-terminates or has computed Q distances
2. **if** self-termination **then return** the output of **greedy**
3. **else return** the last hop vertex p° visited by **greedy**

Note that a "Q query time" guarantee defined as above directly translates into a maximum running time of $O(Q)$ because distance calculation is the bottleneck of **greedy**.

Proximity graphs definitely exist: the complete graph G — namely, there is an edge from a data point to every other data point — is a proximity graph for any $\epsilon > 0$. However, this G has $\Theta(n^2)$ edges and can only ensure a query time of $\Omega(n)$. Research on proximity graphs revolves around two questions:

- **Q1:** How to build a smaller proximity graph with faster query time?
- **Q2:** What are the limitations of proximity graphs?

Besides ϵ and n, we will describe our results using two other parameters, as introduced next.

Aspect Ratio. In general, for any subset $X \subseteq \mathcal{M}$, its *diameter* — denoted as $diam(X)$ — is the maximum distance of two points in X, while its *aspect ratio* is the ratio between $diam(X)$ and the smallest inter-point distance in X. The first extra parameter we adopt is the aspect ratio of P, denoted as Δ.

Doubling Dimension. For any point $q \in \mathcal{M}$ and any real value $r \geq 0$, define $B(q, r)$ — referred to as a *ball* with radius r — as the set $\{p \in \mathcal{M} \mid D(p, q) \leq r\}$. The second extra parameter we adopt is the *doubling dimension* of the metric space $(\mathcal{M}, D)$. This is the smallest value λ satisfying the following condition: for any $r > 0$, every ball of radius r can be covered by the union of at most 2^λ balls of radius $r/2$. The value λ measures the "intrinsic dimensionality" of a metric space. Our discussion throughout the paper will assume λ to be bounded by a constant.

Mathematical Conventions. For an integer $x \geq 1$, the notation $[x]$ represents the set $\{1, 2, ..., x\}$. If p is a point in $\mathbb{R}^d$, its i-th coordinate is denoted as $p[i]$ for each $i \in [d]$. Given two points $p, q \in \mathbb{R}^d$, we use $L_2(p, q)$ and $L_\infty(p, q)$ to represent their distance under the L_2 and L_∞ norms, respectively. All angles are measured in Radians, and all logarithms have base 2. In a directed graph, we use the notation (u, v) to represent a directed edge from vertex u to vertex v. By saying that a random event occurs "with high probability" (w.h.p. for short), we mean that the event happens with probability at least $1 - 1/n^c$ where c can be set to an arbitrarily large constant.

1.2 Previous Work

Many methods have been proposed in the system community for proximity graph construction, with small-world graph [21], DiskANN [19], NSG [12], and HNSW [22] being notable examples. Although the efficiency of these methods has been demonstrated through extensive experiments, limited research has focused on understanding their theoretical characteristics. In [18], Indyk and Xu addressed the issue by conducting a comprehensive analysis of the worst-case performance of the existing PG-based methods. They found that DiskANN is the only method that enjoys non-trivial guarantees. Specifically, DiskANN builds a $(1+\epsilon)$-PG in $O(n^3)$ time that has $O((1/\epsilon)^\lambda \cdot n \log \Delta)$ edges and guarantees a query time of $O((1/\epsilon)^\lambda \cdot \log^2 \Delta)$.

Diwan et al. [11] considered the special scenario where $P = \mathcal{M}$ (or equivalently, every query point originates from P). In that case, they proved the existence of a $(1 + \epsilon)$-PG that has $O(n^{1.5} \log n)$ edges and ensures $O(\sqrt{n} \log n)$ query time. Given any $q \in P$, their PG allows the greedy algorithm of Section 1.1 to find an exact NN of q (which is q itself) within the aforementioned time bound; hence, their construction works for any value of ϵ.

An important class of metric spaces is the Euclidean space $\mathbb{R}^d$ coupled with the L_2 norm, for which several authors have studied how to leverage geometry to build PGs, assuming the dimensionality d to be a constant. Specifically, the construction by Arya and Mount [3] produces a $(1+\epsilon)$-PG with $O((1/\epsilon)^d \cdot n)$ edges but $\Omega(n)$ query time. Clarkson [8] presented another construction that yields a $(1+\epsilon)$-PG with $O((1/\epsilon)^{(d-1)/2} \cdot n \log(\Delta/\epsilon))$ edges and $O((1/\epsilon)^{(d+1)/2} \cdot \log(\Delta/\epsilon) \cdot \log \Delta)$ query time. The construction of [3] and [8] takes $O((1/\epsilon)^d \cdot n^2)$ expected time and $O((1/\epsilon)^{d-1} \cdot n^2 \log(\Delta/\epsilon))$ time, respectively. As the doubling dimension λ of $\mathbb{R}^d$ satisfies $d \leq \lambda = O(d)$, Clarkson's bounds are better than those of DiskANN on $(\mathbb{R}^d, L_2)$.

We emphasize that the purpose of this work is to explore the theory of proximity graphs — in particular, to seek answers for **Q1** and **Q2** in Section 1.1 — rather than designing new data structures for ANN search. Readers interested in **non**-PG-based ANN structures with strong performance guarantees may refer to the representative works [1, 2, 4, 9, 10, 15, 17, 20] and the references therein.

1.3 Our Results

We present new answers to both questions **Q1** and **Q2**. Our first main result is an algorithm for building proximity graphs:

Theorem 1.1. *For a metric space with a constant doubling dimension λ, there is a $(1+\epsilon)$-PG that has $O((1/\epsilon)^\lambda \cdot n \log \Delta)$ edges and $O((1/\epsilon)^\lambda \cdot \log^2 \Delta)$ query time, where n and Δ are the size and aspect ratio of the data input, respectively. We can construct such a graph in $(1/\epsilon)^\lambda \cdot n \operatorname{polylog}(n\Delta)$ time.*

Our algorithm, which improves DiskANN, is the first in the literature whose construction cost avoids a quadratic dependence on n. We then continue to explore whether the graph size in Theorem 1.1 can be significantly reduced, in particular:

- **Q2.1:** For constant ϵ, both DiskANN and our solution produce a graph of $O(n \log \Delta)$ edges. Is the $\log \Delta$ factor an artifact? In other words, for constant ϵ, is there a $(1+\epsilon)$-PG of $O(n)$ edges? What if the query time is allowed to be arbitrarily large?

- **Q2.2:** For non-constant ϵ, both DiskANN and our solution have the term $(1/\epsilon)^\lambda \cdot n$ in the graph size. Is the $(1/\epsilon)^\lambda$ factor necessary? Again, what if the query time is allowed to be arbitrarily large?

Our second theorem gives lower bounds justifying both the $\log \Delta$ and the $(1/\epsilon)^\lambda$ factors.

Theorem 1.2. *The following statements are true:*

1. *For any integers Δ and n that are powers of 2 satisfying $n \geq 2$ and $n^2 \leq 2\Delta \leq 2^n$, there is a set P of $\Theta(n)$ points with aspect ratio Δ from a metric space whose doubling dimension is 1, such that any 2-PG for P must have $\Omega(n \log \Delta)$ edges, regardless of the query time allowed.*

2. *Given any integers $s \geq 2$, $t \geq 1$, and constant $d \geq 1$, there is a set P of $n = s^d \cdot t$ points with aspect ratio $\Delta = O(n)$ from a metric space whose doubling dimension λ is at most $\log(1+2^d)$ such that, for $\epsilon = 1/(2s)$, any $(1+\epsilon)$-PG for P must have $\Omega(s^d \cdot n)$ edges, regardless of the query time allowed.*

Statement (1) of Theorem 1.2 answers **Q2.1** in a straightforward manner (it is worth mentioning that the theorem still holds even when the constant 2 in "2-PG" is replaced with any other constant greater than 1). To see the connection between Statement (2) and **Q2.2**, note that the difference between λ and d converges to 0 when d increases. The term $s^d \cdot n$ — which

is $(\frac{1}{2\epsilon})^d \cdot n$ — is greater than $(\frac{1}{2\epsilon})^{\lambda-\delta} \cdot n$ for any arbitrarily small constant $\delta > 0$ when d is sufficiently large. Thus, an $(1+\epsilon)$-PG must have $\Omega((1/\epsilon)^\lambda \cdot n)$ edges up to a subpolynomial factor. This also provides a justification on the construction time in Theorem 1.1, which can no longer be improved by more than a sub-polynomial factor. It is worth mentioning that the parameter t in Statement (2) permits the lower bound to hold for a wide range of ϵ.

The astute reader may have noticed from Statement (2) that, when $\epsilon = O(1/n^{1/\lambda})$, in the worst case every $(1+\epsilon)$-PG must have $\Omega(n^2)$ edges under our construction of P, essentially the worst possible! This does not contradict the result of [11] — which as mentioned before argues for the existence of a $(1+\epsilon)$-PG of size $O(n^{1.5} \log n)$ for any ϵ — because the result of [11] holds only in the (very) special case where $P = \mathcal{M}$.

The hard instances utilized to establish Theorem 1.2 are non-geometric. This prompts another intriguing question: does the Euclidean space allow the fast construction of a $(1+\epsilon)$-PG of $O((1/\epsilon)^\lambda \cdot n)$ edges and $(1/\epsilon)^\lambda \cdot \text{polylog}(n\Delta)$ query time? Our last main result answers the question in the affirmative:

Theorem 1.3. *Given any set P of n points in the metric space $(\mathbb{R}^d, L_2)$ where $d = O(1)$, there is a $(1+\epsilon)$-PG that has $O((1/\epsilon)^\lambda \cdot n)$ edges and ensures a query time of $O((1/\epsilon)^\lambda \cdot \log^2 \Delta + (1/\epsilon)^{d-1} \log n \cdot \log^2 \Delta)$, where n and Δ are the size and aspect ratio of the data input, respectively, and $\lambda = O(d)$ is the doubling dimension of $(\mathbb{R}^d, L_2)$. W.h.p., we can build such a graph in $(1/\epsilon)^\lambda \cdot n \,\text{polylog}(n\Delta)$ time.*

For constant ϵ (an important use case in practice), our PG is the first in the literature that has $O(n)$ edges and guarantees $\text{polylog}(n\Delta)$ query time, not to mention that it is also the first PG that can be constructed in $n\,\text{polylog}(n\Delta)$ time. Its size bound draws a separation between the Euclidean space and general metric spaces (as per Statement (1) of Theorem 1.2).

A Paradigm Critique. Our results enable an objective critique on the proximity-graph paradigm, at least in the regime where the doubling dimension is small. On the bright side, Theorem 1.1 shows that it *is* possible to build a good PG in time near-linear to n (expensive construction has been a major issue in the paradigm's literature). However, our hardness results in Theorem 1.2 clearly indicate that *space* is an inherent defect of the paradigm. In particular, one should abandon the hope to attain a clean space complexity of $O(n)$, except in the restricted scenario where both ϵ and Δ are constants. In contrast, the theory field has already discovered [9, 15] a data structure of $O(n)$ space that can answer a $(1+\epsilon)$-ANN query in $O(\log n) + (1/\epsilon)^{O(\lambda)}$ time, regardless of Δ. Our lower bounds, however, do not rule out a $(1+\epsilon)$-PG of $O((1/\epsilon)^\lambda \cdot n + n \log \Delta)$ edges. Finding a way to meet this bound or arguing against its possibility would make an interesting intellectual challenge.

We emphasize that it is not our objective to dismiss PGs as an inferior paradigm. The constituting concepts of the paradigm are elegant, especially the convenient flexibility in choosing the p_{start} point for **greedy**, which suggests that the paradigm may have strengths in enforcing load-balancing in network-scale distributed computing (found in "Internet-of-Things" applications).

2 A Proximity Graph with Fast Construction

This section serves as a proof of Theorem 1.1. The key of our proof is to explain how "r-nets" — a tool from computational geometry as defined below — are useful for building proximity graphs:

Given a subset $X \subseteq \mathcal{M}$ and a value $r > 0$, an *r-net* of X is a subset $Y \subseteq X$ satisfying:

- (separation property) $D(y_1, y_2) \geq r$ for any two distinct points $y_1, y_2 \in Y$;
- (covering property) $X \subseteq \bigcup_{y \in Y} B(y, r)$, i.e., for $\forall x \in X$, $\exists$ a point $y \in Y$ with $D(x, y) \leq r$.

The rest of the section is organized as follows. We will first define the proposed proximity graph in Section 2.1 and then analyze its properties, size, and query time in Sections 2.2 and 2.3. Finally, Section 2.4 will explain how to construct the graph efficiently.

2.1 The Graph

We consider that the smallest inter-point distance in P is 2 (as can be achieved by scaling D appropriately). In other words, the aspect ratio of P is $\Delta = diam(P)/2$. Define

$$h = \lceil \log diam(P) \rceil . \tag{1}$$

For each $i \in [0, h]$, define

$$Y_i = \text{a } 2^i\text{-net of } P. \tag{2}$$

Note that Y_0 must be P (as the smallest inter-point distance is 2). Furthermore, define

$$\eta = \lceil \log(1 + 2/\epsilon) \rceil \tag{3}$$
$$\phi = 1 + 2^{\eta+1}. \tag{4}$$

Clearly, $\eta \geq 2$ and $9 \leq \phi = \Theta(1/\epsilon)$.

We now formulate a graph G_{net}. Every vertex of G_{net} is a point in P and vice versa. For each $p \in P$, decide its out-edges as follows:

for each $i \in [0, h]$, create an edge (p, y) in G_{net} for every $y \in Y_i$ satisfying $D(p, y) \leq \phi \cdot 2^i$.

Proposition 2.1. *Every vertex (a.k.a. point) in G_{net} has an out-degree at least 1.*

Proof. Fix any point $p \in P$. If $p \notin Y_i$ for some $i \in [0, h]$, there is a point $y \in Y_i$ with $D(p, y) \leq 2^i$ due to the covering property. This y must be an out-neighbor of p. Next, we assume that $p \in Y_i$ for all $i \in [0, h]$.

Denote by j the highest value of i satisfying $|Y_i| \geq 2$. Let y be any point in Y_i different from p. We argue that y must be an out-neighbor of p, i.e., $D(p, y) \leq \phi \cdot 2^j$. Indeed, this is true if $j = h$ because $D(p, y) \leq diam(P) \leq 2^h$. Consider now $j < h$. It follows from the definition of j that $|Y_{j+1}| = 1$, in which case the covering property tells us $D(p, y) \leq 2^{j+1} < \phi \cdot 2^j$. □

2.2 Properties of G_{net}

Let G be a simple directed graph whose vertices have one-one correspondence to the points in P. We say that G is $(1+\epsilon)$-*navigable* if the following condition holds for every data point $p \in P$ and every query point $q \in \mathcal{M}$:

- either p is a $(1 + \epsilon)$-ANN of q,
- or p has an out-neighbor p_{out} satisfying $D(p_{\text{out}}, q) < D(p, q)$.

The following fact is folklore (see Appendix A for a proof):

Fact 2.1. G is a $(1 + \epsilon)$-PG of P if and only if G is $(1 + \epsilon)$-navigable.

Next, we show that the graph G_{net} built earlier is $(1+\epsilon)$-navigable and, therefore, a $(1+\epsilon)$-PG of P. In addition, we will establish a *log-drop property* of G_{net} that is crucial for the technical development in the later parts of the paper.

Lemma 2.2. *Fix an arbitrary point $q \in \mathcal{M}$, and an arbitrary point $p^\circ \in P$ that is not a $(1+\epsilon)$-ANN of q. Define*

$$p^+_{out} = \textit{the out-neighbor of } p^\circ \textit{ closest to } q. \tag{5}$$

Both of the following statements are true:

1. $D(p^+_{out}, q) < D(p^\circ, q)$.
2. **[The log-drop property]** *Let ϱ be any point in P satisfying $D(\varrho, q) \leq D(p^+_{out}, q)$. If ϱ is not a $(1+\epsilon)$-ANN of q, then*

$$\lceil \log D(\varrho, p^*) \rceil < \lceil \log D(p^\circ, p^*) \rceil \tag{6}$$

where p^* is an exact NN of q.

Statement (1) of Lemma 2.2 indicates that G_{net} is $(1+\epsilon)$-navigable.

Proof of Lemma 2.2. Define:

$$\alpha = \lceil \log D(p^\circ, p^*) \rceil \tag{7}$$
$$\beta = \max\{\alpha - \eta - 1, 0\} \tag{8}$$

where η is given in (3). Note that $\alpha \geq 1$ and $0 \leq \beta \leq h$, where h is given in (1). Furthermore, $\beta \leq \alpha - 1$. Define:

$$y^\circ = \text{an arbitrary point in } Y_\beta \text{ such that } D(p^*, y^\circ) \leq 2^\beta \tag{9}$$

Such y° exists because Y_β is a 2^β-net of P (the covering property). Note that $y^\circ \neq p^\circ$ because $D(p^\circ, p^*) > 2^{\alpha-1} \geq 2^\beta \geq D(y^\circ, p^*)$. Using $\beta \geq \alpha - \eta - 1$ (see (8)), we can derive

$$D(p^\circ, y^\circ) \leq D(p^\circ, p^*) + D(p^*, y^\circ) \leq 2^\alpha + 2^\beta = 2^\beta \cdot (2^{\alpha-\beta} + 1) \leq 2^\beta \cdot (2^{\eta+1} + 1) = \phi \cdot 2^\beta.$$

Hence, y° must be an out-neighbor of p° by how G_{net} is built.

Fact 2.2. If a point $p \in P$ is not a $(1+\epsilon)$-ANN of q, either $\log D(p, p^*) \leq \alpha - 1$ or $D(p, q) > D(y^\circ, q)$.

Before delving into the fact's proof, let us note how it implies Statements (1) and (2) of Lemma 2.2:

- Applying the fact with $p = p^\circ$ tells us $D(p^\circ, q) > D(y^\circ, q)$ because (7) suggests $\log D(p^\circ, p^*) > \alpha - 1$. This, together with the definition of p^+_{out} in (5), proves Statement (1) of Lemma 2.2.
- Applying the fact with $p = \varrho$ proves Statement (2) because $D(\varrho, q) \leq D(p^+_{\text{out}}, q) \leq D(y^\circ, q)$.

Proof of Fact 2.2. Suppose that $D(p, p^*) > 2^{\alpha-1}$ and $D(p, q) \leq D(y^\circ, q)$ hold simultaneously. We argue that in this case p must be a $(1+\epsilon)$-ANN of q, which will then validate Fact 2.2.

Consider first $\beta = 0$. From (9), we know $D(p^*, y^\circ) \leq 1$, implying that $y^\circ = p^*$ (the inter-point distance in P is at least 2). The condition $D(p, q) \leq D(y^\circ, q)$ asserts that p must be an exact NN of q. The subsequent discussion assumes $\beta = \alpha - \eta - 1 > 0$.

By $D(p, p^*) > 2^{\alpha-1}$ and $D(y^\circ, p^*) \leq 2^\beta = 2^{\alpha-\eta-1}$, we obtain

$$D(y^\circ, p^*) < D(p, p^*)/2^\eta \tag{10}$$
$$\Rightarrow D(p, q) \leq D(y^\circ, q) \leq D(y^\circ, p^*) + D(p^*, q) < D(p, p^*)/2^\eta + D(p^*, q). \tag{11}$$

On the other hand, the triangle inequality shows

$$\begin{aligned} D(p, p^*) &\leq D(p, q) + D(q, p^*) \\ \text{(by (11))} &< D(p, p^*)/2^\eta + 2 \cdot D(p^*, q) \end{aligned}$$

Subtracting $D(p, p^*)/2^\eta$ from both sides and rearranging terms gives:

$$D(p, p^*) < \frac{2^{\eta+1}}{2^\eta - 1} \cdot D(p^*, q)$$

Plugging the above into (11), we obtain:

$$D(p, q) < \frac{2}{2^\eta - 1} D(p^*, q) + D(p^*, q) \leq (1 + \epsilon) \cdot D(p^*, q)$$

where the last step used $2^\eta - 1 \geq 2/\epsilon$ (due to (3)). Hence, p is a $(1+\epsilon)$-ANN of q. □

2.3 Size and Query Time

The following is a rudimentary fact of metric spaces:

Fact 2.3. Consider any subset $X \subseteq \mathcal{M}$. If X has aspect ratio A, then $|X| = O(A^\lambda)$.

See Appendix B for a proof; recall that λ is the doubling dimension of the metric space $(\mathcal{M}, D)$. Fact 2.3 assures us that every vertex in G_{net} has an out-degree of $O(\phi^\lambda \cdot \log \Delta)$, where $\phi = \Theta(1/\epsilon)$ is given in (4). To see why, consider any point $p \in P$ and any $i \in [0, h]$. Set X to the set of points in Y_i that are out-neighbors of p. Recall that p has an edge to $y \in Y_i$ only if $D(p, y) \leq \phi \cdot 2^i$. Hence, X is a subset of the ball $B(p, \phi \cdot 2^i)$, implying that $diam(X) \leq 2\phi \cdot 2^i$. On the other hand, by definition of 2^i-net, the distance between two (distinct) points in X is at least 2^i (the separation property). Thus, the aspect ratio of X is at most 2ϕ, which by Fact 2.3 yields $|X| = O((2\phi)^\lambda) = O(\phi^\lambda)$ because $\lambda = O(1)$. As i has $h + 1 = O(\log \Delta)$ choices, p can have at most $O(\phi^\lambda \cdot \log \Delta)$ out-edges. The total number of edges in G_{net} is therefore $O((1/\epsilon)^\lambda \cdot n \log \Delta)$.

Next, we prove that G_{net} guarantees a query time of $O(\phi^\lambda \cdot \log^2 \Delta)$. Recall that each *iteration* of **greedy** — Lines 3-5 of its pseudocode in Section 1.1 — visits a new hop vertex p°. Once an iteration encounters a p° that is a $(1+\epsilon)$-ANN of q, it will definitely return a $(1+\epsilon)$-ANN of q because the hop vertices encountered in the subsequent iterations can only be closer to q.

We argue that, after at most h iterations, the current hop vertex must be a $(1+\epsilon)$-ANN of q. Our weapon is Statement (2) of Lemma 2.2. Suppose that the hop vertex p° of a certain iteration is not a $(1+\epsilon)$-ANN of q. The next iteration of **greedy** will hop to the vertex p^+_{out} defined in (5). If p^+_{out} is not a $(1+\epsilon)$-ANN either, setting $\varrho = p^+_{\text{out}}$ in Statement (2) of Lemma 2.2 yields:

$$\left\lceil \log D(p^+_{\text{out}}, p^*) \right\rceil < \left\lceil \log D(p^\circ, p^*) \right\rceil. \tag{12}$$

This means that the value of $\lceil \log D(p^\circ, p^*) \rceil$ at the beginning of an iteration must decrease by at least 1 compared to the previous iteration. This can happen at most h times before $D(p^\circ, p^*)$ drops below 2, at which moment we must have $p^\circ = p^*$.

Each iteration calculates $O(\phi^\lambda \cdot \log \Delta)$ distances because, as proved earlier, each point has an out-degree of $O(\phi^\lambda \cdot \log \Delta)$. The total query time is thus $O(\phi^\lambda \cdot \log^2 \Delta) = O((1/\epsilon)^\lambda \cdot \log^2 \Delta)$.

2.4 Construction

A primary benefit of connecting proximity graphs to r-nets is that we can leverage the rich algorithmic literature of r-nets to construct G_{net} efficiently. Consider the following procedure:

build
1. compute $Y_0, Y_1, ..., Y_h$ (which are defined in (2))
2. **for** each $i \in [0, h]$ **do**
3. **for** each point $p \in P$ **do**
4. $S \leftarrow \{y \in Y_i \mid D(p, y) \leq \phi \cdot 2^i\}$
5. create an edge (p, y) for each $y \in S$

Line 1 can be implemented in $O(n \log(n\Delta))$ time using an algorithm due to Har-Peled and Mendel [15, Theorem 3.2]. Next, we will concentrate on Lines 2-5.

At Line 2, prior to entering Line 3, we create a data structure T on Y_i that allows us to answer 2-ANN queries on Y_i. The structure should be fully dynamic, i.e., it can support both insertions and deletions. Denote by t_{qry} the worst-case time for T to answer a 2-ANN query, and by t_{upd} the worst-case time for T to perform an insertion or deletion. Immediately, it follows that T can be built in $O(|Y_i| \cdot t_{\text{upd}}) = O(n \cdot t_{\text{upd}})$ time.

At Line 4, we retrieve S using T as follows. Initially, $S = \emptyset$ and T stores exactly the points in Y_i. We then repeatedly (i) find a 2-ANN y of the point p from T, (ii) add y to S if $D(p, y) \leq \phi \cdot 2^i$, and (iii) delete y from T. The repetition continues until $D(p, y) > 2\phi \cdot 2^i$ for the first time.

We argue that the set S thus computed is precisely the one needed at Line 4. Let S_{del} be the set of points removed from T, and y_{last} be the last point removed, i.e., $D(p, y_{\text{last}}) > 2\phi \cdot 2^i$. If S_{del} misses a point $y' \in Y_i$ with $D(p, y) \leq \phi \cdot 2^i$, then y' must still remain in T. This, however, would contradict the fact that y_{last} is a 2-ANN of p (among the points remaining in T) because $2 \cdot D(p, y') \leq 2\phi \cdot 2^i < D(p, y_{\text{last}})$.

The retrieval of S_{del} incurs a running time of $O(|S_{\text{del}}| \cdot (t_{\text{qry}} + t_{\text{upd}}))$. We argue that $|S_{\text{del}}| = O(\phi^\lambda)$. Note that every point in S_{del} — except y_{last} — falls in $B(p, 2\phi \cdot 2^i)$. Thus, the diameter of $S_{\text{del}} \setminus \{y_{\text{last}}\}$ is at most $4\phi \cdot 2^i$. On the other hand, because all the points of S_{del} come from Y_i, their inter-point distance is at least 2^i. Hence, the aspect ratio of $S_{\text{del}} \setminus \{y_{\text{last}}\}$ is at most 4ϕ. It then follows from Fact 2.3 that $S_{\text{del}} \setminus \{y_{\text{last}}\}$ has $O((4\phi)^\lambda) = O(\phi^\lambda)$ points.

Prior to entering Line 5, we restore T by inserting all the points of S_{del} back in T. This costs another $O(|S_{\text{del}}| \cdot t_{\text{upd}})$ time. We can now conclude that Line 4 takes $O(\phi^\lambda \cdot (t_{\text{qry}} + t_{\text{upd}}))$ time for each point in P. As a result, the total running time of Lines 2-5 is $O(\phi^\lambda \cdot (t_{\text{qry}} + t_{\text{upd}}) \cdot n \cdot h)$.

We have shown that **build** runs in

$$O(n \log(n\Delta) + (1/\epsilon)^\lambda \cdot (t_{\text{qry}} + t_{\text{upd}}) \cdot n \log \Delta) \tag{13}$$

time overall. It remains to choose a good data structure for T. The structure of Cole and Gottlieb [20] ensures $t_{\text{qry}} = O(\log n)$ and $t_{\text{upd}} = O(\log n)$. Plugging these bounds into (13) gives the construction time claimed in Theorem 1.1.

Remark. The above discussion has assumed that we know the minimum and maximum inter-point distances in P, denoted as d_{min} and d_{max}, respectively (note: $d_{max} = diam(P)$). The assumption can be removed using standard techniques [15, 20]. More specifically, we can obtain in $O(n \log n)$ time values $\hat{d}_{min} \in [\frac{1}{2} d_{min}, d_{min}]$ and $\hat{d}_{max} \in [d_{max}, 2d_{max}]$.[1] The ratio $\hat{d}_{max}/\hat{d}_{min}$ approximates the aspect ratio Δ up to a factor of 4. Our algorithm can then be applied after replacing d_{min} and $diam(P)$ with $\hat{d}_{min}$ and $\hat{d}_{max}$, respectively.

[1] To compute $\hat{d}_{max}$, take an arbitrary point $p \in P$ and then set $\hat{d}_{max} = 2 \max_{p' \in P} D(p, p')$. To compute $\hat{d}_{min}$, first build a 2-ANN structure on P. For each point $p \in P$, use the structure to find a 2-ANN p' of p and record the distance $D(p, p')$ for p. Then, $\hat{d}_{min}$ can be set to half of the smallest recorded distance of all points.

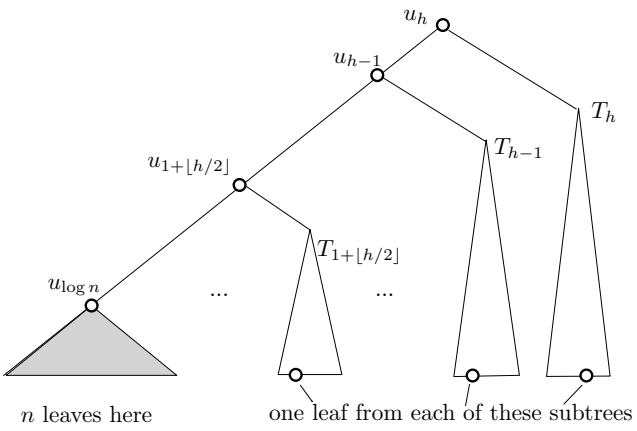


Figure 1: Hard input for Section 3

3 A Size Lower Bound under $\epsilon = 1$

This section serves as a proof of Statement (1) of Theorem 1.2. Recall that we are given integers Δ and n both of which are powers of 2; they satisfy the condition that $n \geq 2$ and $n^2 \leq 2\Delta \leq 2^n$.

We will design a metric space by resorting to a complete binary tree $\mathcal{T}$ of 2Δ leaves. The tree has $h+1$ levels where $h = \log(2\Delta)$. We number the levels bottom-up, with the leaves at level 0 and the root at level h. For each node u of $\mathcal{T}$, we use $level(u)$ to represent its level. To each edge $\{u, v\}$ of $\mathcal{T}$ — w.l.o.g., assume that u is the parent of v — we assign a *weight* that equals 1 if v is a leaf, or $2^{level(v)-1}$ otherwise.

We are now able to clarify the metric space $(\mathcal{M}, D)$:

- $\mathcal{M}$ is the set of leaves in $\mathcal{T}$;
- for any leaves v_1, v_2 in $\mathcal{T}$, their distance $D(v_1, v_2)$ equals the total weight of the edges on the unique simple path connecting v_1 and v_2 in $\mathcal{T}$.

When $v_1 \neq v_2$, our design of weights allows a simple calculation of $D(v_1, v_2)$: if the lowest common ancestor (LCA) of v_1 and v_2 is at level ℓ, then $D(v_1, v_2) = 2^\ell$. It is easy to verify that D satisfies identify of indiscernibles, symmetry, and triangle inequality. The doubling dimension of $(\mathcal{M}, D)$ is 1, as proved in Appendix C.

Next, we describe a set P of n points (a.k.a. leaves) from $\mathcal{M}$, which will serve as our hard input. Let π be the leftmost root-to-leaf path of $\mathcal{T}$. For each $i \in [0, h]$, denote by u_i be the level-i node on π, and by T_i the right subtree of u_i. We create P as follows:

- For each $i \in (h/2, h]$, we add to P one arbitrary leaf in T_i.
- Add to P all the leaves in the subtree of $u_{\log n}$.

See Figure 1 for an illustration. We will use P_1 (resp., P_2) to represent the set of leaves added in the first (resp., second) bullet. As $\log n \leq \frac{1}{2}\log(2\Delta) = h/2$, the sets P_1 and P_2 are disjoint. The total size of P is $n + \lfloor h/2 \rfloor$, which is between n and $3n/2$ (recall that $2\Delta \leq 2^n$ and hence $h \leq n$). Note that $|P_1| = n$ and $|P_2| = \lfloor h/2 \rfloor \geq 1$ because $2\Delta \geq n^2 \geq 4$. The reader can verify that $diam(P) = 2^h = 2\Delta$ and the smallest inter-point distance of P is 2; hence, the aspect ratio of P is Δ.

Consider any 2-PG G of P; by Fact 2.1, the graph G needs to be 2-navigable. We will argue that G must have an edge (v_1, v_2) for every $(v_1, v_2) \in P_1 \times P_2$. As a result, the number of edges in G must be at least $|P_1||P_2| = \Omega(n \log \Delta)$, as claimed in Statement (a) of Theorem 1.2.

Assume, for contradiction, that G has no edge (v_1, v_2) for some $v_1 \in P_1$ and $v_2 \in P_2$. We will show that G cannot be 2-navigable. W.l.o.g., assume that v_2 is in T_ℓ for some $\ell \in (h/2, h]$.

Figure 2: Hard input for Section 4

It thus holds that $D(v_1, v_2) = 2^\ell$ because the LCA of v_1 and v_2 is u_ℓ. Let us set $q = v_2$; as $q \in P$, the NN of q is v_2 itself (with the NN-distance $D(q, v_2) = 0$). Hence, v_1 is not a 2-ANN of q. We claim that v_1 has no out-neighbor in G that is closer to q than v_1, because of which G is not 2-navigable.

Let p_{out} be an arbitrary out-neighbor of v_1; to prove the claim, it suffices to explain why $D(p_{\text{out}}, q) \geq D(v_1, q)$. Clearly, $p_{\text{out}} \neq v_2$ because the edge (v_1, v_2) is absent in G. Where else can p_{out} be? If p_{out} is a descendant of u_i for some $i \leq \ell - 1$, then the LCA of p_{out} and $q = v_2$ must be u_ℓ, because of which $D(p_{\text{out}}, q) = 2^\ell = D(v_1, q)$. If p_{out} is in T_k for some $i \geq \ell + 1$, then the LCA of p_{out} and $q = v_2$ is u_i, because of which $D(p_{\text{out}}, q) = 2^i > D(v_1, q)$. As no other cases are possible, we conclude that G is not 2-navigable.

4 A Size Lower Bound under Small ϵ

This section serves as a proof of Statement (2) of Theorem 1.2. Let us start by introducing several useful notations. Recall that the statement assumes that three integers $s \geq 2, t \geq 1$, and $d \geq 1$ have been given. We use the notation $\mathbb{Z}_s$ to represent the set $\{0, 1, ..., s-1\}$. Given two points $p, w \in \mathbb{R}^d$, we define the output of $p + w$ to be the point whose i-th coordinate is $p[i] + w[i]$ for each $i \in [d]$. If S is a (possibly infinite) set of points in $\mathbb{R}^d$, given a point w, we define the *w-translated copy* of S to be $\{p + w \mid p \in S\}$.

Define $M = (\mathbb{Z}_s)^d$, which is a set of s^d points. Furthermore, define

$$W \quad = \quad \{(i \cdot 2s, 0, 0, ..., 0) \mid i \in [0, t-1]\} \tag{14}$$

namely, every point $w \in W$ has a non-zero coordinate only on the first dimension, with $w[1]$ being a multiple of $2s$ in $[0, 2s(t-1)]$. For each $w \in W$, define

$$M_w \quad = \quad \text{the } w\text{-translated copy of } M.$$

We will refer to each M_w as a *block*.

The hard data input we use is

$$P \quad = \quad \bigcup_{w \in W} M_w. \tag{15}$$

See Figure 2 for an illustration in $\mathbb{R}^2$. It is clear that $n = |P| = s^d \cdot t$.

Next, we will design the metric space $(\mathcal{M}, D)$. The set $\mathcal{M}$ includes P and one extra point q, which is non-Euclidean (i.e., q is not in $\mathbb{R}^d$). The definition of D — which will be clarified later — depends on p^*, which is a point in P. As we will see, varying the choice of p^* will result in a different D. For this reason, we will represent the distance function as D_{p^*}. This gives rise to a set of distance functions:

$$\mathcal{D} \quad = \quad \{D_{p^*} \mid p^* \in P\}. \tag{16}$$

We now specify the details of D_{p^*}. Denote by w^* the unique point in W such that p^* is in the block M_{w^*}. Then:

- for any $p_1, p_2 \in P$, define $D_{p^*}(p_1, p_2) = D_{p^*}(p_2, p_1) = L_\infty(p_1, p_2)$;
- for any $p \in P \setminus M_{w^*}$, define $D_{p^*}(p, q) = D_{p^*}(q, p) = L_\infty(p, w^*)$;
- for any $p \in M_{w^*}$ and $p \neq p^*$, define $D_{p^*}(p, q) = D_{p^*}(q, p) = s$;
- define $D_{p^*}(p^*, q) = D_{p^*}(q, p^*) = s - 1$;
- define $D(q, q) = 0$.

We prove in Appendix D:

Lemma 4.1. *For every $p^* \in P$, $(\mathcal{M}, D_{p^*})$ is a metric space with doubling dimension $\lambda \leq \log(1 + 2^d)$.*

Set $\epsilon = 1/(2s)$, as in Statement (2) of Theorem 1.2. When an algorithm constructs a $(1+\epsilon)$-PG G of P, it has access only to the points in P, but not the non-Euclidean point q. Thus, the algorithm can evaluate (as it wishes) only the distances between the points in P, but not the distance between q and any $p \in P$.

The above observation gives rise to an adversarial argument. Imagine an adversary — named Alice — who does not finalize the function D until after seeing the PG G produced by the algorithm. Of course, Alice cannot lie: her ultimate choice of D must be consistent with the distances already exposed to the algorithm. However, this will not be a problem as long as she chooses D from the class $\mathcal{D}$ (see (16)), noticing that every $D \in \mathcal{D}$ gives exactly the same $D(p_1, p_2)$ for any $p_1, p_2 \in P$.

Next, we will argue that, for every $w \in W$ (see (14) for W) and any distinct p_1, p_2 in the same block M_w, there must be an edge (p_1, p_2) in G. As each block has s^d points and there are t blocks, the total number of edges in G will then be at least

$$s^d \cdot (s^d - 1) \cdot t = \Omega(s^d \cdot n)$$

because $s \geq 2$. This will establish Statement (2) of Theorem 1.2.

Assume, for contradiction, that G has no edge (p_1, p_2) for some points p_1 and p_2 that appear in the same block M_w, for some $w \in W$. Seeing this, adversary Alice sets p^* to p_2 and thereby finalizes D to D_{p_2}. We will see that under the metric space $(\mathcal{M}, D_{p_2})$, the graph G cannot be $(1 + \epsilon)$-navigable, which by Fact 2.1 means that G cannot be a $(1 + \epsilon)$-PG of P.

Under D_{p_2}, the NN of q is p_2 with $D(q, p_2) = s - 1$; indeed, by our design $D(q, p) \geq s$ for all the other points $p \in P$. Furthermore, as p_1 is from the same block as p_2, we have $D(q, p_1) = s$. This means that p_1 is not a $(1+\epsilon)$-ANN of q because $s > s - \frac{1}{2} - \frac{1}{2s} = (s-1)(1+\frac{1}{2s}) = (s-1)(1+\epsilon)$. We claim that p_1 has no out-neighbor in G that is closer to q than p_1, because of which G is not $(1 + \epsilon)$-navigable.

Let p_{out} be an arbitrary out-neighbor of p_1; to prove our claim above, it suffices to explain why $D(p_{\text{out}}, q) \geq D(p_1, q)$. Clearly, $p_{\text{out}} \neq p_2$ because the edge (p_1, p_2) is absent in G. If p_{out} is in M_w (i.e., the block of p_2), then $D(p_{\text{out}}, q) = s = D(p_1, q)$. If p_{out} is a block different from M_w, then $D(p_{\text{out}}, q) = L_\infty(p_{\text{out}}, w)$, which is at least $s + 1$ and hence greater than $D(p_1, q)$. We thus conclude that G is not $(1 + \epsilon)$-navigable.

Finally, let us note that, under any $D \in \mathcal{D}$, the maximum distance (under D) between two points in S is less than $2st = O(n)$, while the smallest inter-point distance is 1. Hence, the aspect ratio of P is $O(n)$. This completes the proof of Theorem 1.2.

Remark. When $t = 1$, the set P degenerates into a hard input used in [15] to prove a lower bound on the *query time* of ANN data structures. Generalizing that hard input to establish a *size* lower bound for $(1 + \epsilon)$-PGs under a wide range of ϵ demands additional ideas, as we have shown above.

5 Smaller Proximity Graphs in the Euclidean Space

This section serves as a proof of Theorem 1.3. The goal is to improve the size bound of Theorem 1.1 in the special metric space of $(\mathbb{R}^d, L_2)$ by shaving-off the $\log \Delta$ factor. Our discussion will assume that the smallest inter-point distance in P is 2 (as can be achieved by scaling the dimensions of $\mathbb{R}^d$) and that the value of $diam(P)$ is available. The assumption can be removed using the same techniques explained in the remark of Section 2.4.

Let us first apply the algorithm of Theorem 1.1 to obtain a $(1+\epsilon)$-PG for the data input P in $(\mathbb{R}^d, L_2)$; we will denote the graph as G_{net}, where the subscript reminds us that it is obtained using an algorithm designed for general metric spaces. As analyzed in Section 2.3, each vertex of G_{net} has an out-degree of $O((1/\epsilon)^\lambda \cdot \log \Delta)$ such that the graph has $O((1/\epsilon)^\lambda \cdot n \log \Delta)$ edges in total (recall that λ is the doubling dimension of $(\mathbb{R}^d, L_2)$).

Consider the following drastic idea to "force" the edge number to drop by a $\log \Delta$ factor:

- sample each vertex independently with probability

$$\tau = \frac{z}{\log \Delta} \tag{17}$$

  where z is a constant to be determined later;

- keep the edges of only the sampled vertices and discard all other edges (non-sampled vertices are retained, even though their out-degrees are now 0).

In expectation, the resulting graph — denoted as G'_{net} — has $O((1/\epsilon)^\lambda \cdot n)$ edges, exactly what we hope for. However, the idea does not work (yet) because G'_{net} may no longer be a $(1+\epsilon)$-PG of P.

A second idea now kicks in: how about "patching up" G'_{net} by merging it with a "small-but-slow" $(1+\epsilon)$-PG G_{geo} that has $O((1/\epsilon)^\lambda \cdot n)$ edges but possibly very poor query time? The subscript of G_{geo} serves as a reminder that G_{geo} is indeed a blessing of geometry — Statement (1) of Theorem 1.2 has ruled out the existence of such a graph in every general metric space, no matter how bad the query time is. Formally, the merging of G'_{net} and G_{geo} gives us a graph G defined as follows:

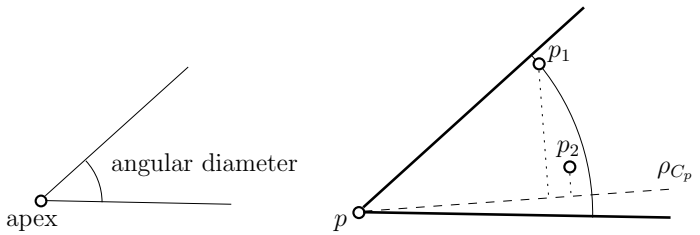
- The vertices of G have one-one correspondence to P (recall that both G'_{net} and G_{geo} have the same vertex set, i.e., P).
- The out-edge set of each point $p \in P$ in G is the union of those in G'_{net} and G_{geo}.

The rest of the section will develop the above ideas into a concrete algorithm to build a proximity graph meeting the requirements of Theorem 1.3.

5.1 Small-but-Slow Proximity Graphs

This subsection deals with the following problem: given a set P of n points in $\mathbb{R}^d$, build a $(1+\epsilon)$-PG G_{geo} of P of $O((1/\epsilon)^\lambda \cdot n)$ edges under L_2 norm. Arya and Mount [3] have proven such a graph's existence, but their construction takes $\Omega((1/\epsilon)^d \cdot n^2)$ expected time. Our objective is to achieve a near-linear dependence on n in construction time. To achieve the objective, we will introduce a variant of the so-called "θ-graph" known to permit fast construction. Then, we will prove that an appropriate choice of θ will guarantee that the graph is a $(1+\epsilon)$-PG of P.

A *halfspace* in $\mathbb{R}^d$ is the set of points $\{x \in \mathbb{R}^d \mid \sum_{i=1}^d x[i] \cdot c_i \geq c_{d+1}\}$, where $c_1, c_2, ..., c_{d+1}$ are constants. The boundary of the halfspace is the plane $\sum_{i=1}^d x[i] \cdot c_i = c_{d+1}$. A set of halfspaces is said to be in *general position* if no two halfspaces have parallel boundary planes. A (simplicial) *cone* C is the intersection of d halfspaces in general position. The *apex* of C is the intersection of



(a) Cone (b) Nearest Point on Ray

Figure 3: Key concepts underlying the θ-graph

the boundary planes of those halfspaces, and the *angular diameter* is the largest angle between two rays inside C emanating from the apex; see Figure 3(a) for a 2D example.

For any angle (measured in Radians) θ satisfying $0 < \theta < \pi$, Yao [28] gave an algorithm to compute in $O((1/\theta)^{d-1})$ time a set $\mathscr{C}$ of cones with the properties below:

- each cone in $\mathscr{C}$ has its apex at the origin and has an angular diameter at most θ;
- the union of all cones in $\mathscr{C}$ is $\mathbb{R}^d$.

Let us associate each cone $C \in \mathscr{C}$ with an arbitrary ray — denoted as ρ_C — that emanates from the origin and is contained in C.

Fix an arbitrary point $w \in \mathbb{R}^d$. For each $C \in \mathscr{C}$, we use C_w to denote the w-translated copy of C (the reader may wish to review Section 4 for what is a "w-translated copy"). Define

$$\mathscr{C}_w \quad = \quad \{C_w \mid C \in \mathscr{C}\}. \tag{18}$$

The set $\mathscr{C}_w$ comprises $|\mathscr{C}| = O((1/\theta)^{d-1})$ cones with apex w and angular diameter at most θ whose union covers $\mathbb{R}^d$. For each cone $C_w \in \mathscr{C}_w$, denote by ρ_{C_w} the w-translated copy of the ray ρ_C. We will refer to ρ_{C_w} the *designated ray* of cone C_w.

Now, set w to a point $p \in P$. We say that a cone $C_p \in \mathscr{C}_p$ is *non-empty* if C_p covers at least one other point of P besides p. For each non-empty C_p, identify a point p' as the *nearest-point-on-ray* of p in C_p as follows:

- Let S be the set of points in P covered by C_p, after excluding p itself.
- Project all the points of S onto ρ_{C_p}, i.e., the designated ray of C_p.
- Then, p' is the point whose projection on ρ_{C_p} has the smallest L_2 distance to p.

Figure 3(b) illustrates an example where $S = \{p_1, p_2\}$. Point p_1 is the nearest-point-on-ray of p in C_p because its projection on ray ρ_{C_p} is closer to p than that of p_2 (note: p_1 actually has a greater L_2 distance from p than p_2).

We are ready to define the θ-graph of P. This is a simple directed graph where

- the vertices have one-one correspondence to P;
- for any distinct points $p, p' \in P$, there is an edge from p to p' if and only if p' is the nearest-point-on-ray of p in some non-empty cone of $\mathscr{C}_p$.

Every vertex $p \in P$ in the θ-graph has an out-degree at most $|\mathscr{C}| = O((1/\theta)^{d-1})$ (at most one out-edge from p for each non-empty cone of $\mathscr{C}_p$). The total number of edges is thus $O((1/\theta)^{d-1} \cdot n)$. Such a graph can be constructed in $(1/\theta)^{d-1} \cdot n \operatorname{polylog} n$ time [5, 25].

We prove the next lemma in Appendix E.

Lemma 5.1. *A $(\epsilon/32)$-graph of P is a $(1+\epsilon)$-proximity graph of P.*

In the subsequent discussion, we will set G_{geo} to an $(\epsilon/32)$-graph of P, which can be constructed in $(1/\epsilon)^{d-1} \cdot n \operatorname{polylog} n$ time. Recall that the value d never exceeds the doubling dimension λ.

5.2 The Power of Merging

We have obtained two graphs: G'_{net} and G_{geo}. In particular, G'_{net} was obtained from G_{net} — which itself was built using Theorem 1.1 — via vertex sampling. We call a point $p \in P$ a *jackpot* point/vertex if it was sampled (in the process of building G'_{net}); recall that all the out-edges of p in G_{net} are retained by G'_{net}.

Merging G'_{net} and G_{geo} gives graph G, which has $O((1/\theta)^{\lambda} \cdot n)$ edges in expectation as explained earlier. G must be $(1+\epsilon)$-navigable (and hence a $(1+\epsilon)$-PG of P by Fact 2.1). To see why, take an arbitrary point $p \in P$ and an arbitrary query $q \in \mathbb{R}^d$ such that p is not an $(1+\epsilon)$-ANN of q. As G_{geo} is $(1+\epsilon)$-navigable (because it is a $(1+\epsilon)$-PG; see Lemma 5.1), there must exist an out-neighbor p_{out} of p in G_{geo} with $L_2(p_{\text{out}}, q) < L_2(p, q)$. Point p_{out} remains as an out-neighbor of p in G, thus confirming that G is $(1+\epsilon)$-navigable.

Next, we will prove that w.h.p. the merged graph G achieves a small query time for one single query point $q \in \mathbb{R}^d$. The next subsection will extend the result to all query points.

Let us temporarily ignore G'_{net} and focus on G_{geo}. For each $p \in P$, if we run **greedy** on G_{geo} with parameters $p_{\text{start}} = p$ and q, the algorithm visits a sequence of hop vertices (i.e., the p° vertices in the pseudocode in Section 1.1); let us denote that sequence as $\sigma_{\text{geo}}(p)$. We say that $\sigma_{\text{geo}}(p)$ is *long* if it has at least $\ln n \cdot \log \Delta$ vertices. The following is a condition we would like to have:

> **The jackpot condition:** Every long $\sigma_{\text{geo}}(p)$ (where $p \in P$) encounters a jackpot point within the first $\lceil \ln n \cdot \log \Delta \rceil$ vertices.

The jackpot condition holds w.h.p.. To see why, notice that, for each long $\sigma_{\text{geo}}(p)$, the probability for none of the first l vertices on $\sigma_{\text{geo}}(p)$ to be sampled is at most $(1-\tau)^l \leq e^{-\tau \cdot l}$, which is at most $1/n^z$ for $l = \lceil \ln n \cdot \log \Delta \rceil$ and the value of τ in (17). As there are at most n long sequences (at most one for each $p \in P$), the probability that all of them obey the jackpot condition is at least $1 - 1/n^{z-1}$.

The remainder of this subsection will prove that, under the jackpot condition, the merged graph G guarantees a query time of $O((1/\epsilon)^{\lambda} \cdot \log^2 \Delta + (1/\epsilon)^{d-1} \log n \cdot \log^2 \Delta)$ for q. Run **greedy** on G with an arbitrary p_{start} and stop the algorithm after it has visited

$$k = 1 + \lceil \log(2\Delta) \rceil$$

hop vertices that are jackpot points (provided that it has not already self-terminated). Denote by σ the sequence of hop vertices visited by **greedy**. Chop σ into subsequences, each of which (i) either ends at a jackpot vertex or is the last subsequence of σ, and (ii) includes no jackpot vertex except possibly at the end.

Lemma 5.2. *Every subsequence has at most* $\lceil \ln n \cdot \log \Delta \rceil$ *vertices.*

Proof. Consider any subsequence σ' of length at least 2. Let $p_{1\text{st}}$ be the first vertex of σ', which must be a non-jackpot point. Observe that σ' must be a prefix of $\sigma_{\text{geo}}(p_{1\text{st}})$. To see why, take any vertex p on σ' except the last vertex of σ'. As p is not a jackpot point, all its out-edges originate from G_{geo}. When p is the hop vertex, **greedy** must choose the same next hop as it would when running on G_{geo}, explaining why σ' is a prefix of $\sigma_{\text{geo}}(p_{1\text{st}})$.

Because σ' has at most one jackpot vertex, its length must be at most $\lceil \ln n \cdot \log \Delta \rceil$ under the jackpot condition. □

If **greedy** terminates without seeing k jackpot hop vertices, it must return a $(1+\epsilon)$-ANN of q because G is a $(1+\epsilon)$-PG. Next, we consider the situation where **greedy** is forced to terminate. We will argue that σ must contain at least one $(1+\epsilon)$-ANN of q. This implies that

the last vertex of σ must be a $(1+\epsilon)$-ANN of q because the vertices on σ have descending distances to q.

Assume, for contradiction, that no vertex on σ is a $(1+\epsilon)$-ANN of q. Denote by p^* the NN of q. Since we manually stopped **greedy**, the sequence σ consists of exactly k subsequences, each of which ends with a jackpot point. For each $i \in [k]$, define

$$p_i^\circ \quad = \quad \text{the last vertex of the } i\text{-th subsequence}$$

Lemma 5.3. *For each* $1 \leq i \leq \lceil \log(2\Delta) \rceil$, *it holds that* $\lceil \log L_2(p_i^\circ, p^*) \rceil > \lceil \log L_2(p_{i+1}^\circ, p^*) \rceil$.

Proof. Let us write out the vertices of the $(i+1)$-th subsequence of σ as $v_1, v_2, ..., v_l$ for some $l \leq \lceil \ln n \cdot \log \Delta \rceil$. Note that v_1 succeeds p_i° in σ and $v_l = p_{i+1}^\circ$. As G is $(1+\epsilon)$-navigable and no vertex on σ is a $(1+\epsilon)$-ANN of q, the $(1+\epsilon)$-navigable definition tells us $L_2(v_1, q) > L_2(v_2, q) > ... > L_2(v_l, q)$.

Because p_i° is not a $(1+\epsilon)$-ANN of q, by Lemma 2.2 its out-degree in G_{net} is at least 1. Let p_{out}^+ be the vertex defined in (5), i.e., the out-neighbor of p_i° in G_{net} closest to q. Because p_i° is a jackpot point, p_{out}^+ must be an out-neighbor of p_i° in G. As **greedy** always hops to the out-neighbor closest to q, we have $L_2(v_1, q) \leq L_2(p_{\text{out}}^+, q)$, which leads to $L_2(v_l, q) \leq L_2(p_{\text{out}}^+, q)$.

Now, apply Statement (2) of Lemma 2.2 by setting $p^\circ = p_i^\circ$, $\varrho = v_l$, and $D = L_2$. The application yields $\lceil \log L_2(v_l, p^*) \rceil < \lceil \log L_2(p_i^\circ, p^*) \rceil$, as claimed. □

As $L_2(p_1^\circ, p^*) \leq \mathit{diam}(P) = 2\Delta$ (recall that the smallest inter-point distance in P is 2), Lemma 5.3 implies that $L_2(p_k^\circ, p^*)$ must be strictly less than 2, indicating that $p_k^\circ = p^*$. This contradicts the fact that σ contains no $(1+\epsilon)$-ANNs of q.

Recall that each jackpot vertex has an out-degree of $O((1/\epsilon)^\lambda \cdot \log \Delta)$ in G, while each non-jackpot vertex has an out-degree of $O((1/\epsilon)^{d-1}))$ in G. Our algorithm visits $O(\log \Delta)$ jackpot vertices and $O(\log n \cdot \log^2 \Delta)$ non-jackpot vertices (due to Lemma 5.2). The total query time is therefore $O((1/\epsilon)^\lambda \cdot \log^2 \Delta + (1/\epsilon)^{d-1} \cdot \log n \cdot \log^2 \Delta)$.

5.3 Achieving High Probability

So far our query time holds w.h.p. on only one query point in $\mathbb{R}^d$. To prove Theorem 1.3, we must argue that w.h.p. the same query time holds on all query points in $\mathbb{R}^d$. The key observation that makes this possible is that, even though there are infinitely many query points, only $O(n^{2d})$ representative ones need to be considered.

The execution of **greedy** is decided by the outcome of distance comparisons of the form "*which of* $L_2(p_1, q)$ *and* $L_2(p_2, q)$ *is larger?*" Imagine two query points q_1 and q_2 with the property:

$$\text{for any distinct points } p_1, p_2 \in P\text{: } L_2(p_1, q_1) < L_2(p_2, q_1) \Leftrightarrow L_2(p_1, q_2) < L_2(p_2, q_2).$$

For any $p_{\text{start}} \in P$, the behavior of **greedy** invoked with parameters (p_{start}, q_1) is exactly the same as invoked with (p_{start}, q_2). This is true regardless of which proximity graph is adopted.

The points in P define $\binom{n}{2}$ perpendicular bisectors, which dissect $\mathbb{R}^d$ into $O(n^{2d})$ polytopes. Queries in each polytope have the same NN and induce the same behavior of **greedy**. Take a query representative from each polytope. Section 5.2 has shown that the merged graph G guarantees a low query time on one query with probability at least $1 - 1/n^{z-1}$, where z is the constant in (17). We can thus conclude that G guarantees a low query time on all the representatives (and hence all queries in $\mathbb{R}^d$) with probability at least $1 - n^{O(d)}/n^{z-1}$, which is greater than $1 - 1/n^c$ for any constant c by making z sufficiently large.

Only one issue remains. Currently, the number of edges in G is $O((1/\epsilon)^\lambda \cdot n)$ in expectation. To ensure this size bound w.h.p., we run our construction algorithm $z' \cdot \log n$ times for a sufficiently large constant z'. With probability at least $1 - z' \log n \cdot n^{O(d)}/n^{z-1}$, the proximity graphs

produced by all the runs guarantee query time $O((1/\epsilon)^\lambda \cdot \log^2 \Delta + (1/\epsilon)^{d-1} \log n \cdot \log^2 \Delta)$ on all queries in $\mathbb{R}^d$. By Markov's inequality, in each run, the graph size exceeds twice the expectation with probability at most $1/2$. Therefore, the probability for the smallest G of all runs to have $O((1/\epsilon)^\lambda \cdot n)$ edges is at least $1 - 1/n^{z'}$.

We now conclude that w.h.p. we can compute in $(1/\epsilon)^\lambda \cdot n \operatorname{polylog}(n\Delta)$ time a $(1+\epsilon)$-PG that has $O((1/\epsilon)^\lambda \cdot n)$ edges and ensures query time $O((1/\epsilon)^\lambda \cdot \log^2 \Delta + (1/\epsilon)^{d-1} \log n \cdot \log^2 \Delta)$ for all queries. This completes the proof of Theorem 1.3.

References

[1] Alexandr Andoni and Ilya P. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 793–801, 2015.

[2] Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *Journal of the ACM (JACM)*, 57(1):1:1–1:54, 2009.

[3] Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 271–280, 1993.

[4] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[5] Sunil Arya, David M. Mount, and Michiel H. M. Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry*, 13(2):91–107, 1999.

[6] Ilias Azizi, Karima Echihabi, and Themis Palpanas. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proceedings of the ACM on Management of Data (PACMMOD)*, 3(1), 2025.

[7] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. Roargraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. *Proceedings of the VLDB Endowment (PVLDB)*, 17(11):2735–2749, 2024.

[8] Kenneth L. Clarkson. An algorithm for approximate closest-point queries. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 160–164, 1994.

[9] Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 574–583, 2006.

[10] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 253–262, 2004.

[11] Haya Diwan, Jinrui Gou, Cameron Musco, Christopher Musco, and Torsten Suel. Navigable graphs for high-dimensional nearest neighbor search: Constructions and limits. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2024.

[12] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment (PVLDB)*, 12(5):461–474, 2019.

[13] Jianyang Gao and Cheng Long. High-dimensional approximate nearest neighbor search: With reliable and efficient distance comparison operations. *Proceedings of ACM Management of Data (SIGMOD)*, 1(2):1–27, 2023.

[14] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. Symphonyqg: Towards symphonious integration of quantization and graph for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data (PACMMOD)*, 3(1), 2025.

[15] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal of Computing*, 35(5):1148–1184, 2006.

[16] Ben Harwood and Tom Drummond. Fanng: Fast approximate nearest neighbour graphs. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5713–5722, 2016.

[17] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[18] Piotr Indyk and Haike Xu. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2023.

[19] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[20] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 798–807, 2004.

[21] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68, 2014.

[22] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 42(4):824–836, 2020.

[23] James Jie Pan, Jianguo Wang, and Guoliang Li. Survey of vector database management systems. *The VLDB Journal*, 33(5):1591–1615, 2024.

[24] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. Efficient approximate nearest neighbor search in multi-dimensional databases. *Proceedings of ACM Management of Data (SIGMOD)*, 1(1):1–27, 2023.

[25] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete euclidean graph. In *Proceedings of Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

[26] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. Starling: An i/o-efficient disk-resident graph index framework for high-dimensional vector similarity search on data segment. *Proceedings of the ACM on Management of Data (PACMMOD)*, 2(1):1–27, 2024.

[27] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment (PVLDB)*, 14(11):1964–1978, 2021.

[28] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal of Computing*, 11(4):721–736, 1982.

[29] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. *Proceedings of the VLDB Endowment (PVLDB)*, 16(8):1979–1991, 2023.

A Proof of Fact 2.1

The If-Direction ($\Leftarrow$). To prove this direction, we assume that G is $(1+\epsilon)$-navigable. Fix an arbitrary query point $q \in \mathcal{M}$ and an arbitrary data point $p_{\text{start}} \in P$. Let p be the point returned by the **greedy** algorithm when invoked with parameters (p_{start}, q). Suppose that p is not a $(1+\epsilon)$-ANN of q. Because G is $(1+\epsilon)$-navigable, p must have an out-neighbor closer to q than p itself, meaning that the greedy algorithm cannot terminate at p, giving a contradiction. Hence, p must be a $(1+\epsilon)$-ANN of q; and thus G is a $(1+\epsilon)$-PG.

The Only-If Direction ($\Rightarrow$). To prove this direction, we assume that G is a $(1+\epsilon)$-PG. Fix an arbitrary query point $q \in \mathcal{M}$ and an arbitrary data point $p \in P$ such that p is not a $(1+\epsilon)$-ANN of q. Run the **greedy** algorithm with $p_{\text{start}} = p$ and q. Because G is a $(1+\epsilon)$-PG, the algorithm must return a $(1+\epsilon)$-ANN of q and hence cannot return p_{start}. As a result, **greedy** must be able to identify an out-neighbor p_{out} of p_{start} with $D(p_{\text{out}}, q) < D(p_{\text{start}}, q)$. The presence of p_{out} indicates that G is $(1+\epsilon)$-navigable.

B Proof of Fact 2.3

Let d_{min} and d_{max} be the minimum and maximum inter-point distances in X, respectively. Hence, $A = d_{max}/d_{min}$. By definition of diameter, the set X can be covered by a ball $B(p, d_{max})$ where p can be any point in X. Inductively, suppose that $B(p, d_{max})$ can be covered by $2^{i \cdot \lambda}$ balls of radius $d_{max}/2^i$ for some $i \geq 0$. By definition of doubling dimension, we can cover each of those balls with 2^λ balls of radius $d_{max}/2^{i+1}$. This means that $B(p, d_{max})$ can be covered by $2^{(i+1)\cdot\lambda}$ balls of radius $d_{max}/2^{i+1}$.

The above argument tells us that $B(p, d_{max})$ and, hence, X can be covered by $2^{k\cdot\lambda}$ balls of radius $A \cdot d_{min}/2^k$ for any $k \geq 0$. Now, set $k = 2 + \lceil \log A \rceil$, with which we have

$$\frac{d_{max}}{2^k} = \frac{A \cdot d_{min}}{2^k} < \frac{d_{min}}{2}.$$

Therefore, X can be covered by

$$2^{k\lambda} \leq 2^{\lambda \cdot (3 + \log A)} = (2^{3+\log_2 A})^\lambda = (8A)^\lambda$$

balls of radius less than $d_{min}/2$. Each ball can cover at most one point in X. This is because the maximum distance of two points in a ball is less than d_{min}, which, let us recall, is the smallest inter-point distance in X. It thus follows that $|X| \leq (8A)^\lambda$, which is $O(A^\lambda)$ for $\lambda = O(1)$.

C Doubling Dimension of the Hard Input in Section 3

Given an arbitrary ball $B(p, r)$ where $p \in \mathcal{M}$ and $r > 0$, we will explain how to cover it with at most two balls of radius $r/2$. This indicates that the metric space $(\mathcal{M}, D)$ has doubling dimension 1.

Let us start by reminding the reader that, for two distinct points $v_1, v_2 \in \mathcal{M}$ (which are leaves of the binary tree $\mathcal{T}$), their distance is 2^ℓ, where $\ell \geq 1$ is the level of the LCA of v_1 and v_2. Therefore, if $r < 2$, then $B(p, r)$ contains only p itself and thus can be covered by a single ball of radius $r/2$. The subsequent discussion assumes $r \geq 2$.

Let $r' = 2^\ell$ be the largest power of 2 within the range $[2, 2\Delta]$ that does not exceed r. We have $B(p, r') = B(p, r)$ because every inter-point distance in $\mathcal{M}$ is a power of 2, as mentioned. Note that ℓ is some integer between 1 and $h = \log(2\Delta)$. Thus, point p, which is a leaf of $\mathcal{T}$, has an ancestor at level ℓ, which we denote as u_{anc}. The ball $B(p, r')$ is precisely the set of leaves in the subtree of u_{anc}.

Denote by u_1 and u_2 the left and right children of u_{anc}, respectively. Let X_1 (resp., X_2) be the set of leaves in the subtree of u_1 (resp., u_2). Clearly, $B(p, r') = X_1 \cup X_2$. Each of X_1 and X_2 is covered by a ball of radius $r'/2 \leq r/2$. By symmetry, it suffices to prove this only for X_1. Take an arbitrary leaf v from X_1. We argue that $X_1 \subseteq B(v, r'/2)$. Indeed, for any $v' \in X_1$ that differs from v, the LCA of v and v' must be a descendant of u_1 and hence must be at an level at most $\ell - 1$, meaning that $D(v, v') \leq 2^{\ell-1} = r'/2$.

D Proof of Lemma 4.1

First, let us note the following property of our design: if two points $p_1, p_2 \in P$ are from different blocks, then $D_{p^*}(p_1, p_2) = L_\infty(p_1, p_2) = |p_1[1] - p_2[1]| \geq s + 1$.

Triangle Inequality. To prove $(\mathcal{M}, D_{p^*})$ is a metric space, it suffices to prove that D_{p^*} satisfies the triangle inequality. Consider any p_1, p_2, and $p_3 \in \mathcal{M}$. If all of them originate from P, then their distances under D_{p^*} are the same as under L_∞-norm. Hence, we must have $D_{p^*}(p_1, p_2) \leq D_{p^*}(p_1, p_3) + D_{p^*}(p_2, p_3)$. Next, we will assume that $p_3 = q$. Furthermore, we assume $p_1 \neq p_2$ because otherwise $D_{p^*}(p_1, p_2) = 0$ and the triangle inequality holds on $D_{p^*}(p_1, p_2)$, $D_{p^*}(p_1, q)$, and $D_{p^*}(p_2, q)$.

If neither p_1 nor p_2 is from M_{w^*} (i.e., the block of p^*), then the mutual distances of p_1, p_2, and q under D_{p^*} are the same as those of p_1, p_2, and w^* under L_∞. Those mutual distances must satisfy the triangle inequality.

Consider now the case where both p_1 and p_2 are from M_{w^*}. W.l.o.g., suppose that $D_{p^*}(p_1, q) \leq D_{p^*}(p_2, q)$. Thus, $D_{p^*}(p_1, p_2) \in [1, s-1]$, $D_{p^*}(p_1, q) = s - 1$ or s, while $D_{p^*}(p_2, q) = s$. The three distances obey the triangle inequality.

It remains to examine the case where $p_1 \in M_{w^*}$ but $p_2 \notin M_{w^*}$. We must have $D_{p^*}(p_1, q) = s - 1$ or s, $D_{p^*}(p_1, p_2) > s$, and $D_{p^*}(p_2, q) = L_\infty(p_2, w^*) > s$. Let us first derive

$$\begin{aligned}
D_{p^*}(p_1, q) + D_{p^*}(p_1, p_2) &\geq s - 1 + L_\infty(p_1, p_2) \\
&= s - 1 + |p_1[1] - p_2[1]| \\
(\text{as } p_1 \in M_{w^*},\ w^* \in M_{w^*}) &\geq |p_1[1] - w^*[1]| + |p_1[1] - p_2[1]| \\
&\geq |p_2[1] - w^*[1]| \\
&= D_{p^*}(p_2, q).
\end{aligned}$$

Similarly, we can derive

$$\begin{aligned}
D_{p^*}(p_1, q) + D_{p^*}(p_2, q) &\geq s - 1 + L_\infty(p_2, w^*) \\
&= s - 1 + |p_2[1] - w^*[1]| \\
&\geq |p_1[1] - w^*[1]| + |p_2[1] - w^*[1]| \\
&\geq |p_1[1] - p_2[1]|
\end{aligned}$$

$$= \quad D_{p^*}(p_1, p_2).$$

We now conclude that $D_{p^*}(p_1, q)$, $D_{p^*}(p_1, p_2)$, and $D_{p^*}(p_2, q)$ satisfy the triangle inequality.

Doubling Dimension. We will speak about balls under two different metric spaces: $(\mathcal{M}, D_{p^*})$ and (P, L_∞). To avoid confusion, we will adopt the notations below:

- given a point $p \in \mathcal{M}$, let $B_{p^*}(p, r)$ be the ball $B(p, r)$ under $(\mathcal{M}, D_{p^*})$;
- given a point $p \in P$, let $B_\infty(p, r)$ be the ball $B(p, r)$ under (P, L_∞).

When p comes from P, we will refer to $B_{p^*}(p, r)$ as the *D_{p^*}-corresponding ball* of $B_\infty(p, r)$. These two balls have the following relationship:

- $B_\infty(p, r) \subseteq B_{p^*}(p, r)$;
- if $B_\infty(p, r) \neq B_{p^*}(p, r)$, then $B_{p^*}(p, r)$ contains only one extra point — namely, q — outside of $B_\infty(p, r)$.

The metric space $(\mathbb{R}^d, L_\infty)$ is known to have doubling dimension d. As $P \subseteq \mathbb{R}^d$, the metric space (P, L_∞) has doubling dimension at most d. We will utilize this fact to analyze the doubling dimension λ of $(\mathcal{M}, D_{p^*})$. Given an arbitrary ball $B_{p^*}(p, r)$, we will show how to cover $B_{p^*}(p, r)$ with at most $1 + 2^d$ balls of radius $r/2$ under $(\mathcal{M}, D_{p^*})$, meaning that $\lambda \leq \log(1 + 2^d)$.

Consider first $p \in P$ (in other words, $p \neq q$). If $B_{p^*}(p, r) = B_\infty(p, r)$, we cover $B_{p^*}(p, r)$ with a set S of at most 2^d balls under $(\mathcal{M}, D_{p^*})$ found using the procedure below:

- Initialize S to be the empty set.
- Cover $B_\infty(p, r)$ with at most 2^d balls of radius $r/2$ under (P, L_∞).
- For each of the above ball, add its D_{p^*}-corresponding ball to S.

If $B_{p^*}(p, r) \neq B_\infty(p, r)$, we cover $B_{p^*}(p, r)$ with a set S of at most $1 + 2^d$ balls under $(\mathcal{M}, D_{p^*})$ as follows:

- Obtain a set S using the procedure for the case $B_{p^*}(p, r) = B_\infty(p, r)$.
- Add to S the ball $B_{p^*}(q, r/2)$.

The subsequent discussion will focus on the scenario where $p = q$:

- If $r < s - 1$, then $B_{p^*}(q, r) = \{q\}$ and, hence, can be covered with a single ball of radius $r/2$ under $(\mathcal{M}, D_{p^*})$.
- If $r = s - 1$, then $B_{p^*}(q, r) = \{q, p^*\}$ and, hence, can be covered with two balls of radius $r/2$ under $(\mathcal{M}, D_{p^*})$.
- If $r \geq s$, then $B_{p^*}(q, r) = B_{p^*}(w^*, r) = \{q\} \cup B_\infty(w^*, r)$; recall that w^* is the point in W such that $p^* \in M_{w^*}$. As $w^* \in P$, we have already explained how to cover $B_{p^*}(w^*, r)$ with at most $1 + 2^d$ balls under $(\mathcal{M}, D_{p^*})$. The same approach therefore works for $B_{p^*}(q, r)$.

We now complete the proof of Lemma 4.1.

E Proof of Lemma 5.1

E.1 Basic Facts

Let us start with three facts that will be useful in our technical derivation.

Fact E.1. For any $0 \leq x \leq 1/2$, we have $\tan x \leq 2x$.

Proof. Define $f(x) = \tan x - 2x$. Thus, $f'(x) = 1/(\cos x)^2 - 2$, which is negative for $0 \leq x \leq 1/2$. The fact then follows from $f(0) = 0$. □

Given two points p, q, we will use the following notation frequently:

$$\rho_{u,v} = \text{the ray that emanates from } p \text{ and passes } q. \tag{19}$$

Fact E.2. Let a, b, and c be three distinct points in $\mathbb{R}^d$ such that the angle γ between rays $\rho_{a,b}$ and $\rho_{a,c}$ satisfies $0 < \gamma < \pi/2$. If $L_2(a,b) = L_2(a,c) = l > 0$, then $L_2(b,c) < l \cdot \tan\gamma$.

Proof. By applying basic geometric reasoning to the isosceles triangle abc, we obtain $L_2(b,c) = 2 \cdot l \sin(\gamma/2)$. Next, we will prove $2\sin(\gamma/2) < \tan\gamma$.

As $0 < \gamma/2 < \pi/4$, we have

$$\begin{aligned} & (2\cos(\gamma/2) + 1)(\cos(\gamma/2) - 1) < 0 \\ \Rightarrow \quad & 2\cos^2(\gamma/2) - \cos(\gamma/2) - 1 < 0 \\ \Rightarrow \quad & \cos^2(\gamma/2) - \sin^2(\gamma/2) < \cos(\gamma/2) \end{aligned}$$

As $\cos^2(\gamma/2) > \sin^2(\gamma/2)$ when $0 < \gamma/2 < \pi/4$, we can derive from the above

$$\begin{aligned} & 1 < \frac{\cos(\gamma/2)}{\cos^2(\gamma/2) - \sin^2(\gamma/2)} \\ \Rightarrow \quad & 2 \cdot \sin(\gamma/2) < \frac{2\sin(\gamma/2)\cos(\gamma/2)}{\cos^2(\gamma/2) - \sin^2(\gamma/2)} = \frac{\sin\gamma}{\cos\gamma} \end{aligned}$$

which is $\tan\gamma$. □

Fact E.3. If $0 \leq \gamma \leq \epsilon/32$, then $(2+\epsilon) \cdot (2\tan\gamma + 1 - \cos\gamma) < \epsilon$.

Proof. Because $\gamma \leq \epsilon/32 \leq 1/32$, we have from Fact E.1:

$$\tan\gamma \leq 2\gamma \leq \epsilon/16. \tag{20}$$

Later, we will prove:

$$1 - \cos\gamma < \epsilon/6. \tag{21}$$

Hence

$$\begin{aligned} (2+\epsilon) \cdot (2\tan\gamma + 1 - \cos\gamma) &< (2+\epsilon) \cdot \left(2 \cdot \frac{\epsilon}{16} + \frac{\epsilon}{6}\right) \\ &= (2+\epsilon) \cdot \frac{7\epsilon}{24} \\ (\text{as } 0 < \epsilon \leq 1) &\leq 3 \cdot \frac{7\epsilon}{24} \end{aligned}$$

which is less than ϵ, as claimed.

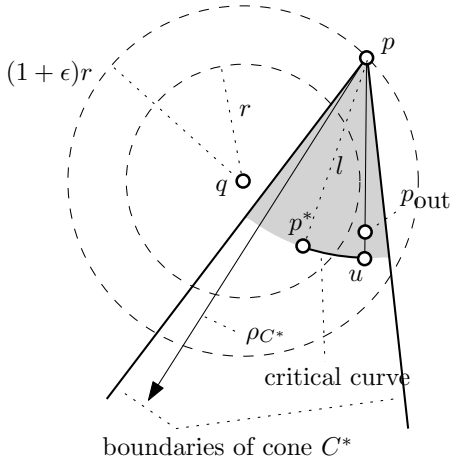


Figure 4: Case 1 of the proof in Section E

It remains to explain why (21) is correct. First, using (20), we can derive:

$$\cos^2\gamma = \frac{1}{1+\tan^2\gamma} \geq \frac{16^2}{16^2+\epsilon^2}. \tag{22}$$

Define $f(x) = x^3 - 12x^2 + 292x - 3072$. We have $f'(x) = 3x^2 - 24x + 292$, which is always positive. As $f(1) < 0$, we can assert that $f(x) < 0$ for all $x \leq 1$. This yields:

$$\begin{aligned} \epsilon^3 - 12\epsilon^2 + 292\epsilon &< 3072 \\ \Rightarrow \epsilon^4 - 12\epsilon^3 + 292\epsilon^2 &< 3072\epsilon \end{aligned}$$

Rearranging terms from the above gives

$$\begin{aligned} & & \frac{16^2}{16^2+\epsilon^2} &> (1-\epsilon/6)^2 \\ &\text{(by (22))} &\Rightarrow\ \cos^2\gamma &> (1-\epsilon/6)^2 \\ &\text{(as } \cos\gamma > 0 \text{ and } \epsilon < 1\text{)} &\Rightarrow\ \cos\gamma &> 1-\epsilon/6 \end{aligned}$$

thus giving the claim in (21). □

E.2 The Proof

Let G be an $(\epsilon/32)$-graph of P. Our objective is to prove that G is a $(1+\epsilon)$-PG of P. Fix an arbitrary data point $p \in P$ and an arbitrary query point $q \in \mathbb{R}^d$ such that p is not a $(1+\epsilon)$-ANN of q. We will show that p has an out-neighbor p_{out} in G satisfying $L_2(p_{\text{out}}, q) < L_2(p, q)$. This indicates that G is $(1+\epsilon)$-navigable and thus a $(1+\epsilon)$-PG of P by Fact 2.1.

Let us introduce two notions related to balls. Given a ball $B(p, r)$, we define its *surface* as the set $\{x \in \mathbb{R}^d \mid L_2(p, x) = r\}$. In addition, we say that a point $x \in \mathbb{R}^d$ is

- *in the interior* of $B(p, r)$ if $L_2(p, x) < r$;
- *on the surface* of $B(p, r)$ if $L_2(p, x) = r$.

Let p^* be the (exact) NN of q. Henceforth, we will fix

$$r = \frac{L_2(p, q)}{1+\epsilon}. \tag{23}$$

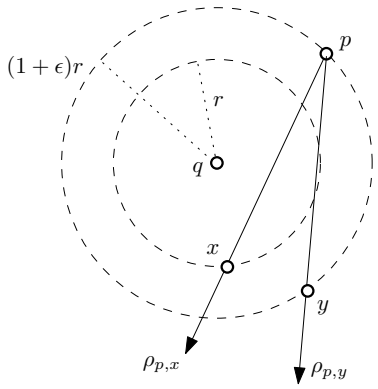


Figure 5: Illustration of Lemma E.1

As p is not a $(1+\epsilon)$-ANN of q, we must have $L_2(q,p^*) < r$, i.e., p^* is in the interior of $B(q,r)$.

Recall from Section 5.1 that the union of the cones in $\mathscr{C}_p$ is $\mathbb{R}^d$ (see (18) for the definition of $\mathscr{C}_p$). Hence, there must be a cone $C^* \in \mathscr{C}_p$ covering p^*. Define

$$p_{\text{out}} \quad = \quad \text{the nearest-point-on-ray of } p \text{ in cone } C^*. \tag{24}$$

As explained in Section 5.1, p_{out} is the point whose projection on ρ_{C^*} — the designated ray of C^* — is the closest to p under L_2 norm among the projections of all the points of $P \setminus \{p\}$ covered by C^*; see Figure 4 for an illustration. By the definition of θ-graph, p_{out} is an out-neighbor of p in G.

The rest of the proof will show

$$L_2(p_{\text{out}}, q) < (1+\epsilon)r \tag{25}$$

which, by the value of r in (23), says $L_2(p_{\text{out}}, q) < L_2(p,q)$, thus leading us to the conclusion that G is $(1+\epsilon)$-navigable. Define

$$l \quad = \quad L_2(p, p^*) \tag{26}$$

Next, we proceed differently depending on the relationship between $L_2(p, p_{\text{out}})$ and l.

Case 1: $L_2(p, p_{\text{out}}) \leq l$. That is, p_{out} is in $B(p,l)$. This is the scenario illustrated in Figure 4. Before proceeding, the reader may wish to review the definition in (19) first.

Lemma E.1. *Let x and y be two points that are on the surfaces of $B(q,r)$ and $B(q,(1+\epsilon)r)$, respectively. If $L_2(p,x) = L_2(p,y)$, then the angle between the rays $\rho_{p,x}$ and $\rho_{p,y}$ is strictly larger than $\epsilon/8$.*

See Figure 5 for an illustration.

Proof of Lemma E.1. Denote by γ the angle between $\rho_{p,x}$ and $\rho_{p,y}$. It must hold that $\gamma > 0$. Indeed, if $\gamma = 0$, then x and y are on the same line, in which case the condition $L_2(p,x) = L_2(p,y)$ implies $x = y$. This contradicts the fact that x and y are on the surfaces of two different balls.

Assume, for contradiction, that $\gamma \leq \epsilon/8 \leq 1/8$. As $\gamma < \pi/2$, we can use Fact E.2 to derive

$$\begin{aligned} L_2(x,y) &< L_2(p,x) \cdot \tan\gamma \\ \text{(by triangle inequality)} &\leq (L_2(p,q) + L_2(q,x)) \cdot \tan\gamma \\ &= ((1+\epsilon)r + r) \cdot \tan\gamma \end{aligned}$$

$$\begin{aligned}
(\text{by Fact E.1 and } 0 < \gamma \le \epsilon/8) &\le ((1+\epsilon)r + r) \cdot (2\gamma) \\
&\le (2+\epsilon)r \cdot (\epsilon/4) \\
(\text{as } 0 < \epsilon \le 1) &< 3\epsilon r/4 \\
&< \epsilon r.
\end{aligned}$$

On the other hand, by the triangle inequality, we have $L_2(x,y) \ge L_2(q,y) - L_2(q,x) = \epsilon r$, thus giving a contradiction. □

In general, given two different points $p_1, p_2 \in P$, we use the term "segment p_1p_2" to refer to the line segment connecting them. Define

$$u = \text{the intersection point between the ray } \rho_{p,p_{\text{out}}} \text{ and the surface of } B(p,l).$$

Because (as mentioned) p_{out} is in $B(p,l)$, the point p_{out} must be on the segment pu; see Figure 4.

We argue that u must be in the interior of $B(q,(1+\epsilon)r)$. Once this is done, we know that the entire segment connecting p and u — except point p — must be in the interior of $B(q,(1+\epsilon)r)$. Thus, p_{out}, which is different from p, must be in the interior of $B(q,(1+\epsilon)r)$, which indicates $L_2(p_{\text{out}},q) < (1+\epsilon)r$, as claimed in (25).

As both p^* and u are on the surface of $B(p,l)$, we must be able to travel on the surface of $B(p,l)$ from p^* to u. We will do so on a particular curve — referred to as the *critical curve* — decided as follows:

- For each point p_{seg} on segment p^*u, shoot a ray from p towards p_{seg}, and take the point p_{curve} at which the ray intersects the surface of $B(p,l)$.
- The critical curve is the set of all the p_{curve} points produced.

See Figure 4 for an illustration of the critical curve. As both p^* and u are in cone C^*, the angle between the rays ρ_{p,p^*} and $\rho_{p,u}$ is at most $\epsilon/32$ (because G is an $(\epsilon/32)$-graph of P). For any points x, y on the critical curve, the angle of the rays $\rho_{p,x}$ and $\rho_{p,y}$ can only be smaller and hence is at most $\epsilon/32$.

Assume, for contradiction, that u is not in the interior of $B(q,(1+\epsilon)r)$. Remember that p^* is in the interior of $B(q,r)$. As we walk from p^* towards u on the critical curve, we must first hit the surface of $B(q,r)$ at some point x and then hit the surface of $B(q,(1+\epsilon)r)$ at another point y. That both x and y are on the curve means that they are both on the surface of $B(p,l)$ and, hence, $L_2(p,x) = L_2(p,y)$. By Lemma E.1, the angle between the rays $\rho_{p,x}$ and $\rho_{p,y}$ is larger than $\epsilon/8$. This is impossible because as mentioned the angle can be at most $\epsilon/32$.

Case 2: $L_2(p, p_{\text{out}}) > l$. That is, p_{out} is outside $B(p,l)$, as illustrated in Figure 6. We will prove later

$$L_2(p^*, p_{\text{out}}) < \epsilon r \tag{27}$$

where p_{out} is defined in (24). The above will give us

$$L_2(p_{\text{out}}, q) \le L_2(q,p^*) + L_2(p^*, p_{\text{out}}) < r + \epsilon r = (1+\epsilon)r$$

as claimed in (25).

In the rest of our proof, we will fix

$$\gamma = \text{the angular diameter of } C^*.$$

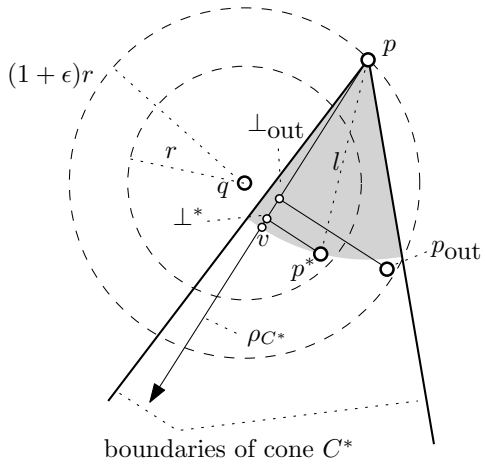


Figure 6: Case 2 of the proof in Section E

Hence, $\gamma \leq \epsilon/32$ (because G is an $(\epsilon/32)$-graph). Define:

$$\begin{aligned} v &= \text{the intersection point between the ray } \rho_{C^*} \text{ and the surface of } B(p,l) \\ \perp^* &= \text{the projection point of } p^* \text{ onto ray } \rho_{C^*} \\ \perp_{\text{out}} &= \text{the projection point of } p_{\text{out}} \text{ onto ray } \rho_{C^*} \end{aligned}$$

See Figure 6 for an illustration.

Lemma E.2. *The following inequalities are correct:*

$$l < (2+\epsilon) \cdot r \tag{28}$$
$$L_2(p^*, v) \leq l \cdot \tan\gamma \tag{29}$$
$$L_2(v, \perp_{out}) < l \cdot (1 - \cos\gamma) \tag{30}$$
$$L_2(\perp_{out}, p_{out}) \leq l \cdot \tan\gamma. \tag{31}$$

Proof. We have

$$l = L_2(p,p^*) \leq L_2(p,q) + L_2(q,p^*) < (1+\epsilon)r + r = (2+\epsilon)\cdot r$$

which proves (28).

Next, let us attend to (29). As both v and p^* are on the surface of $B(p,l)$, it holds that $L_2(p,v) = L_2(p,p^*) = l$. Define γ' to be the angle between rays $\rho_{p,v} = \rho_{C^*}$ and ρ_{p,p^*}. If $\gamma' = 0$, then p^* coincides with v; thus, $L_2(p^*,v) = 0$ and (29) holds trivially. If $\gamma' > 0$, we must have $\gamma' \leq \gamma \leq \epsilon/32$ by definition of γ. This allows us to apply Fact E.2, which gives:

$$\begin{aligned} L_2(p^*,v) &< L_2(p,v)\cdot\tan\gamma' \\ (\text{as } \gamma' \leq \gamma \leq \epsilon/32 < \pi/2) &\leq l\cdot\tan\gamma \end{aligned}$$

as claimed in (29).

Define γ'' to be the angle between rays $\rho_{p,v} = \rho_{C^*}$ and $\rho_{p,p_{\text{out}}}$. We must have $\gamma'' \leq \gamma$ by definition of γ. Thus:

$$\begin{aligned} L_2(p,\perp_{\text{out}}) &= L_2(p,p_{\text{out}})\cdot\cos\gamma'' \\ (\text{as } \gamma'' \leq \gamma \leq \epsilon/32 < \pi/2) &\geq L_2(p,p_{\text{out}})\cdot\cos\gamma \\ (\text{as } p_{\text{out}} \text{ is outside } B(p,l)) &> l\cdot\cos\gamma. \end{aligned} \tag{32}$$

As p^* is on the surface of $B(p,l)$ and the angle between rays $\rho_{p,v}$ and ρ_{p,p^*} is at most $\gamma < \pi/2$, the projection $\perp^*$ of p^* on ρ_{C^*} must be in the interior of $B(p,l)$. Hence, p^* must be on the segment pv. On the other hand, by the definition of p_{out} (see (24)), its projection $\perp_{\text{out}}$ on ρ_{C^*} cannot be farther from p than $\perp^*$. This means that $\perp_{\text{out}}$ must be on the segment connecting p and $\perp^*$. See Figure 6. Therefore:

$$\begin{aligned} L_2(v, \perp_{\text{out}}) &= L_2(p,v) - L_2(p, \perp_{\text{out}}) \\ \text{(by (32))} &< l - l \cdot \cos\gamma \end{aligned}$$

which proves (30).

As mentioned, $\perp_{\text{out}}$ is on the segment connecting p and $\perp^*$, while $\perp^*$ is on the segment connecting p and v. This means that $\perp_{\text{out}}$ must be on segment pv, suggesting $L_2(p, \perp_{\text{out}}) \leq l$. Hence:

$$L_2(\perp_{\text{out}}, p_{\text{out}}) = L_2(p, \perp_{\text{out}}) \cdot \tan\gamma'' \leq l \cdot \tan\gamma'' \leq l \cdot \tan\gamma \tag{33}$$

which proves (31). □

Consequently, we have

$$\begin{aligned} L_2(p^*, p_{\text{out}}) &\leq L_2(p^*, v) + L_2(v, p_{\text{out}}) \\ &\leq L_2(p^*, v) + L_2(v, \perp_{\text{out}}) + L_2(\perp_{\text{out}}, p_{\text{out}}) \\ \text{(by (29), (30), (31))} &< l \cdot \tan\gamma + l \cdot (1 - \cos\gamma) + l \cdot \tan\gamma \\ &= l \cdot (2\tan\gamma + 1 - \cos\gamma) \\ \text{(by (28))} &< (2+\epsilon)(2\tan\gamma + 1 - \cos\gamma) \cdot r \\ \text{(by Fact E.3)} &< \epsilon r \end{aligned}$$

as needed in (27).