Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases

YUN PENG, Institute of AI and BlockChain in Guangzhou University, Department of Computer Science in Hong Kong Baptist University, China
BYRON CHOI*, Department of Computer Science in Hong Kong Baptist University, China
TSZ NAM CHAN, Department of Computer Science in Hong Kong Baptist University, China
JIANYE YANG*, Cyberspace Institute of Advanced Technology in Guangzhou University, China
JIANLIANG XU, Department of Computer Science in Hong Kong Baptist University, China

Approximate nearest neighbor (ANN) search is a fundamental search in multi-dimensional databases, which has numerous real-world applications, such as image retrieval, recommendation, entity resolution, and sequence matching. Proximity graph (PG) has been the state-of-the-art index for ANN search. However, the search on existing PGs either suffers from a high time complexity or has no performance guarantee on the search result. In this paper, we propose a novel *τ-monotonic graph* (τ-MG) to address the limitations. The novelty of τ-MG lies in a τ-monotonic property. Based on this property, we prove that *if the distance between a query q and its nearest neighbor is less than a constant τ, the search on τ-MG guarantees to find the exact nearest neighbor of q and the time complexity of the search is smaller than all existing PG-based methods.* For index construction efficiency, we propose an approximate variant of τ-MG, namely τ-monotonic neighborhood graph (τ-MNG), which only requires the neighborhood of each node to be τ-monotonic. We further propose an optimization to reduce the number of distance computations in search. Our extensive experiments show that our techniques outperform all existing methods on well-known real-world datasets.

CCS Concepts: • **Information systems** → **Nearest-neighbor search**; **Query optimization**.

Additional Key Words and Phrases: Proximity graph; ANN search; τ-monotonic; Edge occlusion rule

ACM Reference Format:
Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1, Article 54 (May 2023), 27 pages. https://doi.org/10.1145/3588908

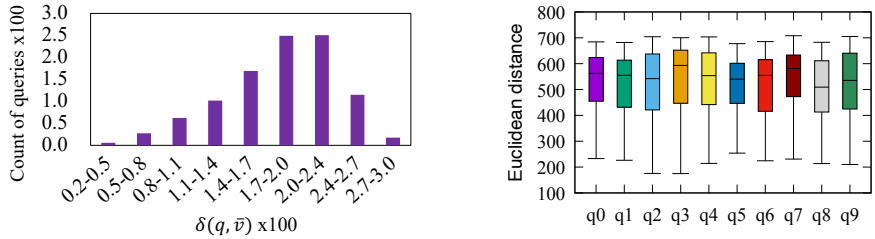
1 INTRODUCTION

Approximate nearest neighbor (ANN) search in multi-dimensional databases is a fundamental search and has many applications, such as image retrieval [28, 44], recommendation [12], entity resolution [21], and sequence matching [10]. Many ANN search methods have been proposed, such as the tree-based methods [33, 41, 52], quantization-based methods [25, 38, 55], hashing-based methods

*Byron Choi and Jianye Yang are the corresponding authors of this paper.

Authors' addresses: Yun Peng, yunpeng@gzhu.edu.cn, Institute of AI and BlockChain in Guangzhou University, Department of Computer Science in Hong Kong Baptist University, China; Byron Choi, Department of Computer Science in Hong Kong Baptist University, China, bchoi@comp.hkbu.edu.hk; Tsz Nam Chan, Department of Computer Science in Hong Kong Baptist University, China, edisonchan@comp.hkbu.edu.hk; Jianye Yang, Cyberspace Institute of Advanced Technology in Guangzhou University, China, jyyang@gzhu.edu.cn; Jianliang Xu, Department of Computer Science in Hong Kong Baptist University, China, xujl@comp.hkbu.edu.hk.

2836-6573/2023/5-ART54 $15.00
https://doi.org/10.1145/3588908



(a) (b)

Fig. 1. Distance distribution on the SIFT dataset: (a) Histogram of the distances between 1,000 queries and their nearest neighbors; (b) Distribution of the distances between 10 queries and all data points in the SIFT dataset

[34, 49, 60], and proximity graph-based methods [17, 31, 36]. Recent studies [6, 31, 32, 36, 40, 48, 53] show that *proxmity graph (PG)-based methods provide superior performance over other methods* in many large-scale applications of ANN search.

Given a database D with n points in an m-dimensional space, the PG-based methods construct a proximity graph G to index D, where each node in G corresponds to a point in D and two nodes have an edge if they satisfy some proximity property. For a query point q, a greedy routing on G can be used to find the ANN of q. Specifically, at each routing step, we compute the distances between q and the neighbors of the current node. Then, we select the neighbor that is the closest to q as the next current node and proceed to the next routing step. The routing stops if no neighbor of the current node is closer to q than itself.

Existing PG-based methods mainly lie in either of two following extremes. Let $\bar{v}$ denote the nearest neighbor of q and $\delta(q, \bar{v})$ denote the distance of $\bar{v}$ to q. At one extreme, some research studies (*e.g.*, [13, 17, 20]) impose the assumption $\delta(q, \bar{v}) = 0$ (*i.e.*, q is a point in D). For example, MRNG [17] guarantees to find $\bar{v}$ by the greedy routing and the expected time complexity is $O(n^{2/m} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$.[1] However, the assumption $\delta(q, \bar{v}) = 0$ is not always practical as $q \in D$ may not always hold. At the other extreme, many works (*e.g.*, [16, 29, 32, 36]) study simply the setting $\delta(q, \bar{v}) < \infty$. However, these works either cannot provide an error guarantee of the greedy routing (*e.g.*, SSG [16], HNSW [36], DPG [32]) or take $O(n)$ time to retrieve the nearest neighbor $\bar{v}$ (*e.g.*, DG [29]).

In this paper, we study a practical setting $\delta(q, \bar{v}) < \tau$, which falls between the two extremes, where τ is a user-defined constant. It is motivated by the observation that in real-world benchmark datasets, the queries are usually not in D but close to their nearest neighbors in D. For example, in our preliminary experiments on the SIFT dataset with 1 million data points, we observe that $\delta(q, \bar{v})$ is not zero but small when compared with the distances to all points in the dataset. Fig. 1(a) shows the histogram of $\delta(q, \bar{v})$ for 1,000 randomly selected queries. We can see that $20 < \delta(q, \bar{v}) < 270$ for most queries. Fig. 1(b) shows the box plot of the distances between ten randomly selected queries and all the points in SIFT. We can see that q is *much closer* to its nearest neighbor than the other points in D. The only existing work that has considered the setting $\delta(q, \bar{v}) < \tau$ is FANNG [20]. FANNG guarantees to find $\bar{v}$ by the greedy routing if $\delta(q, \bar{v}) < \tau$. However, we prove that the greedy routing of FANNG has a high time complexity. This paper proposes a solution, which has a lower time complexity, to find the nearest neighbor $\bar{v}$ of q when q satisfies $\delta(q, \bar{v}) < \tau$.

[1] The original expected time complexity proved in [17] is $O(\frac{1}{\Delta} n^{1/m} \ln n)$, where Δ is the smallest distance between any two points in D. In this paper, we prove that $\Delta \leq O((1/n)^{1/m})$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$.

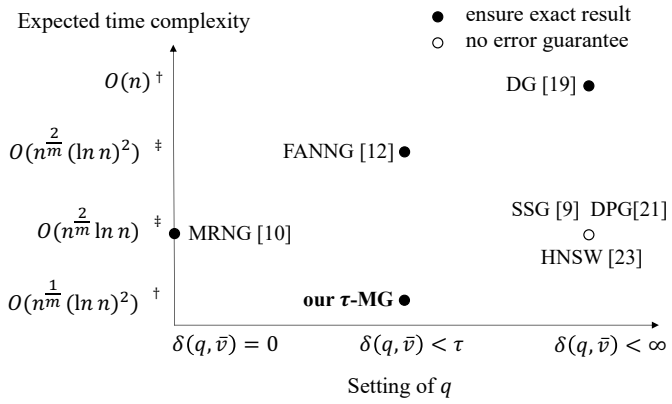


Fig. 2. A comparison of τ-MG with existing methods ($\bar{v}$ denotes the nearest neighbor of the query q, δ denotes the Euclidean distance, τ is a constant, n is the number of points in the database, m is the dimensionality, † means with probability 1.0, and ‡ means with probability at least $1-(1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$)

We analyze the greedy routing and find the key reason for the high time complexity of FANNG is that each routing step gets closer to q by only Δ, where Δ denotes the smallest distance between any two points in D. We show that $\Delta \leq O((1/n)^{1/m})$ with probability at least $1-(1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$. We further prove that the length of the greedy routing on any PG is $O(n^{2/m}\ln n)$ with probability at least $1-(1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$. This is the first work that can bound the length of the greedy routing in any PG.

Based on the analysis, we propose a novel proximity graph, namely *τ-monotonic graph* (τ-MG). The core of τ-MG is a new edge occlusion rule: for two nodes u and v in τ-MG, if u has a neighbor in the intersection of the ball centered at u with radius $\delta(u,v)$ and the ball centered at v with radius $\delta(u,v)-3\tau$, the edge (u,v) is not in the τ-MG. This edge occlusion rule makes τ-MG has a τ-monotonic property: if $\delta(q,\bar{v})<\tau$, τ-MG has a path from any node to $\bar{v}$, in which each step monotonically gets closer to q by *at least* τ. This property ensures that $\bar{v}$ can be found by the greedy routing. We further prove that the expected time complexity of the greedy routing is $O(n^{1/m}(\ln n)^2)$. Fig. 2 shows a comparison of τ-MG with existing methods. To the best of our knowledge, *the ANN search method of this paper has the smallest time complexity compared with all existing PG-based ANN search methods.*

To reduce the time complexity of PG construction, we propose an approximate variant of τ-MG, called *τ-monotonic neighborhood graph* (τ-MNG). The main idea is that we only require the subgraph induced by the neighborhood of each node is τ-monotonic. The idea of τ-monotonic neighborhood is generic and τ-MNG can be constructed from any PG. The time complexity of τ-MNG construction is $O(nh^2\ln h)$, where h is the size of the neighborhood and h is much smaller than n. Since the greedy routing on τ-MNG may get stuck in local optima, which reduces search accuracy, a beam search is used on τ-MNG for ANN search in order to strike a balance between search accuracy and efficiency. We prove that the probability that the beam search finds $\bar{v}$ is no less than the probability that the search enters the neighborhood of $\bar{v}$.

To optimize the beam search on τ-MNG, we propose a query-aware edge occlusion (QEO) method. The main idea is that if the current node u of the search is not in the top $p\%$ of the priority queue of the beam search, we use a lower bound δ_{lb} of the distance δ to order the neighbors of u and prune the tail neighbors. It is based on the observation that the farther the current node u away from q is, the higher chance the neighbors of u can be pruned. We also identify two important implementation details for query efficiency.

Contributions. The contributions of this paper are as follows.

- We find the key reason for the long routing of existing PGs is that each routing step gets closer to q by only Δ, where $\Delta = O((1/n)^{1/m})$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$;
- We propose a novel PG τ-MG, which guarantees to find the nearest neighbor $\bar{v}$ of q by the greedy search if $\delta(q, \bar{v}) < \tau$. The time complexity of the greedy search is $O(n^{1/m}(\ln n)^2)$, which is lower than all existing PG-based methods;
- We propose τ-MNG as an approximate variant of τ-MG. τ-MNG can be efficiently constructed in $O(h^2 \ln h)$ time, where h is the neighborhood size, and a beam search on τ-MNG has a high chance of returning $\bar{v}$;
- We propose an optimization to reduce the number of distance computations in the beam search on τ-MNG; and
- Our extensive experiments verify that our proposed techniques outperform the state-of-the-art methods on real-world benchmark datasets.

Organizations. The rest of this paper is organized as follows. Section 2 discusses the related work. The preliminaries are presented in Section 3. Section 4 presents the analysis of the inefficiency of existing PGs. Section 5 presents τ-MG. τ-MNG is presented in Section 6. The experimental evaluation is presented in Section 7. Section 8 concludes this paper. For presentation clarity, we put the detailed proofs in Section 9.

2 RELATED WORK

In this section, we focus on the ANN works that are closely related to proximity graphs (PGs). PGs have been studied recently [13, 14, 16, 17, 19, 20, 31, 32, 35]. Most existing PGs are based on three fundamental graph models: the Delaunay graph, the navigable small world graph, and the relative neighborhood graph. They are briefly reviewed as follows.

Delaunay graph (DG) is the dual graph of the Voronoi diagram [7]. For any query q in the m-dimensional Euclidean space E^m, DG guarantees to find the nearest neighbor of q by a greedy routing [36]. However, when m is large, DG becomes a complete graph [20], which makes the routing time-consuming. To reduce the node degree of DG, k-nearest neighbor graph (kNNG) is proposed as an approximation of DG, where each node is connected to its top-k nearest neighbors. For example, Jin et al. [26] and Hajebi et al. [19] propose IEH and GNNS using kNNG for ANN search, respectively. Since constructing kNNG is time-consuming, which takes $O(n^2)$ time, some research studies propose to construct the approximate kNNG. In particular, Dong et al. [14] propose a PG, namely KGraph to approximate kNNG. KGraph initializes the neighbors of each node randomly and then iteratively improves the neighbors of each node based on the principle that "a neighbor of a neighbor is likely a neighbor". Instead of a random initialization, EFANNA [15] first builds KD-trees on the database and uses ANN search on the KD-trees to initialize the neighbors of each node before executing NN-Descent. However, KGraph and EFANNA cannot ensure the connectivity of the constructed graph, which can significantly degrade the accuracy of the search results [17]. Recently, Wen et al. [32] propose DPG that diversifies the neighboring edges of each node. However, DPG neither reduces the time complexity nor provides an error guarantee for the search results.

Navigable small world graph (NSWG) has attracted much research attention since the well-known Milgram's social experiment [39]. Milgram observes that two nodes in a large graph are connected by a short path and the path can be found by a greedy routing. Many works are proposed to explain and analyze the performance of NSWG. For example, Watts and Strogatz [54] propose a 2-dimensional lattice model and prove the existence of a path of the length $O(\ln n)$ between two nodes in the lattice. Kleinberg [27] proves that the greedy routing cannot find the path. Kleinberg [27] proposes another 2-dimensional lattice model that guarantees to find the path of the length $O(\ln n)$ by the

greedy routing in $O((\ln n)^2)$ expected time. Martel and Nguyen [37] extend the work of Kleinberg [27] to support m-dimensional lattice. Inspired by Kleinberg's model, Malkov et al. [35] propose a PG (namely NSW) to support approximate ANN search in the m-dimensional Euclidean space. However, the node degree of NSW is high, which makes the routing costly. NSW does not ensure connectivity, which affects search accuracy. Recently, Malkov et al. [36] propose a hierarchical version of NSW (namely HNSW) to ensure connectivity and support routing in polylogarithmic time. However, the analysis of the time cost lacks rigorous theoretical support. Moreover, HNSW has no error guarantee on the search results.

Relative neighborhood graph (RNG) eliminates the longest edge in all possible triangles among the points in the database D, *i.e.*, if an edge (u, v) is in the graph, D has no point u' satisfying $\delta(u, u') < \delta(u, v)$ and $\delta(u', v) < \delta(u, v)$. RNG guarantees the average degree of each node is a small constant [23]. Later, Dearholt et al. [13] prove that RNG does not have sufficient edges to guarantee the accuracy of the search results of the greedy routing. Fu et al. [17] propose the monotonic relative neighborhood graph (MRNG). MRNG ensures that the average degree of each node is a constant and guarantees to find the nearest neighbor of q if $q \in D$. However, if $q \notin D$, MRNG has no error guarantee on the search results. FANNG [20] ensures to find the nearest neighbor $\bar{v}$ of q if $\delta(q, \bar{v}) < \tau$. However, FANNG does not provide theoretical analysis on node degree and time complexity of the greedy routing. Recently, Fu et al. [16] extend MRNG to a satellite system graph (SSG). Although SSG is designed for any $q \notin D$, SSG has no error guarantee on the search results of the greedy routing.

Several works study improving the routing algorithm on PG. For example, Muñoz et al. [50] propose to prune the neighbors that are not in the same quadrant as q at each routing step. Baranchuk et al. [9] use a graph neural network to select the neighbor to route to. Peng et al. [43] use neural networks to prune unpromissing neighbors to reduce the number of distance computations. Li et al. [31] propose a learning-based method to early stop the routing to avoid unnecessary routing steps. However, these works have no error guarantee on the search results. Zhao et al. [59] and Yu et al. [58] propose using GPUs to accelerate the routing on PG. However, this paper focuses on CPU, which is orthogonal to them.

Non-PG-based methods. There are also some ANN works not based on PGs, such as the tree-based methods [2, 30, 33, 41, 52], inverted index-based methods [8, 31], the quantization-based methods [18, 25, 38, 55], and the hashing-based methods [34, 49, 60]. Since recent studies [6, 31, 32, 36, 40, 48, 53] show that these methods are outperformed by PG-based methods, we do not include these methods in this section. We refer interested readers to excellent surveys (*e.g.*, [22, 32, 38, 48]) for more details.

There have been studies on the meaningfulness of ANN search [11, 30, 47]. With the increasing of dimensionality, the *contrast* (*i.e.*, the ratio of the distances between the query q and its nearest and farthest points) tends to 1 and ANN search becomes meaningless, as the NN of q could not be separated from other points. However, if the datasets have a low intrinsic dimensionality or the distance between the query and its nearest neighbor is no more than a constant, contrast exists and ANN search is meaningful [11].

3 PRELIMINARIES

In this section, we first present the problem setting and then present the proximity graph.

3.1 Problem setting

In this paper, we use E^m to denote the m-dimensional Euclidean space. The L_2 norm $\delta(u, v)$ of two points u and v is used to measure the distance between u and v. The approximate nearest neighbor (ANN) search is defined as follows.

Algorithm 1 ANN search on proximity graph (`beam_search`)

Input: PG G, query q, beam size b ▷ $b = 1$ means greedy routing
Output: ANN of q
1: initialize a priority queue $W = \emptyset$ to store candidates
2: v.explored is `false` by default for each node v in G
3: v_{init} = select an initial node in G
4: W.add($(\delta(v_{init}, q), v_{init})$) ▷ W is in ascending order of δ
5: **while** W has unexplored nodes **do**
6: u = the unexplored node with the smallest distance to q in W
7: **for** each neighbor v of u in G **do**
8: W.add($\delta(v, q), v$) into W
9: **end for**
10: u.explored = `true`
11: resize W to size b ▷ keep the b nodes with the smallest distances to q
12: **end while**
13: **return** the node in W that is the closest to q

ANN search problem. Given a database D with n data points in E^m, a query point q in E^m, and a small constant $\epsilon \geq 0$, we aim to efficiently find a point p in D such that $\delta(p, q) \leq (1 + \epsilon)\delta(\bar{v}, q)$, where $\bar{v}$ is the nearest neighbor of q in D.

For the convenience of modeling and evaluation, the exact value of ϵ is usually not used directly, and instead, other measures are used to measure search accuracy [16, 17, 31, 36, 43, 53]. The widely used accuracy measures for ANN search include the rank-based measures [16, 17, 53] and the distance-based measures [32, 42]. The ANN search problem can be naturally generalized to the approximate k-nearest neighbor (k-ANN) search problem [16, 17].

In this paper, we use $ball(u, r)$ to denote an open ball centered at u with radius r. We use $ball(r)$ if the ball center is not interested. We use $lune(u, v)$ to denote the intersection of two balls $ball(u, \delta(u, v))$ and $ball(v, \delta(u, v))$.

3.2 Proximity graph

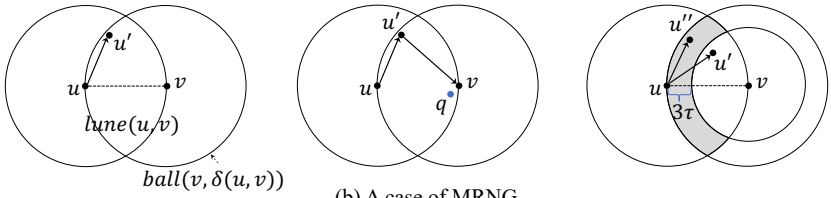
A proximity graph (PG) of a database D is a directed graph G, where the nodes in G are the points in D and two nodes have an edge if they satisfy some proximity property. For a node u of G, we use $N_G(u)$ to denote the set of outgoing neighbors of u in G and $N_G^-(u)$ to denote the nodes in G but not in $N_G(u)$. A path $P = [u_0, u_1, ..., u_{|P|-1}]$ in a PG is a *monotonic path for a query q* if each step gets closer to q, *i.e.*, $\delta(u_i, q) < \delta(u_{i-1}, q)$, for $i = 1, ..., |P| - 1$.

Algorithm 1 presents the search algorithm on a PG, which is a beam search and widely used in existing works (*e.g.*, [16, 17, 32, 35, 45, 51, 53, 59]). Greedy routing is a special case of Algorithm 1 when $b = 1$. The larger the beam size, the higher the search accuracy but the slower the search. Algorithm 1 can be easily extended to support k-ANN search by setting $b \geq k$ and returning the top-k best nodes in W in Line 13.

3.2.1 Monotonic relative neighborhood graph. The monotonic relative neighborhood graph (`MRNG`) [17] is a well-known PG. The core of `MRNG` is a rule of using shorter edges to occlude longer edges, defined as follows. (This edge occlusion rule is also used in other PG-based methods including `NSSG` [16], `HNSW` [36], and `FANNG` [20].)

Definition 1. *(Edge occlusion rule of* `MRNG`*) Given three nodes u, u', and v in G, if $(u, u') \in G$ and $u' \in lune(u, v)$, then $(u, v) \notin G$. Alternatively, we say (u, u') occludes (u, v).*

Fig. 3(a) illustrates that the edge (u, u') occludes the edge (u, v). Based on the edge occlusion rule, `MRNG` is defined as follows.



(a) MRNG (b) A case of MRNG missing the NN of q (c) τ-MG

Fig. 3. Illustration of the edge occlusion rules of MRNG and τ-MG. In (a) and (c), the edge (u, u') can occlude the edge (u, v). In (c), the edge (u, u'') cannot occlude (u, v). (b) shows a case where the greedy search misses the NN of q due to the edge occlusion rule of MRNG.

Definition 2. *(MRNG) Given a database D, a proximity graph G is an MRNG if G has an edge (u, u') occluding the edge (u, v) for any two nodes $u, v \in G$ satisfying $(u, v) \notin G$.*

MRNG has a performance guarantee as follows.

Lemma 1. *If $q \in D$, the greedy search on MRNG finds q starting from any node. However, if $q \notin D$, the greedy search on MRNG may not find the nearest neighbor of q.*

Fig. 3(b) shows an example where the greedy search on MRNG cannot find the NN of q. The edge (u, v) is occluded by (u, u'). Suppose that u is the current node of the greedy routing. Since $\delta(q, u) < \delta(q, u')$, the greedy routing will stop and return u. However, v is the NN of q.

We remark that although MRNG is designed for the Euclidean space [17], MRNG can still be used in general metric spaces. However, the analysis of MRNG for time and space complexities in [17] does not hold in general metric spaces.

4 ANALYSIS OF THE INEFFICIENCY OF EXISTING PROXIMITY GRAPHS

In this section, we analyze the reason for the inefficiency of existing PGs. The time cost of ANN search on a PG is mainly determined by two factors: the routing length (*i.e.*, the number of routing steps) and the node degree. We focus on the first factor in this section, as in many existing PGs, the node degree is bounded by a constant [16, 17, 36].[2]

Fu et al. [17] analyze the expected length of the greedy routing in the monotonic search network (MSNET). However, the analysis of Fu et al. has two limitations. First, MSNET is a special PG, which requires that for each node v, the PG has a monotonic path from any node to v. Second, q has to be a point in the database. In contrast, we analyze the expected length of the greedy routing in any PG and q can be any point in R^m. Our result is more general than that of [17].

Theorem 1. *Recall the same assumptions from [17] as below.*

- *Given a database D of n points, the points in D are uniformly distributed in a finite subspace of E^m and m is a constant.*
- *There exists a constant ψ, such that $\psi V_D \geq Vol(ball(R))$, where V_D denotes the volume of the minimum convex hull containing D, R denotes the maximum distance between two points in D, and $ball(R)$ denotes a ball with radius R.*

The point density g can increase with the growth of n and we assume that g is $O(\ln n)$.[3] We further assume that the query q has the same distribution as the points in D.

[2] MRNG [17] and SSG [16] prove their max node degrees are constants. HNSW [36] bounds the node degree by a predefined constant.

[3] It is a relaxation of the assumption in [17], where g is assumed to be a constant as n increasing.

For any PG G of D, the expected length of the greedy routing for q is $O(\frac{1}{\Delta} n^{\frac{1}{m}} \ln n^{\frac{1}{m}})$, where Δ denotes the smallest distance between any two points in D and $\Delta \leq O((1/n)^{1/m})$ with probability at least $1 - (\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$. Therefore, the expected length of the greedy routing is at least $O(n^{\frac{2}{m}} \ln n)$ with probability at least $1 - (\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$.

Proof Sketch. Let $[v_0, v_1, ..., v_x]$ be the path found by the greedy routing. First, we prove the expected length of the path $\mathbb{E}[x] = O(\frac{1}{\Delta} n^{\frac{1}{m}} \ln n^{\frac{1}{m}})$, which follows the framework of [17]. Second, we prove $\Delta \leq O(\sqrt{m}(m/n)^{1/m})$ with probability at least $1 - (\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$. The main idea is to prove that the probability that two points are both in a hypercube with volume $O(m/n)$ is at least $1 - (\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$. □

In Theorem 1, we can see that the routing is long on any existing PG. The reason is that each routing step gets closer to q by only Δ and $\Delta \to 0$ when $n \to \infty$.

5 τ-MONOTONIC GRAPH (τ-MG)

To address the limitation of exising PGs (cf. Section 4), this section proposes τ-monotonic graph (τ-MG). If $\delta(q, \bar{v}) < \tau$, each step of the greedy routing, except the last step, gets closer to q by a constant τ. Therefore, the expected length of the greedy routing is reduced to be $O(n^{\frac{1}{m}} \ln n)$. Moreover, if $\delta(q, \bar{v}) < \tau$, the greedy routing on τ-MG must find $\bar{v}$. τ-MG is designed based on a more tighter edge occlusion rule as follows.

Definition 3. *(Edge occlusion rule of τ-MG) Given three nodes u, u', and v in G, if $(u, u') \in G$ and u' is in the intersection of $ball(u, \delta(u, v))$ and $ball(v, \delta(u, v) - 3\tau)$, then $(u, v) \notin G$.*

Fig. 3(c) illustrates that the edge occlusion rule of τ-MG.

Definition 4. *(τ-MG) Given a constant $\tau \geq 0$, a τ-monotonic graph (τ-MG) is a directed proximity graph $G = (V, E)$, where for any two nodes u and v,*

- *if $\delta(u, v) \leq 3\tau$, $(u, v) \in G$; and*
- *if $\delta(u, v) > 3\tau$ and $(u, v) \notin G$, G has an edge (u, u') occluding the edge (u, v).*

We propose a new concept of τ-monotonic path. A τ-monotonic path ensures getting closer to q by at least τ in each step except the last step.

Definition 5. *(τ-monotonic path) A path $P = [v_0, v_1, ..., v_x, v_{x+1}]$ on a PG G is a τ-monotonic path for a query q if $\delta(v_{i+1}, q) < \delta(v_i, q) - \tau$, for $i = 0, ..., x - 1$, and $\delta(v_{x+1}, q) < \delta(v_x, q)$.*

Based on the τ-monotonic path, we define the τ-monotonic property as follows.

Definition 6. *(τ-monotonic property) Given a database D and a constant $\tau > 0$, a PG G of D is τ-monotonic if for any query q satisfying $\delta(q, \bar{v}) < \tau$, G has a τ-monotonic path starting from any node in G to the nearest neighbor $\bar{v}$ of q in D.*

Based on the edge occlusion rule (Definition 3), we can prove that if $\delta(q, \bar{v}) < \tau$, τ-MG must have a τ-monotonic path starting from any node to $\bar{v}$. Therefore, we have the following lemma.

Lemma 2. *A τ-MG of D is τ-monotonic.*

5.1 Construction of τ-MG

The overall idea of τ-MG construction is that we first construct an MRNG and then insert edges to the MRNG to obtain a τ-MG. It is because that by the definitions of MRNG and τ-MG, a τ-MG is an MRNG but an MRNG may lack some edges to satisfy the definition of τ-MG. From the example shown in Fig. 3(c),

Algorithm 2 Construction of τ-MG

Input: database D, parameter τ
Output: τ-MG G of D
1: construct an MRNG G_0 using the method in [17] and set $G = G_0$
2: **for** each node u in G **do**
3: L is the list of the nodes that are not out-going neighbors of u
4: sort L by the distances to u in ascending order
5: **for** $i = 0$ to $|L| - 1$ **do**
6: $v = L[i]$
7: **if** $(u, v) \in G$, **continue**
8: **if** $\delta(u, v) \leq 3\tau$, insert (u, v) to G
9: **else if** $\nexists (u, u') \in G$ s.t. $u' \in ball(u, \delta(u, v)) \cap ball(v, \delta(u, v) - 3\tau)$
10: insert (u, v) to G ▷ *e.g.*, Fig. 3(c)
11: **end for**
12: **end for**
13: **return** G

if u has an outgoing neighbor u'' in the gray region, (u, u'') can occlude (u, v) in MRNG, whereas (u, u'') cannot occlude (u, v) in τ-MG.

Algorithm 2 shows the construction algorithm of τ-MG. Specifically, Line 1 constructs an MRNG. For each node u in the MRNG, Lines 3-4 sort the list L of the nodes that are not outgoing neighbors of u in the ascending order of their distances to u. For the i-th node v of L, Lines 6-10 check Definition 4 to decide whether (u, v) needs to be inserted into the τ-MG.

The node degree of τ-MG is analyzed in the following lemma.

Lemma 3. *Given a database D of n points in E^m and a constant $\tau > 0$, under the assumptions in Theorem 1, the expected node degree of the τ-MG G constructed by Algorithm 2 is $O(\ln n)$ and the expected size of G is $O(n \ln n)$.*

Proof Sketch. The work [17] has proved that the expected node degree of MRNG is a constant. Hence, we only need to prove that the expected numbers of edges inserted by Line 8 and Line 10 for each node u in G_0 are both $O(\ln n)$.

For Line 8, since the n points are uniformly distributed and the density is g, the number of edges inserted by Line 8 for u is $Vol(ball(u, 3\tau)) \times g$. Since τ is a constant and $g = O(\ln n)$, $Vol(ball(u, 3\tau)) \times g$ is $O(\ln n)$.

For Line 10, if Line 10 inserts an edge (u, v), u must have a neighbor u'' in G_0 s.t. u'' is in $lune(u, v) \backslash ball(v, \delta(u, v) - 3\tau)$ (see the gray region in Fig. 3(c)). The probability of inserting (u, v) is $Pr(u, v) = Vol(lune(u, v) \backslash ball(v, \delta(u, v) - 3\tau)) / Vol(lune(u, v))$ as the points are uniformly distributed. The expected number of edges inserted by Line 10 is $\sum_{v \in N^-_{G_0}(u)} Pr(u, v)$. By applying arithmetic derivation and geometry theorems, we can derive $\sum_{v \in N^-_{G_0}(u)} Pr(u, v)) = O(\ln n)$. □

The time complexity for τ-MG construction is analyzed as below.

Lemma 4. *Given a database D of n points, the time complexity for τ-MG construction (Algorithm 2) is $O(n^2 \ln n)$.*

The time complexity of constructing τ-MG is the same as that of MRNG, FANNG, and SSG.

5.2 ANN search on τ-MG

In this subsection, we propose a new greedy routing algorithm on τ-MG. The novelty is that each routing step can get closer to q by at least τ, except the last step, such that the expected routing length is $O(n^{\frac{1}{m}} \ln n)$.

Algorithm 3 ANN search on τ-MG

Input: τ-MG G, query q
Output: ANN of q
1: select a node u in G as the start of searching
2: **while** true **do**
3: $u' = \arg\min_{v \in N_G(u) \wedge v \notin ball(u,3\tau)} \delta(q,v)$
4: **if** $\delta(q,u') > \delta(q,u)$ **then**
5: **return** $\arg\min_{v \in N_G(u) \wedge v \in ball(u,3\tau)} \delta(q,v)$
6: **end if**
7: $u = u'$
8: **end while**

The main idea of the greedy routing algorithm is that let u be the current node of the routing. We first try to route to u's neighbors out of $ball(u, 3\tau)$. If all the neighbors out of $ball(u, 3\tau)$ are farther from q than u, the routing stops and we scan u's neighbors in $ball(u, 3\tau)$ to find the search result. Algorithm 3 presents the detailed routing algorithm on τ-MG. Importantly, the greedy routing is designed based on the following property of τ-MG.

Lemma 5. *Given a τ-MG G and a query q satisfying $\delta(q, \bar{v}) < \tau$, let u be any node in G, if all outgoing neighbors of u out of $ball(u, 3\tau)$ are farther from q than u, then $\bar{v}$ is in $ball(u, 3\tau)$.*

We analyze the length of the routing path on τ-MG as follows.

Theorem 2. *Given a database D of n points in E^m, under the same assumptions as in Theorem 1, for a query q satisfing $\delta(q, \bar{v}) < \tau$, where $\tau > 0$ is a constant and $\bar{v}$ is the nearest neighbor of q in D, the expected length of the routing of Algorithm 3 starting from any node of G to $\bar{v}$ is $O(n^{\frac{1}{m}} \ln n)$ and the time complexity of Algorithm 3 is $O(n^{\frac{1}{m}} (\ln n)^2)$.*

Proof Sketch. Let $[v_0, v_1, ..., v_x]$ be the path found by Algorithm 3. Since $\delta(q, \bar{v}) < \tau$, we can prove $\mathbb{E}[x] \leq \frac{\mathbb{E}[\ln \delta(q,v_x)] - \ln(R+\tau)}{\ln R - \ln(R+\tau)}$, where R is the largest distance between any two points in D. Since the points are uniformly distributed, we can prove $\frac{\mathbb{E}[\ln \delta(q,v_x)] - \ln(R+\tau)}{\ln R - \ln(R+\tau)} \leq O\big(\frac{-\ln(r(n)+\tau)}{\ln r(n) - \ln(r(n)+\tau)}\big)$, where $r(n) = (\frac{n}{C})^{1/m}$ and C is a constant. Further, we prove that $\frac{-\ln(r(n)+\tau)}{\ln r(n) - \ln(r(n)+\tau)}$ has the same order of growth rate with $\frac{(r(n)+\tau)\ln(r(n)+\tau)}{\tau}$. Hence, $\mathbb{E}[x] \leq O(\frac{(r(n)+\tau)\ln(r(n)+\tau)}{\tau})$. Since τ is a constant, $\mathbb{E}[x] \leq O(n^{\frac{1}{m}} (\ln n))$. Because the expected node degree is $O(\ln n)$ as proved in Lemma 3, the time complexity of Algorithm 3 is $O(n^{\frac{1}{m}} (\ln n)^2)$.

5.3 Update of τ-MG

To support updates, we need to reconstruct the τ-MG G periodically. To optimize update cost, we reconstruct G after $O(\ln n)$ updates. Our idea is as follows.

If a node u is inserted to G, we first compute the distances from u to all nodes in G. Second, we compute the out-going neighbors of u using Lines 3-11 of Algorithm 2. Third, we update the out-going neighbors of all nodes in G. Specifically, for each node v in G, if no existing out-going edge of v can occlude the edge (v, u), (v, u) is inserted into G. The time complexity of insertion is $O(n \ln n)$.

If a node u is deleted from G, we adopt the masking strategy [56] to preserve the connectivity of G. Specifically, u is not deleted from G; u can be used in routing; but u is not returned as a query result. The time complexity of deletion is $O(1)$.

The graph G after updates is still a τ-MG by Definition 4. Therefore, Algorithm 3 can find $\bar{v}$ on the updated G if $\delta(q, \bar{v}) < \tau$. However, a node in the updated G can have more than $O(\ln n)$ out-going

Algorithm 4 Construction of τ-MNG

Input: database D, parameter τ, neighborhood size h, beam size b
Output: τ-MNG G of D
1: construct a PG G_0 for D ▷ *e.g.*, NSG [17] or HNSW [36]
2: set $G = (D, \emptyset)$ ▷ edge set is empty
3: **for** each u in G **do**
4: randomly select a node v_0 in G_0
5: H_u = beam_search(G_0, u, v_0, b, h) ▷ approx. h-NNs of u
6: sort the nodes in H_u by the ascending order of their distances to u
7: **for** $i = 0$ to $h - 1$ **do**
8: $v = H_u[i]$
9: **if** $(u, v) \in G$, **continue**
10: **if** $\delta(u, v) \leq 3\tau$, insert (u, v) to G
11: **if** $\nexists (u, u') \in G$ s.t. $u' \in ball(u, \delta(u, v)) \cap ball(v, \delta(u, v) - 3\tau)$ **then**
12: insert (u, v) to G
13: **end if**
14: **end for**
15: **end for**
16: **return** G

neighbors. To retain the same space complexity of τ-MG and time complexity for searching, we reconstruct G after $O(\ln n)$ updates.

As a remark, we can still use τ-MG to find the exact NN $\bar{v}$ for a query q in general metric spaces if the distance between q and $\bar{v}$ is less than τ. However, it is open whether the time and space complexities of τ-MNG proposed in this section hold under this setting.

6 τ-MONOTONIC NEIGHBORHOOD GRAPH (τ-MNG)

Since constructing a τ-MNG takes $O(n^2 \ln n)$ time, this section proposes an approximation of τ-MNG, namely τ-monotonic neighborhood graph (τ-MNG), such that τ-MNG can be constructed efficiently and the search on τ-MNG has a high probability of finding $\bar{v}$.

The main idea of τ-MNG is that we only require the neighborhood of each node in a τ-MNG to be τ-monotonic, where the neighborhood of a node u is the subgraph of G induced by the near neighbors of u. It is motivated by the recent observation that most routing steps in a proximity graph G are in the neighborhood of the nearest neighbor of q [46, 51]. τ-MNG is defined as follows.

Definition 7. *Given a database D and a constant $\tau > 0$, let $H_v \subset D$ denote the approximate h-nearest neighbors of a point $v \in D$, a τ-monotonic neighborhood graph (τ-MNG) is a directed graph $G = (V, E)$, where for any two nodes $v \in G$ and $u \in H_v$*

- *if $\delta(u, v) \leq 3\tau$, $(u, v) \in G$; and*
- *if $\delta(u, v) > 3\tau$ and $(u, v) \notin G$, G has an edge (u, u'), $u' \in H_v$, occluding the edge (u, v).*

6.1 Construction of τ-MNG

Algorithm 4 presents the construction algorithm of τ-MNG. Line 1 constructs a PG, *e.g.*, NSG [17] and HNSW [36]. Then, for each node u, Lines 4-5 find the list H_u of the approximate h-NNs of u. Line 6 sorts the nodes in H_u by their distances to u. Then, for each node v in H_u, if $(u, v) \notin G$, we insert (u, v) based on Definition 4. The time complexity of Algorithm 4 is analyzed in Lemma 6.

Lemma 6. *Given a database D of n points in E^m, a constant $\tau > 0$ and constants h and b, under the assumptions in Theorem 1, the time complexity to construct a τ-MNG by Algorithm 4 is $O(n^{\frac{2+m}{m}} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$.*

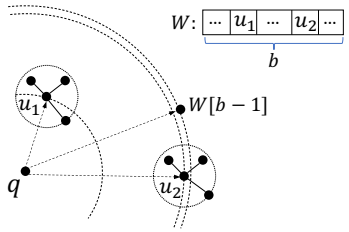


Fig. 4. Illustration of query-aware edge occluding (u_1 and u_2 are two nodes in W; $W[b-1]$ is the last node in W; and solid lines denote edges in PG)

By comparing Lemma 6 with Lemma 4, we can see that the time complexity for τ-MNG construction is much lower than that for τ-MG construction.

6.2 ANN search on τ-MNG

One may attempt to directly use Algorithm 3 on τ-MNG for ANN search. However, τ-MNG may not have a monotonic path from any node to $\bar{v}$. Therefore, Algorithm 3 on τ-MNG may get stuck in local optima, which reduces search accuracy. To alleviate this problem, we adopt the widely used beam search Algorithm 1 to strike a balance between search accuracy and efficiency. In this subsection, we first analyze the theoretical performance of the beam search on τ-MNG. Then, we propose three optimizations to address the performance bottlenecks of the beam search discovered in our experiments.

The accuracy of the result of the beam search depends on if the beam search enters the neighborhood $H_{\bar{v}}$ of $\bar{v}$. We call the beam search enters $H_{\bar{v}}$ if any node in $H_{\bar{v}}$ is added into the priority queue W in Algorithm 1.

Lemma 7. *Given a τ-MNG G of D and a constant $\tau > 0$, for a query q satisfying $\delta(q, \bar{v}) < \tau$, let $H_{\bar{v}}$ be the approximate h-NNs of q and u be the node in $H_{\bar{v}}$ that is the farthest from q, if the beam size is larger than $h + h'$, where h is the neighborhood size and h' is the number of nodes in G closer to q than u, the probability that Algorithm 1 finds $\bar{v}$ is no smaller than the probability that the beam search enters $H_{\bar{v}}$.*

The value of h' depends on q. While there is no definite expression of h', in practice, we can easily use a query workload to determine the optimal value of h' w.r.t the accuracy. For example, as observed in our experiments, h' can be tuned to obtain a recall higher than 0.95. This empirical result is consistent with those from the existing works [46, 51] that the beam search empirically has a high chance to enter the neighborhood $H_{\bar{v}}$ of $\bar{v}$.

6.2.1 Optimization for the search algorithm. We propose a query-aware edge occluding method (QEO) to reduce the number of distance computations in the beam search on τ-MNG.

Line 8 of Algorithm 1 compares the distance between q and each neighbor v of the current node u with the $(b-1)$-th node in W. If $\delta(q, v) > \delta(q, W[b-1])$, v is pruned. In our preliminary experiments, we observe that the farther the current node u from q, the higher probability that the neighbors of u can be pruned by the $(b-1)$-th node in W. The intuition is that assume the neighbors of u are uniformly distributed in $ball_u$ centered at u with radius $\max_{v \in N_G(u)} \delta(u, v)$. The intersection of $ball(q, \delta(q, W[b-1]))$ and $ball_u$ reduces with the growth of the distance between q and u. Fig. 4 illustrates the intuition. u_1 is close to q and the entire $ball_{u_1}$ is in $ball(q, \delta(q, W[b-1]))$, *i.e.*, all neighbors of u_1 cannot be pruned by $W[b-1]$. u_2 is far from q and $ball_{u_2}$ is partially in $ball(q, \delta(q, W[b-1]))$, *i.e.*, some neighbors of u_2 can be pruned by $W[b-1]$.

Based on this observation, we propose a query-aware edge occluding method. Specifically, if the current node u is not in the top $p\%$ of W, for all neighbors of u, we compute a lower bound δ_{lb} of

their distances to q and sort the neighbors by δ_{lb}. Then, we compute δ for the top $p'\%$ neighbors and add them into W; the remaining $(1-p')\%$ neighbors are pruned. If u is in the top $p\%$ of W, we compute δ for all neighbors of u and add them into W.

δ_{lb} of a node v and q is defined as $\delta_{lb}(v,q) = \sqrt{\sum_{i=0}^{z}(v[i]-q[i])^2}$, where $v[i]$ and $q[i]$ denote the i-th dimensions of v and q, respectively, and $0 < z < m$. To improve the tightness of $\delta_{lb}(v,q)$, we perform an orthogonal transformation to v and q, such that the beginning dimensions of v and q have more percentage of their Euclidean distance. Specifically, $\delta_{lb}(v,q) = \sqrt{\sum_{i=0}^{z}((Uv)[i]-(Uq)[i])^2}$, where U is an m by m orthogonal matrix and can be computed by the singular value decomposition (SVD) of D. Uv can be computed offline for any $v \in D$.

6.2.2 Implementation details. We discuss two important implementation details for search efficiency.

Partial distance-based pruning (PDP). During the beam search, for a neighbor v of the current node u, as long as we can decide $\delta(q,v) > \delta(q, W[b-1])$, we can prune v without exactly computing $\delta(q,v)$. It can save many computation costs and motivates a partial distance-based pruning method. Specifically, in the m iterations for computing the sum $\sum_{i=0}^{m}(v[i]-q[i])^2$, if we find $\sum_{i=0}^{j}(v[i]-q[i])^2$ at the j-th iteration is already larger than $(\delta(q, W[b-1]))^2$, we can simply prune v.

Prefix inner product index (PII). The computation of $\delta(q,v)$ can be reformulated as $\langle q,q\rangle + \langle v,v\rangle - 2\times\sum_{i=0}^{m}(v[i]\times q[i])$, where $\langle\cdot,\cdot\rangle$ denotes the inner product. $\langle v,v\rangle$ can be computed offline. We only need to compute $\langle q,q\rangle$ and $v[i]\times q[i]$ online. Since computing $v[i]\times q[i]$ only needs half operations of computing $(v[i]-q[i])^2$ and the cost of computing $\langle q,q\rangle$ can be shared by all distance computations in search, approximately half of the total cost of distance computations can be saved. To integrate with the partial distance-based pruning, we divide the vector v into segments and index the inner products for the prefix segments. Specifically, given a segment size parameter s, we index inner products $\langle v[0, i\times s], v[0, i\times s]\rangle$, $0 < i < \lceil m/s\rceil$. We perform partial distance-based pruning segment by segment.

6.3 Update of τ-MNG

Similar to Section 5.3, we propose to reconstruct the τ-MNG G periodically after $O(\ln n)$ updates. For insertion, we simply adopt the strategy of HNSW. Specifically, we use a beam search on G to find the h-ANNs H_u of the node u to-be-inserted. Then, the edge occlusion rule (Definition 3) is used to find the edges between u and the nodes in H_u. For deletion, the method in Section 5.3 is used.

7 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of our proposed techniques. The experiments verify that our method outperforms five current state-of-the-art methods on well-known real-world datasets.

Datasets and query workloads. We conduct the experiments on eight real-world datasets, which are widely used for ANN search [16, 17, 32, 36, 53]. The datasets cover a wide range of applications, including image (SIFT [3], BIGANN [3], DEEP [32], DEEP1B [57], and GIST [3]), text (GLOVE [24] and CRAWL [5]), and audio (MSONG [4]). Table 1 summarizes some characteristics of the datasets, including the dimensionality, the number of data points (# base), the local intrinsic dimensionality (LID) [1], and the data type. We can observe that the LIDs of the datasets are low. DEEP1B and BIGANN are used in the experiment against dataset size. Following [17, 32, 53], we also use synthetic datasets RAND and GAUSS. RAND is uniformly sampled from the hypercube $[0,1]^{100}$. GAUSS is generated by randomly choosing 10 cluster centers in the hypercube $[0,10]^{100}$, and each cluster follows the Gaussian distribution on each dimension. The query workloads of the real-world and synthetic datasets are given in the datasets and sampled from the datasets, respectively.

Table 1. Statistics of datasets

Dataset	m	# base	# query	LID	Type	Dataset	m	# base	# query	LID	Type
GloVe	100	1.18M	10K	20.0	Text	SIFT	128	1M	10K	9.3	Image
DEEP	256	1M	200	12.1	Image	CRAWL	300	1.98M	10K	15.7	Text
MSONG	420	0.99M	200	9.5	Audio	GIST	960	1M	1K	18.9	Image
DEEP1B	96	1B	10K	11.9	Image	BIGANN	128	1B	10K	9.3	Image

Baseline methods. We compare our method with five current state-of-the-art methods NSG [17], NSSG [16], HNSW [16], DPG [32], and FANNG [20]. NSG and its extension NSSG are the latest RNG-based methods. HNSW is the latest NSWG-based method. DPG is the latest kNNG-based method. FANNG is also an RNG-based method but is the only existing method considering $\delta(q, \bar{v}) < \tau$. The source codes of NSG, NSSG, DPG, and HNSW are obtained online. We implement FANNG, as its source code is not published. The code of τ-MNG is built on top of the code of NSG.

Metrics. The performance metrics follow the previous works [16, 17, 20, 36, 42, 53]. Specifically, we use the recall at k ($recall@k$) and the relative distance error ($rderr$) to measure search accuracy. For a query q, $recall@k = |kANNs \cap kNNs|/k$, where $kANNs$ is the set of the approximate k-NNs and $kNNs$ is the set of the exact k-NNs; $rderr = avg_{i=0}^{k-1}(\delta(q, i^{th}ANN)/\delta(q, i^{th}NN) - 1)$, where $i^{th}ANN$ and $i^{th}NN$ are the i-th approximate NN and exact NN of q, respectively. We report the average $recall@k$ and $rderr$ of all queries in the workloads. Following [16, 17, 36, 45, 53, 59], we use queries per second (QPS) and number of distance computations (NDC) to measure search efficiency. QPS is the number of queries finished in a second and NDC is the number of distance computations to evaluate a query. We focus on search performance in the high recall region.

Experimental settings. The experiments are conducted using C++ on a server with a Quad-Core AMD Opteron CPU and 800G RAM. The codes are compiled by g++ 8.5. Following [17, 32, 53], we evaluate the algorithms with a single thread. The larger the size b, the higher the recall, but the higher the query latency. Following recent works [31, 32, 53], we increase b until the target recall is achieved. We focus on $k = 100$ in the experiments.

7.1 Comparision with the baseline methods

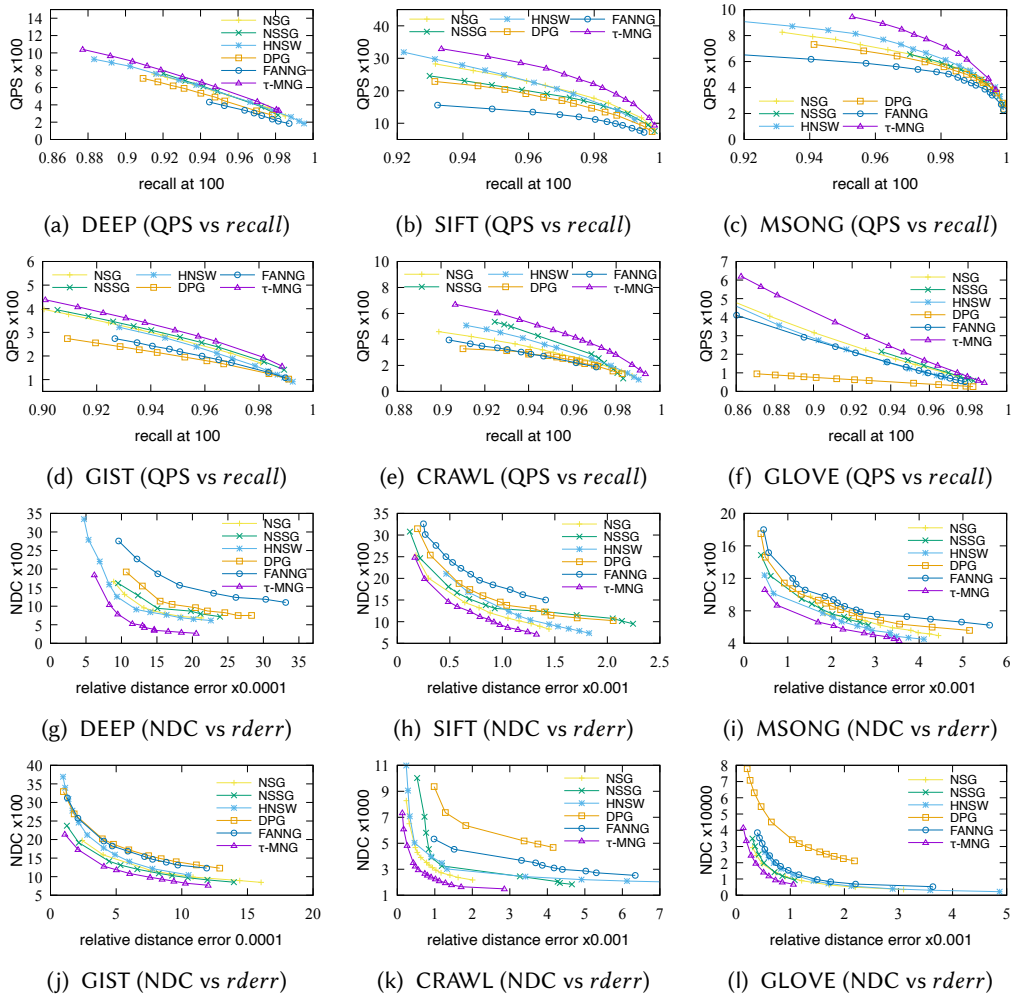
In this experiment, we compare our method with five baseline methods on six datasets. For a fair comparison, we integrate our optimization QEO and two implementation details PDP and PII into all the baseline methods. The results are shown in Fig. 5.

From Fig. 5, we can observe that τ-MNG outperforms all the baseline methods. In particular, on DEEP, SIFT, MSONG, GIST, CRAWL, and GLOVE, when $recall@100$ is 0.95, τ-MNG is ∼1.1 to ∼1.6, ∼1.2 to ∼2.1, ∼1.1 to ∼1.6, ∼1.1 to ∼1.5, ∼1.2 to ∼2.0, and ∼1.2 to ∼4.1 times faster than the baseline methods, respectively; when $rderr$ is 0.001, τ-MNG is ∼1.7 to ∼5.2, ∼1.2 to ∼2.0, ∼1.2 to ∼1.8, ∼1.2 to ∼1.7, ∼1.3 to ∼3.3, and ∼1.4 to ∼5.1 times faster than the baseline methods, respectively.

7.2 Effect of τ

In this experiment, we study the performance of τ-MNG by varying τ. To clearly examine the effect of τ, QEO, PDP, and PII are not used.

Fig. 6 shows the results on the six real-world datasets. From Fig. 6, we observe that the performance of τ-MNG first improves and then deteriorates with the growth of τ. The reason is that the search cost is dominated by the NDC in the search and the expected NDC is bounded by the product of the expected number of search steps and the expected node degree of the PG. If τ increases, the node degree of τ-MNG grows and the connectivity of τ-MNG is better. The search involves less detour, which reduces NDC. However, if τ increases further, the node degree becomes too large and the search has to compute distances for more neighbors at each search step, which results in a large



(a) DEEP (QPS vs *recall*) (b) SIFT (QPS vs *recall*) (c) MSONG (QPS vs *recall*)

(d) GIST (QPS vs *recall*) (e) CRAWL (QPS vs *recall*) (f) GLOVE (QPS vs *recall*)

(g) DEEP (NDC vs *rderr*) (h) SIFT (NDC vs *rderr*) (i) MSONG (NDC vs *rderr*)

(j) GIST (NDC vs *rderr*) (k) CRAWL (NDC vs *rderr*) (l) GLOVE (NDC vs *rderr*)

Fig. 5. Comparison with existing k-ANN search methods

NDC. Since the results of NDC vs *rderr* are consistent with the results of QPS vs *recall*, we simply present the results of QPS vs *recall* in following experiments due to space limitations.

We further conduct experiments on synthetic datasets GAUSS and RAND with different point densities and deviations to study the behavior of τ-MNG. Fig. 7 shows the results. In Figs. 7(a)-(c), we fix the standard deviation (SD) of GAUSS to be 5. In Fig. 7(d), we tune the value of SD.

From Figs. 7(a)-(c), we first observe that k-ANN search on GAUSS is faster than RAND. A reason is that the search on RAND has more steps than GAUSS. For example, on the datasets with 10K points, to achieve *recall*@100 = 0.95, the number of search steps on 0-MNG of RAND is ~400 and that on 0-MNG of GAUSS is ~90. Second, we observe that the effect of τ is more noticeable on GAUSS than RAND. A reason is that the edges among the k-NNs of q have a higher chance to be occluded on RAND than GAUSS. Therefore, the decrease of the number of search steps on GAUSS is faster than RAND as τ increasing. For example, on the datasets with 10K points, to achieve *recall*@100 = 0.95, the number of search steps on 0.2-MNG of RAND is ~300 and that on 8-MNG of GAUSS is ~50. Third, we observe that the value of τ, which changes the performance trend of τ-MNG from an improving trend to a deteriorating one, reduces with the growth of point density. A

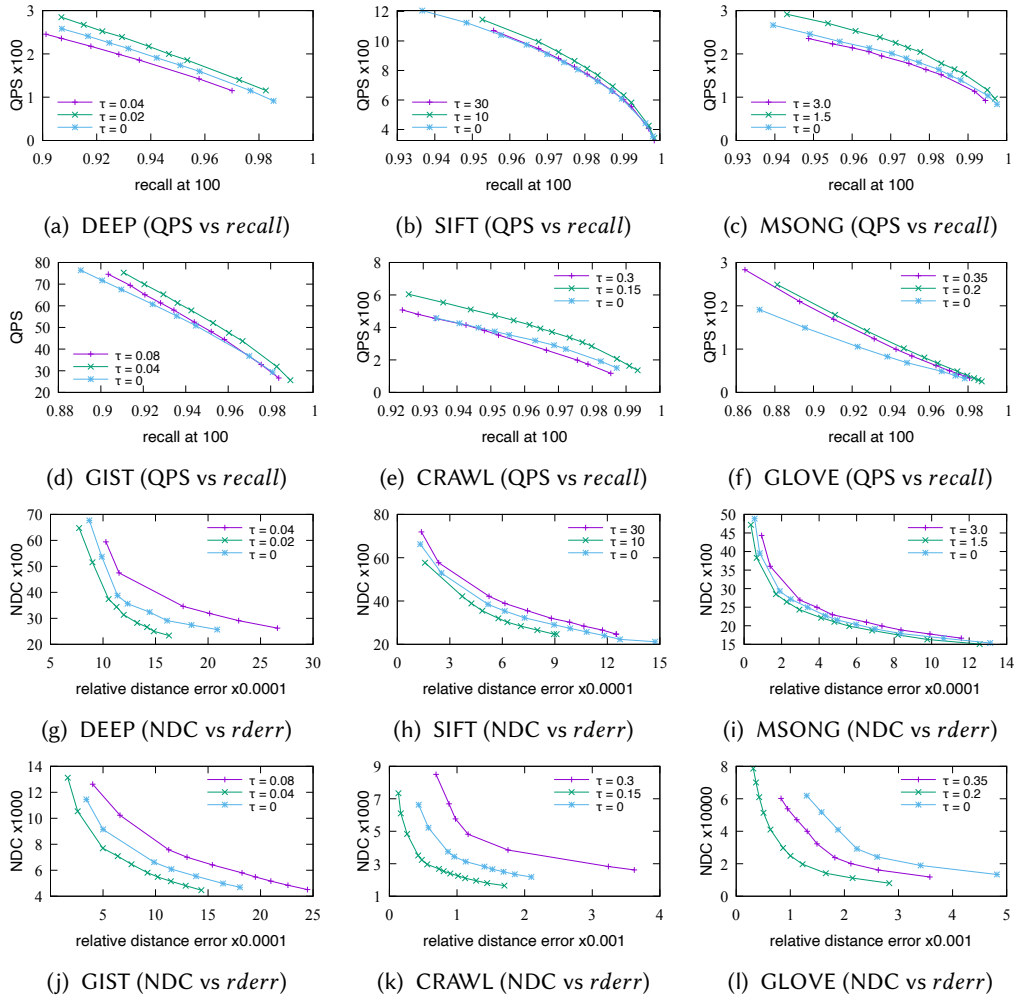


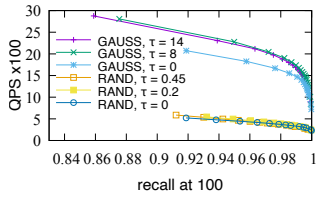
Fig. 6. Effect of τ

reason is that on dense datasets, the growth of τ-MNG's node degree by increasing τ is faster than the decrease of the number of search steps.

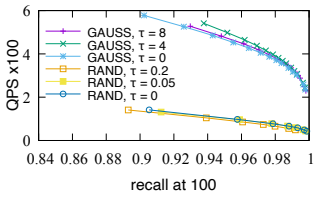
Fig. 7(d) shows the speedup (spd) of τ-MNG on GAUSS with different standard derivations. spd is defined as the QPS of τ-MNG over the QPS of 0-MNG, when $recall@100 = 0.95$. We observe that the performance of τ-MNG first improves and then deteriorates with the growth of τ for each SD value, but the value of τ, which changes the trend from an improving trend to a deteriorating one, first increases and then decreases with the growth of SD. A reason is that the edge length in τ-MNG first increases and then decreases with the growth of SD. For example, the average edge lengths of 0-MNG's are ~36, ~59, ~82, and ~12 for GAUSS with SD 3, 5, 7, and 10, respectively.

7.3 Performance of QEO

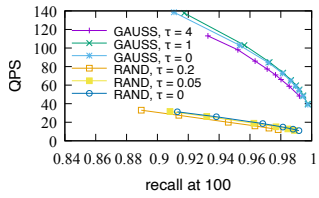
In this experiment, we vary p to study the performance of the query-aware edge occlusion (QEO). $p'\%$ and z are set to be 2 and $m/2$, respectively. To focus on QEO, the partial distance-based pruning and the prefix inner product index techniques are not used in this experiment. Fig. 8 shows the results from the six datasets.



(a) 10K points



(b) 100K points

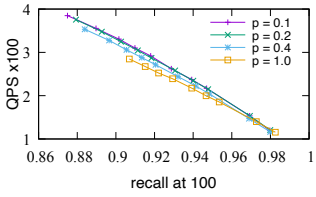


(c) 1M points

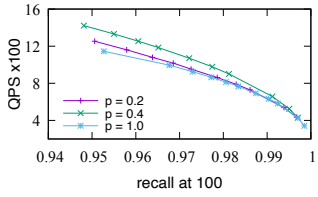
std = 3		std = 5		std = 7		std = 10	
τ	spd	τ	spd	τ	spd	τ	spd
0	1.0	0	1.0	0	1.0	0	1.0
0.5	**1.1**	0.9	1.0	1.2	1.0	**0.2**	**1.1**
2.1	1.0	**3.6**	**1.1**	**5.0**	**1.2**	0.8	1.0
4.8	0.8	7.9	1.0	11.0	1.0	1.7	0.8

(d) 100K points of GAUSS

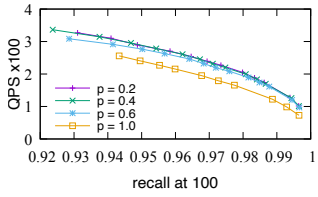
Fig. 7. Effect of τ with respect to different point distributions



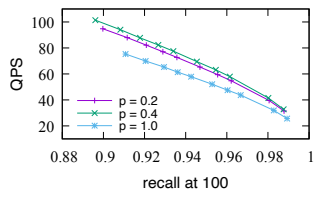
(a) DEEP



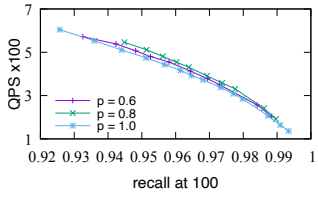
(b) SIFT



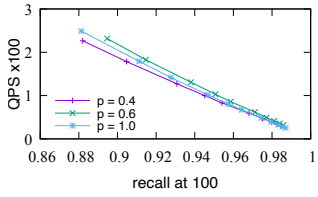
(c) MSONG



(d) GIST



(e) CRAWL



(f) GLOVE

Fig. 8. Performance of QEO

Fig. 8 shows that the search performance first increases and then reduces (Fig. 8(b) and 8(d)-8(f)) or remains stable (Fig. 8(a) and 8(c)) with the reduction of p. The reason is that with the reduction of p, the search occludes more unpromising neighbors at each search step, which reduces NDC. However, if p reduces further, more promising neighbors are occluded and the search has to detour more, which increases NDC.

7.4 Performance of PDP and PII

This experiment evaluates the performance of PDP and PII discussed in Section 6.2.2. For space limitations, we just present the results on SIFT in Fig. 9 and the trends on other datasets are similar.

From Fig. 9, we can observe that PDP improves the search efficiency. The reason is that at each search step, many neighbors of the current node can be pruned by the $(b-1)$-th node in the priority queue without fully computing their distances from q. In particular, when the recall is 0.95, the

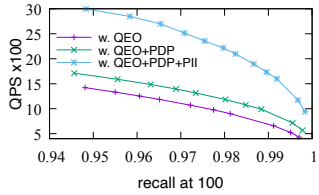


Fig. 9. Effect of PDP & PII

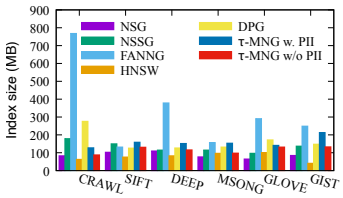
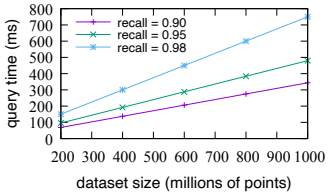
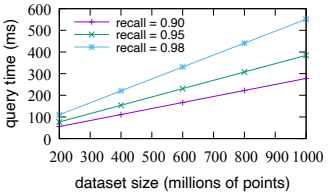


Fig. 10. Index size



(a) DEEP1B



(b) BIGANN

Fig. 11. Performance against dataset size

QPS of τ-MNG + QEO + PDP is 1.2 times larger than the QPS of τ-MNG + QEO on SIFT. Fig. 9 further shows that the search throughput is greatly improved by PII. In particular, when the recall is 0.95, the QPS of τ-MNG + QEO + PDP + PII is 2.1 times larger than the QPS of τ-MNG + QEO on SIFT.

7.5 Comparison of index size

We examine the index size in this experiment. Fig. 10 shows the results. From Fig. 10, we can observe that τ-MNG with and without PII are only slightly larger than NSG, NSSG, and HNSW, which are the top three baseline methods as compared in Fig. 5. τ-MNG with and without PII are much smaller than FANNG, especially on CRAWL, DEEP, and GLOVE. Therefore, τ-MNG can be comfortably stored in main memory.

7.6 Performance against dataset size

In this experiment, we evaluate the performance of τ-MNG against dataset size on DEEP1B and BIGANN. Following the approach of [16, 17, 43, 53] for supporting large datasets, we randomly separate the data points into parts of 10 millions of points. A τ-MNG is built for each part. We perform k-ANN search on each part and return the top-k among the results of all parts as the final results. Fig. 11 shows the performance, where the x-axis is the dataset size and the y-axis is the average running time of a query. We observe that the query time of τ-MNG increases linearly with the growth of dataset size. From Fig. 11, we can also observe that the gap between the plots of recall 0.90 and 0.95 is smaller than the gap between the plots of recall 0.95 and recall 0.98. It is consistent with the observations in previous experiments that the growth of running time is more than a linear function of the growth of recall.

8 CONCLUSION

In this paper, we propose a τ-monotonic graph (τ-MG) for the approximate nearest neighbor search in multi-dimensional databases. The core of τ-MG is a novel edge occlusion rule. When the distance from q to its nearest neighbor in the database is less than a constant τ, the greedy routing on τ-MG

guarantees to find the exact nearest neighbor of q and the expected time complexity of the search is smaller than all existing methods. The expected length of the greedy routing in τ-MG and the expected node degree of τ-MG have been rigorously analyzed and presented in the supplementary materials. For the efficiency of index construction, we propose a τ-monotonic neighborhood graph (τ-MNG), which is an approximate variant of τ-MG. We further propose an optimization to reduce the number of distance computations in the search on τ-MNG. Our extensive experiments show that our method is effective and outperforms the state-of-the-art ANN search methods on real-world benchmark datasets.

In the future, we plan to incorporate distributed and external-memory ANN search methods into our proposed techniques.

9 SUPPLEMENTARY MATERIALS

To enhance the presentation flow of the paper, we present the detailed proofs in this section.

9.1 Proof of Theorem 1

Proof. Since the greedy routing finds a monotonic path for q in G, we prove this theorem by analyzing the expected length of the path. Let $P = [v_0, v_1, ..., v_x]$, $x \geq 0$, denote the monotonic path for q. Note that v_x may not be the nearest neighbor of q. We can build a sequence of concentric balls: $ball(q, \delta(q, v_0)), ..., ball(q, \delta(q, v_x))$. Let $\eta_{q,i} = \frac{Vol(ball(q,\delta(q,v_{i+1})))}{Vol(ball(q,\delta(q,v_i)))}$, $i = 0, ..., x-1$. We have $\eta_{q,i} = \left(\frac{\delta(q,v_{i+1})}{\delta(q,v_i)}\right)^m$, $i = 0, ..., x-1$. Let $\Delta_q = \min_{v,v' \in D} |\delta(q,v) - \delta(q,v')|$. We have $\Delta_q \leq \delta(q, v_i) - \delta(q, v_{i+1})$, $i = 0, ..., x-1$. Let $R_q = \max_{v \in D} \delta(q, v)$. We have

$$\eta_{q,i} \leq \left(\frac{\delta(q,v_i) - \Delta_q}{\delta(q,v_i)}\right)^m = \left(1 - \frac{\Delta_q}{\delta(q,v_i)}\right)^m \leq \left(1 - \frac{\Delta_q}{R_q}\right)^m = \left(\frac{R_q - \Delta_q}{R_q}\right)^m$$

$$Vol(ball(q, \delta(q, v_x))) = Vol(ball(q, \delta(q, v_0)))\eta_{q,0}\eta_{q,1}...\eta_{q,x-1} \leq Vol(ball(q, R_q))\left(\left(\frac{R_q - \Delta_q}{R_q}\right)^m\right)^x$$

It follows that

$$\frac{Vol(ball(q, \delta(q, v_x)))}{Vol(ball(q, R_q))} \leq \left(\left(\frac{R_q - \Delta_q}{R_q}\right)^m\right)^x \tag{1}$$

We perform logarithm operation on both sides of (1) and let $\hat{\eta}_q = (\frac{R_q - \Delta_q}{R_q})^m$ be the base of the logarithm. Since $\hat{\eta}_q < 1$, we have

$$x \leq \log_{\hat{\eta}_q} \frac{Vol(ball(q, \delta(q, v_x)))}{Vol(ball(q, R_q))} = \log_{\hat{\eta}_q} \left(\frac{\delta(q, v_x)}{R_q}\right)^m = m \log_{\hat{\eta}_q} \frac{\delta(q, v_x)}{R_q}$$

The expectation of x over all possible q is $\mathbb{E}[x] \leq m\mathbb{E}[\log_{\hat{\eta}_q} \frac{\delta(q,v_x)}{R_q}]$.

Let $R = \max_{u,v \in D} \delta(u, v)$, $\Delta = \min_{u,v,w \in D} |\delta(u, v) - \delta(u, w)|$, and $\hat{\eta} = (\frac{R - \Delta}{R})^m$. Recall the assumption that q follows the same distribution with the points in D. Therefore, when $n \to \infty$,

$$\mathbb{E}\big[\log_{\hat{\eta}_q} \frac{\delta(q, v_x)}{R_q}\big] \to \mathbb{E}\big[\log_{\hat{\eta}} \frac{\delta(q, v_x)}{R}\big]$$

Since m is a constant, $m\mathbb{E}[\log_{\hat{\eta}_q} \frac{\delta(q,v_x)}{R_q}] \to m\mathbb{E}[\log_{\hat{\eta}} \frac{\delta(q,v_x)}{R}]$. Therefore, $\mathbb{E}[x] \leq m\mathbb{E}[\log_{\hat{\eta}} \frac{\delta(q,v_x)}{R}]$. Since the work [17] has proved that $m\mathbb{E}[\log_{\hat{\eta}} \frac{\delta(q,v_x)}{R}] \leq \frac{\ln \mathbb{E}[\delta(q,v_x)] - \ln(n/C)^{1/m}}{\ln((n/C)^{1/m} - \Delta) - \ln(n/C)^{1/m}}$, where C is a constant, we have

$$\mathbb{E}[x] \leq \frac{\ln \mathbb{E}[\delta(q, v_x)] - \ln(n/C)^{1/m}}{\ln((n/C)^{1/m} - \Delta) - \ln(n/C)^{1/m}} \tag{2}$$

Since the left side of (2) is non-zero, the right side of (2) is non-zero. Since the denominator of the right side of (2) is negative, the numerator of the right side of (2) is not positive. Since $\delta(q, v_x) \geq \delta(q, \bar{v})$, we have

$$\mathbb{E}[x] \leq \frac{\ln \mathbb{E}[\delta(q,\bar{v})] - \ln(n/C)^{1/m}}{\ln((n/C)^{1/m} - \Delta) - \ln(n/C)^{1/m}} \tag{3}$$

Since q follows the same distribution with the points in D, $\mathbb{E}[\delta(q,\bar{v})]$ over all possible q equals to $\mathbb{E}[\delta(u,v')]$ overall possible u, where u is a point in D and v' is the nearest neighbor of u in D. Since the points are uniformly distributed and the point density g is $O(\ln n)$, we consider a hypercube centered at u with volume $2/g$. v' is in the hypercube in the expected case. Since v' is uniformly distributed in the hypercube, $\mathbb{E}[\delta(u,v')] \geq \frac{1}{2}(\frac{2}{g})^{1/m} \geq \frac{1}{2}(\frac{2}{\ln n})^{1/m}$. Since $\mathbb{E}[x] \geq 0$ and the denominator of (3) is negative, the numerator of (3) is negative. Therefore,

$$\mathbb{E}[x] \leq O(\frac{-\ln(n/C)^{1/m}}{\ln((n/C)^{1/m} - \Delta) - \ln(n/C)^{1/m}}) \tag{4}$$

According to [17], the right side of (4) is $O(\frac{1}{\Delta} n^{\frac{1}{m}} \ln n^{\frac{1}{m}})$. Hence,

$$\mathbb{E}[x] = O(\frac{1}{\Delta} n^{\frac{1}{m}} \ln n^{\frac{1}{m}}) \tag{5}$$

Next, we prove $\Delta \leq O(\sqrt{m}(m/n)^{1/m})$ with probability at least $1 - (\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$. Recall that the points are uniformly distributed with density g. We evenly split the whole space into hypercubes with volume $2/g$. We call these hypercubes $(2/g)$-volume hypercubes. The expected number of points in each $(2/g)$-volume hypercube is two. The number of $(2/g)$-volume hypercubes is $\frac{V_D}{2/g}$. Based on the assumption that the n points are uniformly distributed in space, we have $V_D = n/g$. Therefore, the number of $(2/g)$-volume hypercubes is $\frac{n/g}{2/g} = n/2$. Since the expected number of points in a $(2/g)$-volume hypercube is two, there must be some $(2/g)$-volume hypercubes having at least two points, whose count is calculated as follows.

Let *cube* be a randomly selected $(2/g)$-volume hypercube. Let Pr_0 and Pr_1 be the probabilities that *cube* has 0 and 1 point, respectively.

$$Pr_0 = \Big(\frac{n/2-1}{n/2}\Big)^n,\ Pr_1 = \frac{n \times (n/2-1)^{n-1}}{(n/2)^n} = \frac{n}{n/2-1}\Big(\frac{n/2-1}{n/2}\Big)^n$$

The limitations of Pr_0 and Pr_1 when $n \to \infty$ are as below.

$$\lim_{n\to\infty} Pr_0 = \lim_{n\to\infty} \left(\left(1 - \frac{1}{n/2}\right)^{n/2}\right)^2 = (\frac{1}{e})^2$$

$$\lim_{n\to\infty} Pr_1 = \left(\lim_{n\to\infty} \frac{n}{n/2-1}\right) \times \left(\lim_{n\to\infty} \left(\frac{n/2-1}{n/2}\right)^n\right) = 2(\frac{1}{e})^2$$

Let $Pr_{\geq 2}$ denote the probability that *cube* has two or more points. $Pr_{\geq 2} = 1 - Pr_0 - Pr_1$, and $\lim_{n\to\infty} Pr_{\geq 2} = 1 - 3/e^2$. Since the total number of $(2/g)$-volume hypercubes is $n/2$, the number of $(2/g)$-hypercubes having at least two points is $\frac{n}{2} Pr_{\geq 2}$.

To tightly upper bound the smallest distance between any two points, we further split each $(2/g)$-volume hypercube into hypercubes with volume m/ng, which are called (m/ng)-volume hypercubes. The number of (m/ng)-volume hypercubes in a $(2/g)$-volume hypercube is $\frac{2/g}{m/ng} = 2n/m$. Since m is a constant, $\frac{2n}{m} > 1$.

Let us consider a $(2/g)$-volume hypercube $cube_0$ having at least two points. If $cube_0$ has two points, since the two points are uniformly distributed in $cube_0$, the probability that the two points are both in a (m/ng)-volume hypercube is $\frac{2n/m}{(2n/m)^2} = m/2n$. If $cube_0$ has more than two points, the probability that $cube_0$ has one (m/ng)-volume hypercube holding two points is more than $\frac{2n/m}{(2n/m)^2} = m/2n$. Therefore, the probability that $cube_0$ has one (m/ng)-volume hypercube holding two points is at least $\frac{2n/m}{(2n/m)^2} = m/2n$.

Each $(2/g)$-volume hypercube with at least two points is regarded as a random experiment and $m/2n$ is the probability of success in a random experiment. By the Bernoulli's distribution, the probability of having at least one successful experiment in $\frac{n}{2}Pr_{\geq 2}$ experiments is $1-(1-\frac{m}{2n})^{\frac{n}{2}Pr_{\geq 2}}$. We compute the limitation as follows.

$$\lim_{n\to\infty} 1-\left(1-\frac{m}{2n}\right)^{\frac{n}{2}Pr_{\geq 2}} = 1-\lim_{n\to\infty}\left(1-\frac{m}{2n}\right)^{\frac{n}{2}Pr_{\geq 2}}$$

Since it is not obvious to compute $\lim_{n\to\infty}(1-\frac{m}{2n})^{\frac{n}{2}Pr_{\geq 2}}$, we compute $\lim_{n\to\infty}\ln(1-\frac{m}{2n})^{\frac{n}{2}Pr_{\geq 2}}$ as follows.

$$\lim_{n\to\infty}\ln\left(1-\frac{m}{2n}\right)^{\frac{n}{2}Pr_{\geq 2}} = \lim_{n\to\infty} Pr_{\geq 2}\ln\left(1-\frac{m}{2n}\right)^{\frac{n}{2}} = \lim_{n\to\infty} Pr_{\geq 2}\lim_{n\to\infty}\ln\left(1-\frac{m}{2n}\right)^{\frac{n}{2}}$$
$$=\left(1-\frac{3}{e^2}\right)\lim_{n\to\infty}\ln\left(1-\frac{1}{2n/m}\right)^{\frac{n}{2}} = \left(1-\frac{3}{e^2}\right)\lim_{n\to\infty}\ln\left(\left(1-\frac{1}{\frac{2n}{m}}\right)^{\frac{2n}{m}}\right)^{\frac{m}{4}} = \left(1-\frac{3}{e^2}\right)\ln\left(\frac{1}{e}\right)^{\frac{m}{4}} = \ln\left(\frac{1}{e}\right)^{\frac{m}{4}(1-\frac{3}{e^2})}$$

Therefore, $\lim_{n\to\infty} 1-(1-\frac{m}{2n})^{\frac{n}{2}Pr_{\geq 2}} = 1-(\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$.

Since the longest possible distance between any two points in a (m/ng)-volume hypercube is $\sqrt{m}(\frac{m}{ng})^{1/m}$, we have $\Delta \leq \sqrt{m}(\frac{m}{ng})^{1/m}$. Since the density g increases with n, g must be larger than a constant, say g_0. Therefore,

$$\Delta \leq \sqrt{m}\left(\frac{m}{ng}\right)^{1/m} \leq \sqrt{m}\left(\frac{m}{ng_0}\right)^{1/m} = O(\sqrt{m}(m/n)^{1/m}) \tag{6}$$

Since m is a constant, combining (5) and (6) produces that $\mathbb{E}[x] \geq O(\frac{1}{\sqrt{m}m^{1/m}} n^{\frac{2}{m}}\ln n^{\frac{1}{m}}) = O(n^{\frac{2}{m}}\ln n)$ with probability at least $1-(\frac{1}{e})^{\frac{m}{4}(1-\frac{3}{e^2})}$. □

9.2 Proof of Lemma 2

Proof. If the edge $(v_0, \bar{v}) \in G$, it is trivial as $P = [v_0, \bar{v}]$ is a τ-monotonic path for q. If $(v_0, \bar{v}) \notin G$, we prove G has a τ-monotonic path for q by a case analysis.

Case i): *If $\delta(v_0, \bar{v}) \leq 6\tau$, G must have a τ-monotonic path $[v_0, v_1, \bar{v}]$ for q.* The reason is as follows. Since G is a τ-MG, v_0 must have an outgoing neighbor v_1 in $ball(v_0, \delta(v_0, \bar{v})) \cap ball(\bar{v}, \delta(v_0, \bar{v}) - 3\tau)$. Since $\delta(v_1, \bar{v}) \leq 3\tau$, $(v_1, \bar{v}) \in G$ and G must have a path $[v_0, v_1, \bar{v}]$. Since q is in $ball(\bar{v}, \tau)$, $\delta(q, v_1) < \delta(v_0, \bar{v}) - 3\tau + \tau = \delta(v_0, \bar{v}) - 2\tau$. Since $\delta(q, v_0) > \delta(v_0, \bar{v}) - \tau$, $\delta(q, v_1) < \delta(q, v_0) - \tau$. Therefore, $[v_0, v_1, \bar{v}]$ is a τ-monotonic path for q.

Case ii): *If $\delta(v_0, \bar{v}) > 6\tau$, G must have a τ-monotonic path $[v_0, v_1, ..., v_i, v_{i+1}, \bar{v}]$ for q such that $\delta(v_i, \bar{v}) \leq 6\tau$.* The reason is as follows. Since G is a τ-MG, v_0 must have an outgoing neighbor v_1 satisfying $\delta(v_1, \bar{v}) < \delta(v_0, \bar{v}) - 3\tau$ and $\delta(q, v_1) < \delta(q, v_0) - \tau$. Similarly, v_1 must have an outgoing neighbor v_2 satisfying $\delta(v_2, \bar{v}) < \delta(v_1, \bar{v}) - 3\tau$ and $\delta(q, v_2) < \delta(q, v_1) - \tau$. In this way, each step gets closer to $\bar{v}$ by at least 3τ and gets closer to q by at least τ. The search must reach a node v_i satisfying $\delta(v_i, \bar{v}) < 6\tau$. Since G must have a τ-monotonic path $[v_i, v_{i+1}, \bar{v}]$ for q, which has been proved in Case i), $[v_0, v_1, ..., v_i, v_{i+1}, \bar{v}]$ is a τ-monotonic path for q. □

9.3 Proof of Lemma 3

Proof. Since the maximum node degree of the MRNG G_0 is a constant that is independent of n [17], we only need to compute the expected numbers of edges inserted for each node u by Line 8 and Line 10 of Algorithm 2 are both $O(\ln n)$.

First, we prove that the expected number of edges inserted by Line 8 for a node u is $O(\ln n)$. Since Line 8 links u to all points of D in $ball(u, 3\tau)$, the expected number of edges inserted for u by Line 8 is the expected number of points of D in $ball(u, 3\tau)$. Since the points are uniformly distributed in the space and the density of the points is g, the expected number of points in $ball(u, 3\tau)$ is $Vol(ball(u, 3\tau)) \times g = O(\ln n)$ as τ is a constant and g is $O(\ln n)$.

Then, we prove that the expected number of edges inserted for u by Line 10 is $O(\ln n)$. Consider a node v, which is not an outgoing neighbor of u in the MRNG constructed in Line 1 of Algorithm 2. By the definition of MRNG, u must have at least a neighbor u' in $lune(u, v)$. If Line 10 inserts (u, v), u' must be in the gray region of $lune(u, v)$ in Fig. 3(c). The volume of the gray region is $Vol(lune(u, v) \backslash ball(v, \delta(u, v) - 3\tau))$. The probability of inserting (u, v) is $Pr(u, v) = \frac{Vol(lune(u,v) \backslash ball(v,\delta(u,v)-3\tau))}{Vol(lune(u,v))}$, due to the uniform distribution.

Suppose the line uv intersects with $ball(v, \delta(u, v) - 3\tau)$ at a point w, where w is not necessarily a point in D.

$$Pr(u, v) < \frac{Vol(lune(u, v)) - Vol(lune(w, v))}{Vol(lune(u, v))} = \frac{\delta(u, v)^m - (\delta(u, v) - 3\tau)^m}{\delta(u, v)^m} = 1 - \left(1 - \frac{3\tau}{\delta(u, v)}\right)^m$$

The expected number of inserted edges for u by Line 10 is

$$A = \sum_{v \in N^-_{G_0}(u)} \left(1 - \left(1 - \frac{3\tau}{\delta(u, v)}\right)^m\right)$$

Since $\delta(u, v) > 3\tau$, $0 < \frac{3\tau}{\delta(u,v)} < 1$. By Bernoulli's inequality, $(1 - \frac{3\tau}{\delta(u,v)})^m \geq 1 - \frac{m3\tau}{\delta(u,v)}$. Therefore, $A \leq \sum_{v \in N^-_{G_0}(u)} \frac{m3\tau}{\delta(u,v)} = m3\tau \sum_{v \in N^-_{G_0}(u)} \frac{1}{\delta(u,v)}$.

Suppose $\delta(u, v)$ follows some continuous distribution. By the property of continuous distributions, for any two nodes $v_1, v_2 \in N^-_{G_0}(u)$ and $v_1 \neq v_2$, $\delta(u, v_1) \neq \delta(u, v_2)$. Recall that R denotes the largest distance between any two points in D. We have the following.

$$A \leq m3\tau \sum_{v \in N^-_{G_0}(u)} \frac{1}{\delta(u, v)} \leq m3\tau \int_{3\tau}^{R} \frac{1}{x} dx \leq m3\tau(\ln R - \ln 3\tau)$$

Since m and τ are constants and independent of n, we have $A = O(\ln R)$. Since the points in D are uniformly distributed in the space and the density of the points is g, we have $g \cdot V_D = n$. Since there exists a constant ψ s.t. $\psi V_D \geq Vol(ball(R))$, $Vol(ball(R)) \leq \psi V_D = \psi \frac{n}{g}$. Since g increases with n, g must be larger than a constant, say g_0. Therefore, $Vol(ball(R)) \leq \psi \frac{n}{g} \leq \psi \frac{n}{g_0}$.

By the definition of the volume of ball, $Vol(ball(R)) = \frac{\pi^{m/2} R^m}{\Gamma(1+m/2)}$. Let $b = \frac{\pi^{m/2}}{\Gamma(1+m/2)}$. We have $b \cdot R^m \leq \psi \frac{n}{g_0}$ and $R \leq (\frac{\psi n}{g_0 b})^{1/m}$.

Since m, g_0, b, and ψ are constants and independent of n, we have $A = O(\ln n^{\frac{1}{m}}) = O(\ln n)$. The proof is finished. □

9.4 Proof of Lemma 5

Proof. It can be established by proof of contradiction. Assume $\bar{v} \notin ball(u, 3\tau)$. By Definition 4, $\bar{v}$ is a neighbor of u or u has a neighbor u' in $ball(u, \delta(u, v)) \cap ball(v, \delta(u, v) - 3\tau)$.

If $\bar{v}$ is a neighbor of u, it contradicts with the fact that as all the neighbors of u out of $ball(u, 3\tau)$ are farther away from q than u.

If u has a neighbor u' in the intersection of $ball(u, \delta(u, v))$ and $ball(v, \delta(u, v) - 3\tau)$, we have $\delta(q, u') < \delta(q, u)$. It is also a contradiction as all the neighbors of u out of $ball(u, 3\tau)$ are farther away from q than u. □

9.5 Proof of Theorem 2

Proof. Since $\delta(q, \bar{v}) < \tau$, Algorithm 3 must find a monotonic path P from any node v_0 to $\bar{v}$. Let $P = [v_0, v_1, ..., v_x, \bar{v}]$. We can build a sequence of concentric balls: $ball(q, \delta(q, v_0)), ..., ball(q, \delta(q, v_x))$.

Let $\eta_{q,i} = \frac{Vol(ball(q,\delta(q,v_{i+1})))}{Vol(ball(q,\delta(q,v_i)))}$, $i = 0, ..., x-1$. We have $\eta_{q,i} = \left(\frac{\delta(q,v_{i+1})}{\delta(q,v_i)}\right)^m$, $i = 0, ..., x-1$.

By Lemma 2, $\delta(q, v_i) - \delta(q, v_{i+1}) > \tau$, $i = 0, ..., x-1$. Let $R_q = \max_{v \in D} \delta(q, v)$. It follows that

$$\eta_{q,i} \leq \left(\frac{\delta(q, v_i) - \tau}{\delta(q, v_i)}\right)^m = \left(1 - \frac{\tau}{\delta(q, v_i)}\right)^m \leq \left(1 - \frac{\tau}{R_q}\right)^m = \left(\frac{R_q - \tau}{R_q}\right)^m$$

and

$$Vol(ball(q, \delta(q, v_x))) = Vol(ball(q, \delta(q, v_0)))\eta_{q,0}\eta_{q,1}...\eta_{q,x-1} \leq Vol(ball(q, R_q))\left(\left(\frac{R_q - \tau}{R_q}\right)^m\right)^x$$

Therefore, $\frac{Vol(ball(q,\delta(q,v_x)))}{Vol(ball(q,R_q))} \leq ((\frac{R_q-\tau}{R_q})^m)^x$. Recall that R denotes the largest distance between any two points in D. We have $R_q \leq R + \tau$. Therefore,

$$\frac{Vol(ball(q, \delta(q, v_x)))}{Vol(ball(q, R_q))} \leq \left(\left(\frac{R}{R+\tau}\right)^m\right)^x \tag{7}$$

We perform logarithm operation on both sides of (7) and let $\hat{\eta} = (\frac{R}{R+\tau})^m$ be the base of the logarithm. Since $\hat{\eta} < 1$, we have

$$x \leq \log_{\hat{\eta}} \frac{Vol(ball(q, \delta(q, v_x)))}{Vol(ball(q, R_q))} = \log_{\hat{\eta}} \left(\frac{\delta(q, v_x)}{R_q}\right)^m \leq \log_{\hat{\eta}} \left(\frac{\delta(q, v_x)}{R + \tau}\right)^m = m \log_{\hat{\eta}} \frac{\delta(q, v_x)}{R + \tau}$$

The expectation of x over all possible q is $\mathbb{E}[x] \leq m\mathbb{E}[\log_{\hat{\eta}} \frac{\delta(q,v_x)}{R+\tau}] = m\mathbb{E}[\log_{\hat{\eta}} \delta(q, v_x)] - m\mathbb{E}[\log_{\hat{\eta}}(R + \tau)]$. Since R, $\hat{\eta}$, and τ do not depend on q, it follows that

$$\mathbb{E}[x] \leq \frac{m\mathbb{E}[\ln \delta(q, v_x)]}{\ln \hat{\eta}} - \frac{m \ln(R + \tau)}{\ln \hat{\eta}} = \frac{\mathbb{E}[\ln \delta(q, v_x)] - \ln(R + \tau)}{(1/m) \ln \hat{\eta}}$$
$$= \frac{\mathbb{E}[\ln \delta(q, v_x)] - \ln(R + \tau)}{(1/m)(m \ln R - m \ln(R + \tau))} = \frac{\mathbb{E}[\ln \delta(q, v_x)] - \ln(R + \tau)}{\ln R - \ln(R + \tau)}$$

Let $f(R) = \frac{\mathbb{E}[\ln \delta(q,v_x)] - \ln(R+\tau)}{\ln R - \ln(R+\tau)}$. We have $\mathbb{E}[x] \leq f(R)$. Because $\mathbb{E}[\ln \delta(q, v_x)]$ and τ are independent of R, we can compute the derivative of $f(R)$ with respect to R as $f'(R) =$

$$\frac{\frac{1}{R+\tau}(\ln(R + \tau) - \ln(R))}{(\ln R - \ln(R + \tau))^2} + \frac{(\ln(R + \tau) - \mathbb{E}[\ln \delta(q, v_x)])(\frac{1}{R} - \frac{1}{R+\tau})}{(\ln R - \ln(R + \tau))^2}$$

Since $R + \tau > \delta(q, v_x)$, $f'(R) > 0$. Hence, f is an increasing function.

Recall the assumption that the points in D are uniformly distributed with density g. Therefore, $g \cdot V_D = n$. Based on the assumption $\psi V_D \geq Vol(ball(R))$, $Vol(ball(R)) \leq \psi V_D = \psi \frac{n}{g}$. Since g increases with n, g must be larger than a constant, say g_0. Therefore, $Vol(ball(R)) \leq \psi \frac{n}{g} \leq \psi \frac{n}{g_0}$.

Based on the definition of the volume of ball, $Vol(ball(R)) = \frac{\pi^{m/2} R^m}{\Gamma(1+m/2)}$, where Γ is the Gamma function. Let $b = \frac{\pi^{m/2}}{\Gamma(1+m/2)}$. We have $b \cdot R^m \leq \psi \frac{n}{g}$ and $R \leq (\frac{\psi n}{gb})^{1/m} \leq (\frac{\psi n}{g_0 b})^{1/m}$.

Let $r(n) = (\frac{\psi n}{g_0 b})^{1/m}$. Since f is an increasing function, we have the following.

$$\mathbb{E}[x] \leq f(R) \leq f(r(n)) = \frac{\mathbb{E}[\ln \delta(q, v_x)] - \ln(r(n) + \tau)}{\ln r(n) - \ln(r(n) + \tau)} \tag{8}$$

Since $\delta(q, v_x) \geq \delta(q, \bar{v})$, $\mathbb{E}[\ln \delta(q, v_x)] \geq \mathbb{E}[\ln \delta(q, \bar{v})]$. Since q is in $ball(\bar{v}, \tau)$, we need to analyze two cases. i) If $\tau \geq \frac{\sqrt{m}}{2}(\frac{2}{g})^{1/m}$, *i.e.*, $ball(\bar{v}, \tau)$ contains the hypercube centered at $\bar{v}$ with volume $2/g$, since q is uniformly distributed in the hypercube, $\mathbb{E}[\ln \delta(q, \bar{v})] \geq \ln(\frac{1}{2}(\frac{2}{\ln n})^{1/m})$ (cf. the proof of Theorem 1). ii) If $\tau < \frac{\sqrt{m}}{2}(\frac{2}{g})^{1/m}$, since q is uniformly distributed in $ball(\bar{v}, \tau)$, $\mathbb{E}[\ln \delta(q, \bar{v})] \geq \ln(\tau/2)$. In sum, $\mathbb{E}[\ln \delta(q, \bar{v})] \geq \min(\ln(\tau/2), \ln(\frac{1}{2}(\frac{2}{\ln n})^{1/m}))$.

Since $\mathbb{E}[x] \geq 0$, both the denominator and numerator of (8) are negative. It follows that $\mathbb{E}[x] \leq O(\frac{-\ln(r(n)+\tau)}{\ln r(n) - \ln(r(n)+\tau)})$ as τ and m are constants. According to [17], $\frac{-\ln(r(n)+\tau)}{\ln r(n) - \ln(r(n)+\tau)}$ and $\frac{(r(n)+\tau)\ln(r(n)+\tau)}{\tau}$ have the same order of growth rate when $n \to \infty$. Therefore,

$$\mathbb{E}[x] \leq O\Big(\frac{(r(n) + \tau)\ln(r(n) + \tau)}{\tau}\Big) \tag{9}$$

Since τ is is a constant and independent of n, it follows that

$$\mathbb{E}[x] = O(r(n)\ln r(n)) = O(n^{\frac{1}{m}} \ln n^{\frac{1}{m}}) = O(n^{\frac{1}{m}} \ln n) \tag{10}$$

Since the expected node degree of τ-MG is $O(\ln n)$ (cf. Lemma 3), the time complexity of Algorithm 3 is $O(n^{\frac{1}{m}}(\ln n)^2)$. □

9.6 Proof of Lemma 6

Proof. For PG construction (Line 1), the time complexity is $O(n^{\frac{2+m}{m}} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$. Specifically, if G_0 is NSG, the work [17] presents that the time complexity of NSG construction is $O(n^{\frac{1+m}{m}} \ln n + n \ln n)$. If G_0 is HNSW, the time complexity is $O(n^{\frac{2+m}{m}} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$ by the construction logic of HNSW and Theorem 1.

The time complexity of the search on G_0 (Line 5) is $O(n^{\frac{2}{m}} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$ by Theorem 1. For sorting H_u and processing the nodes in H_u (Lines 6-16), the time complexity is a constant, as h is a constant. Since the for-loop (Lines 3-15) needs to repeat n times, the total time complexity is $O(n^{\frac{2+m}{m}} \ln n)$ with probability at least $1 - (1/e)^{\frac{m}{4}(1-\frac{3}{e^2})}$. □

9.7 Proof of Lemma 7

Proof. Suppose the beam search already enters $H_{\bar{v}}$ and a node v in $H_{\bar{v}}$ is added to W. Since the beam size is larger than $h + h'$, v will not be squeezed out of W during the beam search. Further, all nodes that are reachable from v along monotonic paths will be added to W during the beam search. Based on the definition of τ-MNG, G must have a monotonic path from v to $\bar{v}$. It follows that $\bar{v}$ must be added into W. Therefore, the probability of finding $\bar{v}$ is no smaller than the probability that the beam search enters $H_{\bar{v}}$. □

ACKNOWLEDGMENTS

This work is supported by the NSF of China 62002108, 62202401, 62072132, HKRGC C2004-21GF, 12202221, 12201518, 12201119, NSF of Guangdong Province 2023A1515030273, 2023A1515011655, 2019B1515130001, NSF of Hunan Province 2021JJ40123. This work was done when Yun Peng was a research assistant professor with HKBU.

REFERENCES

[1] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, and Michael Nett. 2015. Estimating Local Intrinsic Dimensionality. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 29–38.

[2] Laurent Amsaleg, Björn Þór Jónsson, and Herwig Lejsek. 2018. Scalability of the NV-tree: Three Experiments. In *Proceedings of the International Conference on Similarity Search and Applications (SISAP)*, Vol. 11223. 59–72.

[3] Anon. 2010. Datasets for approximate nearest neighbor search. Retrieved May 2022 from http://corpus-texmex.irisa.fr/.

[4] Anon. 2011. Million Song Dataset Benchmarks. Retrieved May 2020 from http://www.ifs.tuwien.ac.at/mir/msd/.

[5] Anon. unknown. Common Crawl. Retrieved April 2020 from http://commoncrawl.org/.

[6] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* 87 (2020), 101374.

[7] Franz Aurenhammer. 1991. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *Comput. Surveys* 23, 3 (1991), 345–405.

[8] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. 2018. Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Vol. 11216. 209–224.

[9] Dmitry Baranchuk, Dmitry Persiyanov, Anton Sinitsin, and Artem Babenko. 2019. Learning to route in similarity graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*. 475–484.

[10] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P. Drake, Jane M. Landolin, and Adam M. Phillippy. 2015. Assembling Large Genomes with Single-Molecule Sequencing and Locality-Sensitive Hashing. *Nature Biotechnology* 33, 6 (2015), 623–630.

[11] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When Is "Nearest Neighbor" Meaningful?. In *Proceedings of the International Conference on Database Theory (ICDT)*. 217–235.

[12] Abhinandan Das, Mayur Datar, Ashutosh Garg, and Shyamsundar Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the International Conference on World Wide Web (WWW)*. 271–280.

[13] D.W. Dearholt, N. Gonzales, and G. Kurup. 1988. Monotonic Search Networks For Computer Vision Databases. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, Vol. 2. 548–553.

[14] Wei Dong, Moses Charikar, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the International Conference on World Wide Web (WWW)*. 577–586.

[15] Cong Fu and Deng Cai. 2016. EFANNA : An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph. *CoRR* abs/1609.07228 (2016).

[16] Cong Fu, Changxu Wang, and Deng Cai. 2021. High Dimensional Similarity Search with Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1.

[17] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search with the Navigating Spreading-out Graph. *Proceedings of the VLDB Endowment* 12, 5 (2019), 461–474.

[18] Gylfi Þór Gudmundsson, Björn Þór Jónsson, Laurent Amsaleg, and Michael J. Franklin. 2018. Prototyping a Web-Scale Multimedia Retrieval Service Using Spark. *ACM Transactions on Multimedia Computing, Communications, and Applications* 14, 3s (2018), 65:1–65:24.

[19] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1312–1317.

[20] Ben Harwood and Tom Drummond. 2016. FANNG: Fast Approximate Nearest Neighbour Graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5713–5722.

[21] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. 545–554.

[22] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A Survey on Locality Sensitive Hashing Algorithms and their Applications. *CoRR* abs/2102.08942 (2021).

[23] J.W. Jaromczyk and G.T. Toussaint. 1992. Relative neighborhood graphs and their relatives. *Proc. IEEE* 80, 9 (1992), 1502–1517.

[24] Pennington Jeffrey, Socher Richard, and D. Manning Christopher. 2015. GloVe: Global Vectors for Word Representation. Retrieved May 2022 from http://nlp.stanford.edu/projects/glove/.

[25] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

[26] Zhongming Jin, Debing Zhang, Yao Hu, Shiding Lin, Deng Cai, and Xiaofei He. 2014. Fast and Accurate Hashing Via Iterative Nearest Neighbors Expansion. *IEEE Transactions on Cybernetics* 44, 11 (2014), 2167–2177.

[27] Jon Kleinberg. 2000. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. 163–170.

[28] Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 2130–2137.

[29] Govinda D. Kurup. 1992. *Database Organized on the Basis of Similarities with Applications in Computer Vision.* Ph. D. Dissertation.

[30] Herwig Lejsek, Friðrik Heiðar Ásmundsson, Björn Þór Jónsson, and Laurent Amsaleg. 2009. NV-Tree: An Efficient Disk-Based Index for Approximate Search in Very Large High-Dimensional Collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 5 (2009), 869–883.

[31] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. 2539–2554.

[32] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2020), 1475–1488.

[33] Ting Liu, Andrew Moore, Ke Yang, and Alexander Gray. 2004. An Investigation of Practical Approximate Nearest Neighbor Algorithms. In *Advances in Neural Information Processing Systems*, Vol. 17.

[34] Wanqi Liu, Hanchen Wang, Ying Zhang, Wei Wang, Lu Qin, and Xuemin Lin. 2021. EI-LSH: An early-termination driven I/O efficient incremental c-approximate nearest neighbor search. *VLDB Journal* 30, 2 (2021), 215–235.

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

[36] Y. A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836.

[37] Charles U. Martel and Van Nguyen. 2004. Analyzing Kleinberg's (and other) small-world Models. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, 179–188.

[38] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. 2018. A Survey of Product Quantization. *ITE Transactions on Media Technology and Applications* 6, 1 (2018), 2–10.

[39] Stanley Milgram. 1967. The small world problem. *Psychology Today* 1, 1 (1967), 61–67.

[40] Stanislav Morozov and Artem Babenko. 2018. Non-metric Similarity Graphs for Maximum Inner Product Search. In *Advances in Neural Information Processing Systems*, Vol. 31.

[41] Toshiro Ogita, Hidetomo Ichihashi, Akira Notsu, and Katsuhiro Honda. 2014. Improvement of PCA-Based Approximate Nearest Neighbor Search Using Distance Statistics. *Journal of Advanced Computational Intelligence and Intelligent Informatics* 18, 4 (2014), 658–664.

[42] Marco Patella and Paolo Ciaccia. 2008. The Many Facets of Approximate Similarity Search. In *Proceedings of the First International Workshop on Similarity Search and Applications (SISAP)*. 10–21.

[43] Yun Peng, Byron Choi, Tsz Nam Chan, and Jianliang Xu. 2022. LAN: Learning-based Approximate k-Nearest Neighbor Search in Graph Databases. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 2508–2521.

[44] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

[45] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. 2020. Graph-based Nearest Neighbor Search: From Practice to Theory. In *Proceedings of the International Conference on Machine Learning (ICML)*. 7803–7813.

[46] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. 2020. Graph-based Nearest Neighbor Search: From Practice to Theory. In *Proceedings of the International Conference on Machine Learning (ICML))*, Vol. 119. 7803–7813.

[47] Uri Shaft and Raghu Ramakrishnan. 2005. When Is Nearest Neighbors Indexable?. In *Proceedings of the International Conference on Database Theory (ICDT)*. 158–172.

[48] Larissa C. Shimomura, Rafael Seidi Oyamada, Marcos R. Vieira, and Daniel S. Kaster. 2021. A survey on graph-based methods for similarity searches in metric spaces. *Information Systems* 95 (2021), 101507.

[49] Yifang Sun, Wei Wang, Jianbin Qin, Ying Zhang, and Xuemin Lin. 2014. SRS: Solving c-Approximate Nearest Neighbor Queries in High Dimensional Euclidean Space with a Tiny Index. *Proceedings of the VLDB Endowment* 8, 1 (2014), 1–12.

[50] Javier Vargas Muñoz, Marcos A. Gonçalves, Zanoni Dias, and Ricardo da S. Torres. 2019. Hierarchical Clustering-Based Graphs for Large Scale Approximate Nearest Neighbor Search. *Pattern Recognition (PR)* 96 (2019), 106970.

[51] Hongya Wang, Zhizheng Wang, Wei Wang, Yingyuan Xiao, Zeng Zhao, and Kaixiang Yang. 2020. A Note on Graph-Based Nearest Neighbor Search. arXiv:2012.11083 [cs.LG]

[52] Jingdong Wang, Naiyan Wang, You Jia, Jian Li, Gang Zeng, Hongbin Zha, and Xian-Sheng Hua. 2014. Trinary-Projection Trees for Approximate Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 2 (2014), 388–403.

[53] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 14, 11 (2021), 1964–1978.

[54] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 339 (1998), 440–442.

[55] Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N. Holtmann-Rice, David Simcha, and Felix X. Yu. 2017. Multiscale Quantization for Fast Similarity Search. In *Advances in Neural Information Processing Systems*. 5745–5755.

[56] Zhaozhuo Xu, Weijie Zhao, Shulong Tan, Zhixin Zhou, and Ping Li. 2022. Proximity Graph Maintenance for Fast Online Nearest Neighbor Search. *arXiv* (2022).

[57] Artem Babenko Yandex and Victor Lempitsky. 2016. Efficient Indexing of Billion-Scale Datasets of Deep Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2055–2063.

[58] Yuanhang Yu, Dong Wen†, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2022. GPU-accelerated Proximity Graph Approximate Nearest Neighbor Search and Construction. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 552–564.

[59] Weijie Zhao, Shulong Tan, and Ping Li. 2020. SONG: Approximate Nearest Neighbor Search on GPU. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 1033–1044.

[60] Yuxin Zheng, Qi Guo, Anthony K.H. Tung, and Sai Wu. 2016. LazyLSH: Approximate Nearest Neighbor Search for Multiple Distance Functions with a Single Index. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 2023–2037.

Received July 2022; revised October 2022; accepted November 2022