Sparse Neighborhood Graph-Based Approximate Nearest Neighbor Search Revisited: Theoretical Analysis and Optimization

Xinran Ma
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
maxinran22@mails.ucas.ac.cn

Zhaoqi Zhou
Huawei Technologies Co., Ltd.
zhouzhaoqi1@huawei.com

Chuan Zhou
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
zhouchuan@amss.ac.cn

Zaijiu Shang
Shanghai Institute for Mathematics and Interdisciplinary Sciences
zaijiu@simis.cn

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

Zhiming Ma
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
mazm@amt.ac.cn

ABSTRACT

Graph-based approaches to approximate nearest neighbor search (ANNS) enable fast, high-recall retrieval on billion-scale vector datasets. Among them, the Sparse Neighborhood Graph (SNG) is widely used due to its strong search performance. However, the lack of theoretical understanding of SNG leads to expensive tuning of the truncation parameter that controls graph sparsification. In this work, we present OPT-SNG, a principled framework for analyzing and optimizing SNG construction. We introduce a martingale-based model of the pruning process that characterizes the stochastic evolution of candidate sets during graph construction. Using this framework, we prove that SNG has a maximum out-degree of $O(n^{2/3+\epsilon})$, where $\epsilon > 0$ is an arbitrarily small constant, and an expected search path length of $O(\log n)$. Building on these insights, we derive a closed-form rule for selecting the optimal truncation parameter R, thereby eliminating the need for costly parameter sweeping. Extensive experiments on real-world datasets demonstrate that OPT-SNG achieves an average 5.9× speedup in index construction time, with peak improvements reaching 15.4×, while consistently maintaining or improving search performance.

PVLDB Reference Format:
Xinran Ma, Zhaoqi Zhou, Chuan Zhou, Zaijiu Shang, Guoliang Li, and Zhiming Ma. Sparse Neighborhood Graph-Based Approximate Nearest Neighbor Search Revisited: Theoretical Analysis and Optimization. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/maxila320/OPT-SNG.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Nearest Neighbor Search (NNS) is a fundamental problem in computer science [18, 26, 40, 41, 43, 44] and has become a fundamental research topic across various fields such as information retrieval [42, 48, 49], data mining [14, 53, 55], and large language models [3, 33, 66]. Formally, given a dataset $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ of cardinality n and query point $q \in \mathbb{R}^d$, the NNS problem requires finding the k closest points to q under an appropriate distance metric $\|\cdot\|$. The computational complexity of exact NNS grows prohibitively with both dimensionality d due to the well-documented curse of dimensionality [8, 13, 25, 27, 60] and the expanding dataset size n. This fundamental limitation has motivated the development of Approximate Nearest Neighbor Search (ANNS) algorithms, which achieve significant efficiency improvements while sacrificing minimal accuracy.

Modern ANNS approaches achieve remarkable search performance and fall into four categories: tree-based approaches [6, 54], hashing-based approaches [9, 15, 32, 61], vector quantization-based approaches [19, 22], and graph-based approaches [4, 20, 57, 69]. Among these methods, graph-based approaches have demonstrated particular success in practical applications [10, 69], as they effectively capture and exploit local neighborhood structures to enable navigation in high-dimensional spaces. A crucial observation is that despite the diversity of graph construction strategies—which vary in neighbor selection and edge formation rules—most state-of-the-art graph-based ANNS systems share a unified query processing paradigm. The graph-based ANNS framework operates through two complementary phases: (1) an index construction phase that builds a proximity graph preserving neighborhood relationships through various graph refinement strategies, and (2) a query processing phase that employs the *GreedySearch* algorithm [21, 38, 52] to iteratively traverse the graph along decreasing-distance paths until convergence.

Sparse Neighborhood Graph (SNG) is a widely used variant designed to further reduce graph density while preserving practical search quality [2]. It has been widely adopted in several state-of-the-art systems, including DiskANN [52], HNSW [38] and NSG [21].

The worst-case search analysis of SNG by Indyk et al. [28] provides a framework based on aspect ratios[1]. Complementary to this, a developing and expected cardinality-based framework is both meaningful and necessary for understanding SNG behavior under random or empirical data distributions.

The construction process of SNG typically involves two main phases: (1) Initial graph generation through a random or approximate k-NN graph, and (2) Graph refinement through iterative pruning[64]. The pruning phase is the computational bottleneck, with $O(n)$ worst-case complexity per node, as it requires scanning all candidate neighbors and evaluating geometric conditions that ensure the graph supports fast convergence of the *GreedySearch* algorithm. To reduce index construction time, a common practice in SNG construction is to impose a maximum node degree via a truncation parameter R. However, existing methods typically determine R through parameter sweeping (or a parameter study) [52], commonly implemented as a binary search guided by the downstream search performance of the constructed graph. This tuning process is computationally expensive, as each iteration requires constructing and evaluating a full index. These limitations highlight the need for a theoretically grounded approach to selecting R—one that ensures high search performance while reducing tuning overhead.

To address both the theoretical and practical challenges in SNG-based approximate nearest neighbor search, we propose **OPT-SNG**, a martingale-based probabilistic framework for analyzing and optimizing SNG construction. In this framework, we characterize the SNG construction process as a discrete stochastic process driven by successive pruning operations. By exploiting the martingale structure naturally arising from the evolution of candidate sets, we establish that the expected trajectory can be characterized by an associated ordinary differential equation (ODE). Furthermore, we prove that the out-degree of each node corresponds precisely to the stopping time of this stochastic process. Within this framework, we derive bounds on two fundamental structural properties of SNGs—the maximum out-degree and the expected search path length. Building upon these theoretical results, we further quantify index construction complexity, which serves as a principled foundation for optimizing the truncation parameter R in SNG-based ANNS systems.

We conclude our contributions as follows.

- We introduce OPT-SNG, the first martingale-based probabilistic model of SNG pruning dynamics. By formulating the pruning process as a submartingale with respect to a natural filtration, we transform graph refinement into an analyzable probabilistic system. This enables the application of concentration inequalities and differential equation methods to study graph evolution.
- We establish a degree bound of $O(n^{2/3+\epsilon})$ for any small constant $\epsilon > 0$, substantially improving over the previous coarse $O(n)$ bound from [52]. This result highlights the inherent sparsity of SNGs and demonstrates that graph structure can be controlled through proper pruning dynamics analysis.
- We prove that GreedySearch on SNGs converges in $O(\log n)$ expected steps. This analytical guarantee validates the intuition behind SNG's empirical efficiency and provides theoretical justification.
- Using the above theoretical foundations, we analyze the asymptotic complexity of the truncated SNG construction process and derive a closed-form expression for the optimal truncation parameter R. This yields a principled and automated parameter selection method, avoiding costly parameter sweeping over R. Empirically, our method achieves comparable or superior search performance while reducing index construction time by an average of 5.9× across datasets, with peak speedups reaching 15.4×.

All detailed proofs of lemmas and theorems are provided in the full version [37] of this work on our GitHub repository.

The remainder of this paper is organized as follows. Section 2 reviews the necessary background on SNG-based ANNS, formally defines the problem setting, and introduces the key algorithms considered in this study. In Section 3, we present our probabilistic modeling framework that captures the stochastic dynamics of the pruning process. Sections 4 and 5 derive the main theoretical bounds on graph degree and search path length, respectively. Section 6 describes our optimized construction algorithm and the principled method for truncation parameter selection. Experimental results validating our theoretical predictions and demonstrating practical speedups are reported in Section 7. We survey the related works and position our contributions within the broader literature in Section 8, and Section 9 concludes with a summary and directions for future research.

2 k−ANN PROBLEM AND SPARSE NEIGHBORHOOD GRAPHS

In this section, we formalize the approximate nearest search problem formulation, review existing SNG-based ANNS construction, and summarize the key algorithmic components relevant to our theoretical analysis. Given a dataset $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ of n points in d-dimensional Euclidean space with distance metric $\|\cdot\|$, we consider a query point $q \in \mathbb{R}^d$. See Table 1 for a summary of key notations used throughout.

2.1 Problem Formulation

We formally define the k-approximate nearest neighbor search problem as follows.

Definition 1 (k-Approximate Nearest Neighbor Search)**.** Given a dataset $P \subset \mathbb{R}^d$, a query point $q \in \mathbb{R}^d$, and an integer $k \geq 1$, the k-approximate nearest neighbor search (K-ANNS) problem aims to return a set $A \subseteq P$ of k data points that are closest to q, subject to a desired trade-off between recall and query latency.

Graph-based ANNS methods address this problem by modeling the dataset P as the vertex set of a graph and constructing edges between data points to facilitate efficient search. Typically, the construction process begins with an initial graph, such as a random graph or a k-nearest neighbor graph, and then refines this structure through iterative edge pruning. Formally, these methods construct a graph $G = (P, E)$, where each vertex corresponds to a data point, and the edge set E is designed to preserve neighborhood proximity, enabling fast approximate nearest neighbor search via local graph traversal.

[1]The aspect ratio is the ratio of the maximum to minimum distance between any pair of points in the dataset.

Algorithm 1: GreedySearch(G, q, s, k, L)

Input : graph $G = (P, E)$, query point q, entry point s, k for top-k, search width L
Output : top-k approximate nearest neighbors of q

$Visited \leftarrow \{s\}$;
$Queue \leftarrow$ priority queue initialized with s;
while $|Visited| < L$ **and** $Queue \neq \varnothing$ **do**
  $\hat{p} \leftarrow$ extract-min($Queue$);
  /* closest unexpanded point */
  for each $p' \in N_{\text{out}}(\hat{p})$ **do**
    if $p' \notin Visited$ **then**
      $Visited \leftarrow Visited \cup \{p'\}$;
      insert p' into $Queue$;
return the k points in $Visited$ closest to q

Algorithm 2: SNG-Prune(p, S, α, R)

Input: point p, candidate set S, pruning parameter $\alpha \geq 1$, degree bound R
Output: pruned neighbor set $N_{\text{out}}(p)$ with $|N_{\text{out}}(p)| \leq R$

$N_{\text{out}}(p) \leftarrow \emptyset$;
Sort S in ascending order of $\|p - \cdot\|$;
while $S \neq \emptyset$ **do**
  $p^* \leftarrow \arg\min_{p' \in S} \|p - p'\|$;
  $N_{\text{out}}(p) \leftarrow N_{\text{out}}(p) \cup \{p^*\}$;
  Remove p^* from S;
  if $|N_{\text{out}}(p)| = R$ **then**
    break;
  for each $p' \in S$ **do**
    if $\|p - p'\| \geq \alpha \cdot \|p^* - p'\|$ **then**
      Remove p' from S;
return $N_{\text{out}}(p)$;

2.2 *GreedySearch* Algorithm

Despite differences in index construction, many graph-based ANNS methods share a common query-processing phase: a greedy best-first traversal known as *GreedySearch*. Representative examples include KGraph [17], DiskANN [52], NSG [21], HNSW [38], DPG [35], and SPTAG [12]. The procedure is described in Algorithm 1.

Starting from an entry point s, the algorithm maintains (1) a visited set and (2) a priority queue of candidate nodes ordered by distance to the query q. At each iteration, the closest unexpanded vertex $\hat{p}$ is extracted from the priority queue. Its outgoing neighbors are then explored and added to the priority queue if they have not been visited, and simultaneously, $\hat{p}$ is added to the visited set. The process continues until either the search width reaches the predefined budget L or the priority queue becomes empty. The parameter L controls the search width and governs the recall and latency trade-off: larger values of L explore more candidate nodes, improving recall at the cost of increased search time.

Maintaining and accelerating the convergence of *GreedySearch* is a central objective in graph construction. In particular, ensuring graph connectivity while preserving monotonic search paths is both subtle and essential for guaranteeing fast convergence and high recall. The Sparse Neighborhood Graph (SNG) [2] has been shown to effectively achieve these objectives by iteratively selecting nearest neighbors and pruning redundant candidates [16, 28].

2.3 Sparse Neighborhood Graph Construction

A core operation in SNG construction is the pruning procedure, which selectively retains informative neighbors while aggressively eliminating redundant connections. This procedure is a fundamental building block in several state-of-the-art graph-based ANNS systems, including DiskANN [52], NSG [21], and HNSW [38].

The algorithm proceeds iteratively: in each iteration, it selects the nearest point p^* from the candidate set S, adds p^* to the neighbor set $N_{\text{out}}(p)$, and then prunes candidates from S. Specifically, a candidate p' is removed if it satisfies the blocking condition $\|p - p'\| \geq \alpha \cdot \|p^* - p'\|$. This condition captures the intuition that p' is blocked by the recently selected neighbor p^*—that is, accessing p' through the intermediate neighbor p^* is at least as efficient as a direct connection. Consequently, the direct edge $p \rightarrow p'$ is redundant and can be safely removed without compromising graph navigability.

Two parameters control the pruning behavior:

Pruning Aggressiveness Parameter α: In the non-relaxed setting ($\alpha = 1$), the blocking condition is symmetric and purely geometric. However, to avoid pathological configurations (e.g., collinear points), the Vamana algorithm [52] introduces $\alpha > 1$, which relaxes the blocking condition and ensures bounded graph diameter. In practice, α is typically fixed in the range $1 \leq \alpha \leq 2$ [29] and does not require extensive tuning, as the SNG structure remains robust across this range.

Degree Truncation Parameter R: To control graph density and construction cost, a maximum degree limit R is imposed. The algorithm terminates after exactly R neighbors have been selected, bounding the out-degree of each node. This truncation ensures that construction remains tractable while allowing a trade-off between graph quality and computational efficiency.

A crucial property of SNG-based construction is that it preserves *monotonic searchability* [21]. Specifically, for any point p, there exists at least one monotonic path consisting of edges that lead to progressively closer points, eventually reaching the nearest neighbor of p. This property is essential for the effectiveness of *GreedySearch*, as it guarantees that each expansion step can make consistent progress toward the target.

Despite its empirical success across multiple state-of-the-art systems (DiskANN, NSG, HNSW), the theoretical understanding of SNG construction remains incomplete. The introduction of relaxation parameter α substantially complicates the analysis: the blocking condition becomes asymmetric, and the geometric properties become more intricate. While Indyk et al. [28] provide worst-case analysis based on aspect ratios and Prokhorenkova and Shekhovtsov [16] establish bounds under specific distributional assumptions, a comprehensive expected-case, cardinality-based framework characterizing SNG behavior has been lacking.

In particular, the truncation parameter R remains a critical tuning knob. Existing practice relies on expensive parameter sweeping: practitioners repeatedly construct indices with different R values, evaluate their search performance through binary or grid search,

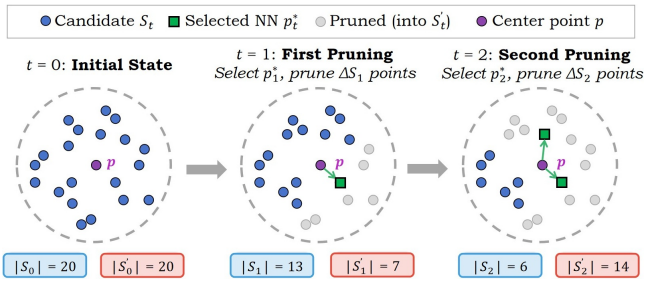


Figure 1: Illustration of the SNG process for a single point.

and select the value that optimizes the recall-latency trade-off. This iterative process is computationally prohibitive for large-scale datasets and obscures the fundamental principles governing the relationship between R, graph structure, and search efficiency.

Our OPT-SNG framework addresses these gaps by providing a principled, theoretically grounded approach to analyzing SNG dynamics and optimizing the truncation parameter R.

3 OPT-SNG PROBABILISTIC FRAMEWORK SETUP

Stochastic Modeling of the Pruning Process. To analyze the construction of SNG from a dynamic perspective, we establish the following probabilistic framework that models the evolution of the pruning process. Consider a fixed point $p \in P$ being processed. Let $t \in \{0, 1, 2, \dots\}$ denote the iteration index, where each iteration is associated with two dynamically evolving sets.

Definition 2 (Candidate Set and Processed Set)**.** At iteration t, the *candidate set* is denoted by S_t. It is initialized as $S_0 = P \setminus \{p\}$, meaning that, initially, the point p is allowed to connect to all other points in the dataset. At each iteration, S_t consists of the points that have not yet been pruned and have not been selected as neighbors. The corresponding *processed set* is defined as $S_t' = P \setminus (S_t \cup \{p\})$, which contains all points that have either been pruned or already selected as nearest neighbors in previous iterations.

In the non-truncated setting, the pruning process terminates when $S_t = \emptyset$, or equivalently when $S_t' = P \setminus \{p\}$. Figure 1 provides a schematic illustration of this construction process. Initially ($t = 0$), all other points belong to the candidate set S_0. At iteration $t = 1$, the nearest neighbor p_1^* is selected, and a subset of points ΔS_1 is pruned and moved into the processed set S_1'. This process repeats at subsequent iterations: at iteration t, a new nearest neighbor p_t^* is selected from S_{t-1}, additional points are pruned, and the candidate set shrinks accordingly.

Due to the randomness of point distributions in high-dimensional space, the number of pruned points at each iteration is itself random, which induces randomness in the resulting graph structure. Consequently, the pruning procedure can be naturally viewed as a stochastic process. The correspondence between iteration indices and the evolution of S_t and S_t' allows us to formalize the construction using tools from probability theory.

Probability Space for SNG construction. We introduce a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω denotes the sample space containing all possible sequences of nearest neighbor selections and pruning outcomes during the overall SNG construction. The σ-algebra $\mathcal{F}$ captures all measurable events on Ω , and $\mathbb{P}$ is the associated probability measure. We further define the *natural filtration* $\{\mathcal{F}_t\}_{t\geq 1}$, representing the information available up to iteration t. Specifically,

$$\mathcal{F}_t = \sigma\left((p_1^*, S_1), (p_2^*, S_2), \dots, (p_t^*, S_t)\right)$$

where $p_i^* = \arg\min_{s\in S_i} \|p - s\|$ is the nearest neighbor selected at iteration i. The candidate set is updated according to the pruning rule $S_{t+1} = S_t \setminus (\{p' \in S_t : \|p - p'\| \geq \alpha \cdot \|p^* - p'\|\} \cup \{p_{t+1}^*\})$ which removes both the selected nearest neighbor and all candidates blocked by it. This formulation captures the pruning process as a sequence of progressively revealed decisions.

Martingale Perspective. We now formalize the dynamic trend of the construction process. Intuitively, at each iteration, the processed set strictly grows while the candidate set shrinks. This monotonic behavior can be rigorously characterized using martingale theory.

Definition 3 (Martingale)**.** A stochastic process $\{X_t\}_{t\geq 0}$ is a *martingale* with respect to a filtration $\{\mathcal{F}_t\}$ if

$$\mathbb{E}[|X_t|] < \infty \quad \textit{and} \quad \mathbb{E}[X_{t+1}|\mathcal{F}_t] = X_t$$

for all $t \geq 0$. It is a *supermartingale* if $\mathbb{E}[X_{t+1}|\mathcal{F}_t] \leq X_t$, and a *submartingale* if $\mathbb{E}[X_{t+1}|\mathcal{F}_t] \geq X_t$.

In the context of SNG construction, the size of the candidate set $|S_t|$ forms a supermartingale, while the size of the processed set $|S_t'|$ forms a submartingale with respect to the natural filtration $\{\mathcal{F}_t\}$. This reflects the fact that, in expectation, candidates are eliminated while processed points accumulate over time. We focus on $|S_t'|$ rather than $|S_t|$ because, as complementary sets, $|S_t'|$'s monotonic growth is suited to our analytical framework.

From the SNG pruning rule and the probability framework, we summarize the following properties of processed sets S_t', which will facilitate the analysis of the construction process. These properties follow directly from our previous results and do not require separate proof.

Lemma 1 (Dynamic Property of S_t')**.** *Given a dataset $P \subset \mathbb{R}^d$ consisting of n points, the SNG construction of $p \in P$ satisfies the properties as follows:*

Table 1: Summary of Notation

Symbol	Description
P	Dataset of n points in $\mathbb{R}^d$
S_t	Candidate set at iteration t
S_t'	Processed set at iteration t: $S_t' = P \setminus (S_t \cup \{p\})$
p_t^*	Nearest neighbor of p in S_t at iteration t
$\mathcal{F}_t$	Natural filtration up to iteration t
ΔS_t	Number of points pruned in iteration t
π_t	Pruning probability at iteration t
R	Truncation parameter (maximum degree limit)
M	Number of processed points at a given level
α	Relaxation parameter
t	Iteration index (time step)

- ***Monotonicity***: *The processed sets form a nested sequence* $\emptyset = S'_0 \subset S'_1 \subset S'_2 \subset \cdots \subset S'_T$, *with a submartingale* $|S'_t|$.
- ***Termination***: *The final processed set includes* $n-1$ *points, excluding only the current point* p.
- ***Degree property***: *Out-degree of* p *equals the number of iterations, as each iteration adds exactly one directed edge from* p *to its nearest neighbor in the current candidate set.*

Based on this framework, we introduce the following metric to measure the progression of graph construction.

Definition 4 (M-t Level)**.** The SNG construction is said to reach *the* M-t *level* at iteration t if t is the smallest iteration that $|S'_t| \geq M$. This represents the *first passage time* at which at least M points have been processed.

The M-t metric serves as a tool to track the progress of graph construction. For a fixed iteration count t, a larger value of M indicates faster pruning and more rapid expansion of the processed set. For a fixed M, a smaller value of t implies that the algorithm has processed M points more quickly.

4 DEGREE BOUNDS OF SNG

In this section, we establish upper bounds on the degree of SNG. As shown in Lemma 1, the out-degree of each point equals the number of pruning iterations during construction, since exactly one edge is added in each iteration. Consequently, bounding the degree reduces to analyzing the dynamics of the pruning process and estimating the total number of iterations required for graph construction. For completeness, all detailed proofs of lemmas and theorems are provided in the full version of this paper on our GitHub repository.

4.1 Two-Phase Behavior: Empirical Motivation

To motivate our theoretical analysis, we first present empirical evidence revealing a distinctive *two-phase behavior* in SNG construction. Figure 2 illustrates two representative cases from the SIFT1M dataset: a fast case completing in only 31 iterations and a slow case requiring over 1000 iterations. Despite this disparity in total iterations, both cases exhibit a consistent structural pattern: an initial phase of rapid candidate reduction, followed by a prolonged plateau phase during which progress slows significantly.

This empirical observation strongly suggests that a phase-wise analysis is both natural and necessary for deriving degree bounds. In the remainder of this section, we formalize this intuition. Specifically, Lemma 4 establishes rapid progress in the initial phase, which in turn enables a degree bound in Theorem 1.

4.2 Pruning Probability Analysis

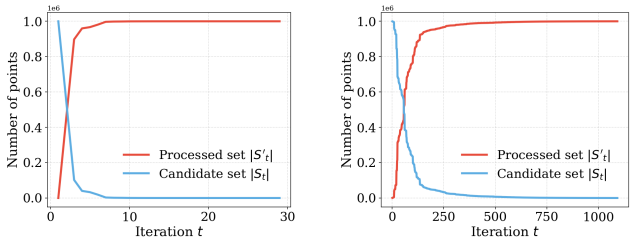
We begin by analyzing the pruning probability at each iteration, which governs the rate at which candidate points are eliminated. This is a key ingredient for quantifying the speed of the initial fast pruning phase.

Lemma 2 (Pruning Probability in SNG Construction)**.** *Given a dataset* $P \subset \mathbb{R}^d$ *consisting of* n *points independently and uniformly distributed within a ball of radius* ρ_0*, centered at* p*, we consider the SNG construction processed on the center point* p *with* $\alpha = 1$*. The probability* π_t *that a point* p' *is pruned in the* $t-th$ *iteration is given by*

$$\pi_t = \frac{I_{1-(\rho_t/\rho_0)^2}\left(\frac{d+1}{2}, \frac{1}{2}\right) - (\rho_t/\rho_0)^d \cdot I_{0.75}\left(\frac{d+1}{2}, \frac{1}{2}\right)}{2\left(1-(\rho_t/\rho_0)^d\right)},$$

where $I_x(a,b)$ *denotes the regularized incomplete Beta function, and* ρ_t *is the distance between the nearest neighbor and* p *in* $t-th$ *iteration. For large* d*, this probability satisfies*

$$\frac{I_{0.75}\left(\frac{d+1}{2}, \frac{1}{2}\right)}{2} < \pi_t < \frac{1}{2}.$$



(a) Fast pruning **(b) Slow pruning**

Figure 2: Evolution of processed and candidate sets during SNG construction. (a) shows a fast pruning case terminating in 31 iterations. (b) shows a slow pruning case requiring over 1,000 iterations.

4.3 Two-Dimensional Case

To build intuition, we first analyze a low-dimensional setting using the M-t level. In particular, for the two-dimensional case, we show that *only four iterations* suffice to prune over 80% of all candidate points with high probability.

Lemma 3 (Two-Dimensional Case)**.** *Given a dataset* $P \subset \mathbb{R}^2$ *with* n *points independently and uniformly distributed within a ball of radius* ρ_0*. Then, with probability* $1 - o(1)$*, the SNG construction process reaches* $(0.8(n-1))$*-4 level.*

Proof. Recall that in the SNG construction of $p \in P$, we iteratively prune a candidate set S_i, starting with $S_0 = P \setminus \{p\}$. At iteration $i = 1, 2, \ldots$, the nearest neighbor is selected from S_{i-1}, and the remaining points are pruned independently with probability π_i.

Let ΔS_i denote the number of points pruned at iteration i. Since the selected nearest neighbor is excluded from pruning, we have $\Delta S_i \sim \mathrm{Bin}(|S_{i-1}| - 1, \pi_i)$. After t iterations, the processed set S'_t consists of all pruned points and the selected nearest neighbors. Therefore, $|S'_t| = \sum_{i=1}^{t}(\Delta S_i + 1)$, $t = 2, 3, \ldots$, with $|S'_1| = \Delta S_1 + 1$, since each iteration contributes exactly one selected nearest neighbor in addition to the pruned points.

From Lemma 2, we know that for $d = 2$, the pruning probability at each iteration satisfies $\pi_i > 1/3$. To estimate the pruning progress, we conservatively assume a uniform lower bound $\pi_{\min} = 1/3$ across all iterations.

The expected number of pruned points at iteration i is $\mathbb{E}[\Delta S_i] = \pi_i(|S_{i-1}| - 1)$, and the variance is $\mathrm{Var}(\Delta S_i) = \pi_i(1-\pi_i)(|S_{i-1}| - 1)$.

Since $\pi_i(1-\pi_i) < 1/3$, the normalized variance can be bounded as

$$\mathrm{Var}\left(\frac{\Delta S_i}{n-1}\right) = \frac{\pi_i(1-\pi_i)(|S_{i-1}|-1)}{(n-1)^2} \le \frac{1}{3(n-1)}. \tag{1}$$

Under the minimal pruning rate assumption, the expected fraction of points pruned at each iteration is at least

$$\pi^1_{\min} = \tfrac{1}{3}, \quad \pi^2_{\min} = \tfrac{2}{9}, \quad \pi^3_{\min} = \tfrac{4}{27}, \quad \pi^4_{\min} = \tfrac{8}{81}.$$

Consequently, the cumulative expected fraction of processed points after four iterations exceeds 0.8.

To ensure that this behavior holds with high probability, we control the deviation of ΔS_i from its expectation by choosing constants $\epsilon_i > 0$ such that

$$\epsilon_1 < \frac{\mathbb{E}[\Delta S_1]}{n-1} - \frac{1}{3}, \quad \epsilon_2 < \frac{\mathbb{E}[\Delta S_2]}{n-1} - \frac{2}{9},$$
$$\epsilon_3 < \frac{\mathbb{E}[\Delta S_3]}{n-1} - \frac{4}{27}, \quad \epsilon_4 < \frac{\mathbb{E}[\Delta S_4]}{n-1} - \frac{8}{81}.$$

Applying Chebyshev's inequality yields, for each iteration i,

$$\Pr\left(\left|\frac{\Delta S_i}{n-1} - \frac{\pi_i(|S_{i-1}|-1)}{n-1}\right| \le \epsilon_i\right) \ge 1 - \frac{\mathrm{Var}\left(\frac{\Delta S_i}{n-1}\right)}{\epsilon_i^2} \tag{2}$$
$$\ge 1 - \frac{1}{3(n-1)^2\epsilon_i^2}. \tag{3}$$

Assuming that all four deviations remain within their respective bounds, the probability that the cumulative number of processed points satisfies $|S'_4| \ge 0.8(n-1)$ is bounded below by

$$\Pr(|S'_4| \ge 0.8(n-1)) \ge \prod_{i=1}^{4}\left(1 - \frac{1}{3(n-1)^2\epsilon_i^2}\right) = 1 - o(1). \tag{4}$$

Therefore, with probability $1-o(1)$, the SNG construction reaches the $(0.8(n-1))$-4 level. □

In high-dimensional settings, the same phenomenon manifests as *sublinear-time progress*: a linear number of points is processed within sublinear iterations. This behavior is formalized in the following lemma.

Lemma 4 (Sublinear Time Progress)**.** *Given a dataset $P \subset \mathbb{R}^d$ with n points distributed uniformly, for all relaxation parameter $\alpha > 0$:*
(1) For any constants $\nu_1, \nu_2 \in (0,1)$ with $\nu_1 > \nu_2$, with probability one, the SNG construction reaches the n^{ν_1}-$O(n^{\nu_2})$ level.
(2) For any constant $\nu \in (0,1)$, with probability one, the SNG construction reaches the $(n - n^{1-\nu})$-$O(n^{\nu})$ level.

Proof. Part (1). Fix constants $\nu_1 > \nu_2$ and define the first-passage time $t_1 := \inf\{t \ge 0 : |S'_t| \ge n^{\nu_1}\}$. We prove that $t_1 = O(n^{\nu_2})$ almost surely, i.e., there exists a constant $C > 0$ such that $\Pr(t_1 \le Cn^{\nu_2}) = 1$. Equivalently, it suffices to show that for any fixed $C > 0$,

$$\Pr\left(|S'_{\lfloor Cn^{\nu_2}\rfloor}| < n^{\nu_1} \text{ i.o.}\right) = 0, \tag{5}$$

where "i.o." (infinitely often) denotes the event

$$\limsup_{n\to\infty}\left\{|S'_{\lfloor Cn^{\nu_2}\rfloor}| < n^{\nu_1}\right\} = \bigcap_{n_0=1}^{\infty}\bigcup_{n\ge n_0}\left\{|S'_{\lfloor Cn^{\nu_2}\rfloor}| < n^{\nu_1}\right\}. \tag{6}$$

Let $t = \lfloor Cn^{\nu_2}\rfloor$. Since $|S'_t| = \sum_{i=1}^{t}(\Delta S_i + 1)$, we obtain the deterministic lower bound $|S'_t| \ge t \cdot \min_{1\le i\le t}(\Delta S_i + 1)$, which implies

$$\Pr(|S'_t| < n^{\nu_1}) \le \Pr\left(\min_{1\le i\le t}\Delta S_i < \frac{n^{\nu_1}-t}{t}\right). \tag{7}$$

Since $\nu_1 > \nu_2$, for sufficiently large n we have $n^{\nu_1} > 2t$, and hence

$$\frac{n^{\nu_1}-t}{t} \ge \frac{1}{2C}\, n^{\nu_1-\nu_2}. \tag{8}$$

Recall that $\Delta S_i \sim \mathrm{Bin}(|S_{i-1}|-1, \pi_i)$ with $\mathbb{E}[\Delta S_i] = \pi_i(|S_{i-1}|-1)$. Moreover, the sampling probability admits the uniform lower bound $\pi_i \ge \frac{1}{2} I_{0.75}\left(\frac{d+1}{2}, \frac{1}{2}\right)$. Applying the Chernoff bound [39], for any $\delta \in (0,1)$,

$$\Pr\big(\Delta S_i \le (1-\delta)\mathbb{E}[\Delta S_i]\big) \le \exp\Big(-\tfrac{\delta^2}{2}\mathbb{E}[\Delta S_i]\Big) \tag{9}$$
$$\le \exp\Big(-\tfrac{\delta^2}{4}\left(|S_{i-1}|-1\right) I_{0.75}\Big(\tfrac{d+1}{2}, \tfrac{1}{2}\Big)\Big). \tag{10}$$

For all $i \le t$ such that $|S_{i-1}| \ge n/2$, we have $\mathbb{E}[\Delta S_i] \ge c_0 n$ for some constant $c_0 > 0$. Choosing $\delta \in (0,1)$ such that

$$(1-\delta)\mathbb{E}[\Delta S_i] \ge \frac{1}{2C}\, n^{\nu_1-\nu_2}, \tag{11}$$

we obtain $\Pr(|S'_t| < n^{\nu_1}) \le \exp(-cn)$ for some constant $c > 0$ and all sufficiently large n. Consequently,

$$\sum_{n=1}^{\infty}\Pr\left(|S'_{\lfloor Cn^{\nu_2}\rfloor}| < n^{\nu_1}\right) < \infty. \tag{12}$$

By the Borel–Cantelli lemma (see Lemma 5 in Appendix),

$$\Pr\left(|S'_{\lfloor Cn^{\nu_2}\rfloor}| < n^{\nu_1} \text{ i.o.}\right) = 0, \tag{13}$$

which completes the proof of Part (i).

Part (2). Let $\nu \in (2/3, 1)$ and define $t_2 := \inf\{t \ge 0 : |S'_t| \ge n - n^{1-\nu}\}$. We show that $t_2 = O(n^{\nu})$ almost surely. Fix $C > 0$ and set $t = \lfloor Cn^{\nu}\rfloor$. As above,

$$\Pr\left(|S'_t| < n - n^{1-\nu}\right) \le \Pr\left(\min_{1\le i\le t}\Delta S_i < \frac{n - n^{1-\nu} - t}{t}\right). \tag{14}$$

For sufficiently large n, we have $t \le n/2$ and thus

$$\frac{n - n^{1-\nu} - t}{t} \ge \frac{1}{2C}\, n^{1-\nu}. \tag{15}$$

Using the same Chernoff bound and the same uniform lower bound on π_i, we again obtain an exponentially small upper bound

$$\Pr\left(|S'_t| < n - n^{1-\nu}\right) \le \exp(-c'n) \tag{16}$$

for some constant $c' > 0$. Therefore,

$$\sum_{n=1}^{\infty}\Pr\left(|S'_{\lfloor Cn^{\nu}\rfloor}| < n - n^{1-\nu}\right) < \infty. \tag{17}$$

Applying the Borel–Cantelli lemma yields

$$\Pr\left(|S'_{\lfloor Cn^{\nu}\rfloor}| < n - n^{1-\nu} \text{ i.o.}\right) = 0, \tag{18}$$

and hence $\Pr(t_2 = O(n^{\nu})) = 1$.

□

4.4 Main Degree Bound

Lemma 4 establishes that the initial phase rapidly processes a large fraction of points. However, prior empirical observations suggest that this rapid progress does not persist throughout the entire construction. The pruning process undergoes a *significant slowdown* after the initial fast pruning phase, often reaching a *plateau phase*.

To analyze this plateau phase, we exploit a key fact: the submartingale $|S'_t|_{t\geq 0}$ has bounded per-step variance along the pruning trajectory. This enables us to apply the Differential Equation Method (DEM) [31, 59, 62], a standard technique for approximating bounded-variance stochastic processes by deterministic trajectories. DEM yields an ordinary differential equation that captures the expected evolution of $|S'_t|$ in the plateau regime. Combining the initial fast-pruning bound from Lemma 4 with the DEM-based characterization of the plateau phase gives a unified view of the pruning dynamics. We now sketch the proof of our main result, Theorem 1. Detailed proofs of all lemmas and theorems are provided in the full version of this work on our GitHub repository.

Theorem 1. *Given a dataset P of n points in d-dimensional space, with probability* 1*, the maximum out-degree in the constructed SNG is bounded by $O(n^{2/3+\epsilon})$ for any small constant $\epsilon > 0$.*

Sketch Proof. As outlined at the beginning of this section, our goal is to show that the total number of iterations T, which corresponds to the degree, satisfies $T = O(n^{2/3+\epsilon})$. We divide the SNG construction into two parts, based on the number of processed points $|S'_t|$:
(1) When $|S_t| < n - n^{1-\nu}$, for some $\nu \in (2/3, 1)$, Lemma 4 ensures that the number of iterations to reach this stage is $t_1 = O(n^{\nu})$ with probability one.
(2) When $|S'_t| \geq n - n^{1-\nu}$, the process is within the plateau phase. Applying DEM, the discrete process S'_t can be approximated by the solution to an ODE. Solving this ODE, the number of iterations required to reach $n-1$ processed points in this phase is $O(n^{1-\nu})$. Adding both phases gives

$$T = O(n^{\nu}) + O(n^{1-\nu}) = O(n^{\nu}), \tag{19}$$

yielding the bound $O(n^{2/3+\epsilon})$ for arbitrarily small $\epsilon > 0$. □

5 SEARCH BOUNDS

We now turn to the analysis of query processing on the constructed SNG. Our goal is to show that, with high probability, *GreedySearch* converges to an approximate nearest neighbor in $O(\log n)$ steps.

Theorem 2. *Given a dataset P with n points in d-dimensional space, with probability* 1*, the SNG search converges in $O(\log n)$ steps.*

Sketch Proof. Let $\{v_0, v_1, \ldots, v_k\}$ denote a greedy search path from the start node $p = v_0$ to the approximate nearest neighbor of the query q. At each step, *GreedySearch* moves to the neighbor closest to q, ensuring that $\|v_i - q\| > \|v_{i+1} - q\|$.

To analyze the expected path length k, we analyze the distribution of neighbors around the destination v_k, they are uniformly distributed in a d-dimensional ball of radius ρ_0. We partition this ball into concentric annular layers and let η_i denote the number of neighbors lying in the i-th layer. Each η_i follows a binomial distribution proportional to the shell volume, and the total number of neighbors is bounded by $\eta = O(n^{2/3+\epsilon})$ from Theorem 1.

Using volume arguments, we upper bound the expected total number of neighbors located within the ball of radius $\|v_k - v_0\|$ by the expected number of dataset points that fall into the same region. Specifically, we show

$$\mathbb{E}[k \cdot \eta] \leq \frac{n(n+1)^{(d-1)/d}\rho_0^{d+1} \cdot E\left[((\rho_0 - \triangle r)/\rho_0)^{d(k-1)}\right]}{\triangle r^{d+1} \cdot d}, \tag{20}$$

where $\triangle r$ is the minimum distinguishable pairwise distance difference among all point triples. Applying Jensen's inequality [39], we define an auxiliary function $g(x)$

$$g(x) := \eta \cdot x - \frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}}{d \cdot \triangle r^{d+1}}\left[-\left(\frac{\rho_0 - \triangle r}{\rho_0}\right)^{d(x-1)} + 2\right], \tag{21}$$

whose root bounds $\mathbb{E}[k]$. We demonstrate $g(k) < 0$, note that $g(x)$ is monotonically increasing, and $g(\log n) > 0$, which together imply that $k < \log n$, hence $\mathbb{E}[k] = O(\log n)$. □

6 TRUNCATION PARAMETER OPTIMIZATION

In the preceding sections, we establish theoretical guarantees for the SNG structure, including sublinear degree growth and logarithmic search complexity. We now leverage these insights to address a practical issue in index construction: how to efficiently select the truncation parameter R, which directly controls the trade-off between construction cost and search performance.

6.1 Complexity Analysis

We summarize the full SNG construction pipeline as implemented in practice, incorporating some widely used optimizations [21, 52]. The process begins by initializing a randomized R-regular graph. For each data point p, a *GreedySearch* procedure is performed from a fixed entry point (e.g., the dataset centroid) to collect an initial candidate set for pruning, with time complexity $T_{\text{search}} = O(R \log n)$. Importantly, in practical implementations the initial candidate set is *not* the entire dataset; instead, it consists only of the points returned by *GreedySearch*. This design significantly reduces construction cost compared to the theoretical baseline that assumes full connectivity. In particular, the size of the initial candidate set satisfies $|S_0| = O(R \log n)$ in expectation. The candidate set is then refined through iterative SNG pruning, where the truncation parameter R bounds the number of pruning iterations. Let $|S_0|$ denote the initial candidate set size; the pruning cost is $T_{\text{prune}} = O(R \cdot |S_0|)$. To further enhance graph connectivity and preserve monotonic searchability, reverse edges are inserted after pruning. These insertions may temporarily violate the degree constraint, triggering additional pruning steps. The amortized cost of reverse-edge handling can be modeled as $T_{\text{reverse}} = O(\alpha \cdot R^3)$, where α is the pruning parameter controlling sparsification aggressiveness.

The computational cost of each stage—*GreedySearch*, SNG pruning, and reverse-edge handling—can be analyzed independently.

Algorithm 3: Optimization of Truncation Parameter R of SNG

Input: test pruning parameter α_1, target pruning parameter α_2, number of data points n
Output: optimized truncation parameter R^*

$R \leftarrow n^{2/3}$; // theoretical upper bound
Build graph index using SNG-Prune(α_1, R);
Compute the average out-degree $\bar{R}$ of the constructed graph;
$K' \leftarrow \frac{\alpha_1^2 \cdot \bar{R}}{\log n}$;
$R^* \leftarrow \frac{K' \cdot \log n}{\alpha_2^2}$;
return R^*

Aggregating these components yields the overall construction complexity, as expressed below.

$$n\big(C_1 \cdot R \log n + b_1 + C_2 \cdot (R^2 \log n/\alpha) + b_2 + C_3 \cdot (\alpha R^3) + b_3\big),$$

where C_i and b_i are implementation-dependent constants.

This expression highlights the competing effects of R and α: increasing R improves graph quality but amplifies both pruning and reverse-edge costs.

6.2 Optimization Principle

As discussed in Section 2.3, in practice, α is typically fixed in the range $1 \le \alpha \le 2$. In contrast, the truncation parameter R introduces an intrinsic trade-off: larger R generally improves recall and robustness of search, but also significantly increases construction cost.

We propose to compute the value of R that achieves optimal efficiency for a given target α. We refer to this approach as the *marginal optimality principle*. Concretely, by differentiating the construction complexity with respect to α and solving for the stationary condition, our theoretical analysis yields the relationship $R^* \propto \frac{\log n}{\alpha^2}$. This relation reveals that the optimal truncation parameter scales logarithmically with the dataset size and inversely with the square of the pruning parameter, with the dimensionality d absorbed into the proportionality constant.

The goal of Algorithm 3 is to determine the dataset-dependent constant of proportionality in the above relationship. We begin with a reference construction using $R = n^{2/3}$ under a test pruning parameter α_1, which incurs much lower cost than parameter sweeping. This choice corresponds to the theoretical upper bound on the SNG degree and closely approximates the structure of a non-truncated graph. We then measure the average out-degree $\bar{R}$ of the constructed graph, which captures the intrinsic sparsity induced by the data distribution. Using this estimate, we directly compute the proportionality constant K' and the optimized truncation parameter R^* for the target pruning parameter α_2 in closed form.

7 EXPERIMENTS

We compare a conventional parameter-sweeping pipeline with our *analytical* parameter determination strategy, and assess both construction efficiency and query performance across different graph-based ANNS algorithms and representative datasets.

Table 2: Summary of Datasets Used in Experiments

Dataset	Cardinality	Dimension	Query Size	Data Type
MSong	992,272	420	200	Audio
DEEP1M	1,000,000	256	1,000	Image
SIFT1M	1,000,000	128	10,000	Image
GIST1M	1,000,000	960	1,000	Image
GloVe	1,193,514	200	1,000	Text
UNIFORM	1,000,000	128	10,000	Synthetic

7.1 Experimental Setup

7.1.1 Datasets. We evaluate our method on six datasets with diverse dimensionalities and application domains, as summarized in Table 2.

The first five datasets are standard benchmarks in the ANNS literature and cover a broad range of real-world applications, including image, audio, and text. **MSong** [7] consists of audio feature vectors for music similarity search. **DEEP1M** [5] and **SIFT1M** [30] are widely used image-retrieval benchmarks based on deep neural features and SIFT descriptors, respectively. **GIST1M** [30] contains high-dimensional GIST descriptors. **GloVe** [46] provides word embeddings for semantic search.

In addition, we include a synthetic dataset, **UNIFORM**, where points are generated uniformly at random in a unit hypercube. This dataset is chosen to stress the construction procedure under an idealized and analytically tractable setting. From an information-theoretic perspective, the uniform distribution attains maximum entropy under bounded support and thus exhibits minimal exploitable density variations. Consequently, the pruning dynamics are typically less favorable than in real-world data, where non-uniform density often accelerates effective neighbor elimination. Importantly, our theoretical bounds (Theorem 1) are proved under precisely this distribution, making UNIFORM a controlled setting for empirically verifying that the proposed optimization remains robust even when geometric shortcuts are scarce.

7.1.2 Performance Metrics. We measure both index-construction efficiency and query behavior using several indicators. **Index Construction Time** measures the end-to-end time to obtain a finalized graph index, which includes both the time spent on selecting construction parameters and the subsequent construction phase.

For query-time evaluation, **Recall@10** measures the fraction of queries for which at least one true nearest neighbor appears in the top-10 retrieved results, formally defined as Recall@k = $|A \cap A'|/|A'|$, where A is the retrieved set and A' is the ground truth with $|A| = |A'| = k$. **Queries Per Second (QPS)** measures throughput as the number of queries processed per second and **Query Latency** reports the average time per query in milliseconds at different recall levels.

7.1.3 Parameter Settings. For each fixed α, we vary the search width L (and analogous search-time controls) to generate recall-latency trade-off. For baseline methods that tune the truncation parameter R via parameter sweeping, we employ a binary-search (or grid-search) strategy guided by a normalized 1:1 weighted loss over query recall and latency, which reflects standard practice in

Table 3: Index Construction Speedup of OPT-SNG over Baseline Methods

Dataset	Speedup over Baseline			Avg
	Vamana	NSG	HNSW	
SIFT1M	7.9×	4.8×	3.4×	5.4×
GIST1M	5.6×	7.1×	4.3×	5.7×
DEEP1M	6.4×	6.4×	4.1×	5.6×
MSong	4.2×	5.3×	2.6×	4.0×
GloVe	8.7×	15.4×	2.2×	8.8×
Average	6.6×	7.8×	3.3×	5.9×

iterative tuning pipelines. In contrast, our approach determines R directly from the analytical relationship derived in Section 6.2.

7.1.4 Computing Environment. All experiments are conducted on a workstation equipped with an Intel Core i7-14700KF CPU (20 cores, 3.4GHz base and up to 5.6GHz boost), 64GB DDR4-3200 memory, and a Samsung 990 EVO Plus 2TB NVMe SSD, running Ubuntu 22.04 LTS. We compile all code with GCC 11.4 and enable `-O3` optimizations.

7.1.5 Compared Algorithms. We evaluate three representative SNG-based ANNS algorithms and apply our optimization principle consistently across them. **Vamana** [52], which underlies DiskANN-style indexing, relies on SNG-like pruning and exposes the parameters α and R that directly control graph sparsification and maximum out-degree. **NSG** [21] constructs a navigable graph with a bounded out-degree constraint and typically requires careful tuning of R to balance construction cost and search quality. **HNSW** [38] builds a hierarchical multi-layer graph, where the construction-time candidate list size (e.g., `efConstruction`) plays a similar role in controlling construction overhead and the resulting graph connectivity. We denote the variants using our proposed parameter optimization strategy as **OPT-DiskANN**, **OPT-NSG**, and **OPT-HNSW**, and refer to their original implementations as **DiskANN**, **NSG**, and **HNSW**, respectively.

7.2 Index Construction Performance

We begin by comparing the end-to-end index construction time, including both parameter tuning and the subsequent graph building. Figure 5 reports results across all datasets, where the stacked bars separate tuning time from construction time. Across the board, the most salient difference is the tuning component: parameter sweeping repeatedly triggers expensive partial or full constructions to probe candidate R values, whereas our method largely removes this iterative overhead by computing R analytically once the necessary reference statistics are obtained. As a result, the total construction time decreases substantially even when the actual build phase remains comparable.

Table 3 summarizes the speedups achieved over binary-search baselines. We observe consistent improvements across datasets and algorithms, ranging from 2.2× to 15.4×, with an average speedup of 5.9×. The gains are particularly pronounced on **GloVe** , where NSG achieves a 15.4× speedup, and on **SIFT1M**, where Vamana achieves 7.9×. These large gains occur because the binary-search pipeline typically requires more iterations to converge to a good truncation level on these datasets, leading to large cumulative tuning costs.

Table 4: Recall@10 comparison at fixed latency targets. Latency thresholds are dataset- and algorithm-specific, chosen to reflect typical operating points.

Dataset	Vamana		NSG		HNSW	
	Binary	OPT	Binary	OPT	Binary	OPT
MSong	99.26	**99.32**	99.79	**99.82**	86.27	**91.85**
DEEP1M	96.77	**98.76**	97.94	**99.33**	98.93	**99.71**
SIFT1M	99.84	**99.90**	99.64	**99.70**	99.92	**99.99**
GIST1M	99.63	**99.76**	96.16	**96.79**	98.20	**99.63**
GloVe	88.91	**89.94**	90.34	**91.83**	92.97	**94.61**
UNIFORM	98.78	**99.62**	93.25	**95.96**	96.53	**97.64**

7.3 Query Performance Evaluation

We next evaluate whether the analytically chosen parameters (`OPT`) preserve query-time behavior compared to the parameters selected by parameter sweeping. We report recall and QPS trends by varying the search width L under each α.

Figures 3 present recall versus QPS curves under different α settings. Across datasets, the curves produced by `OPT` closely track and in several cases slightly improve upon the curves of the original baseline, indicating that removing iterative tuning does not come at the expense of query quality. Intuitively, this behavior is consistent with our derivation: the analytical rule targets the better construction-quality regime that the parameter sweep attempts to locate empirically, but avoids repeatedly paying the cost of probing suboptimal intermediate choices.

To provide a compact comparison under fixed latency budgets, Table 4 summarizes the achieved Recall at representative, dataset and algorithm specific latency targets. Specifically, the latency budgets are set as follows: Vamana—5ms (DEEP1M), 200ms (GIST1M, GloVe, MSong), and 10ms (UNIFORM, SIFT1M); NSG—5ms (MSong), 10ms (GloVe, UNIFORM), 0.15ms (SIFT1M), 5ms (GIST1M), and 2ms (DEEP1M); HNSW—0.5ms (SIFT1M), 5ms (GIST1M), 1ms (DEEP1M), 10ms (UNIFORM), 15ms (GloVe), and 10ms (MSong). Bold entries indicate the better result between Binary Search and `OPT` for each algorithm–dataset pair.

The results show that `OPT` consistently matches or improves Recall@10 across all three algorithms and all datasets. The improvements are most pronounced on DEEP1M, MSong, GIST1M, and GloVe, where analytical parameter selection avoids locally optimal but globally suboptimal truncation choices. Notably, HNSW exhibits the largest recall gains on SIFT1M, while on the challenging UNIFORM dataset, the most significant improvement is observed for NSG.

7.4 Discussion on UNIFORM: A Worst-Case Construction Regime

As shown in Table 2, UNIFORM shares the same scale and dimensionality as SIFT1M, enabling a controlled comparison that isolates the effect of data distribution.

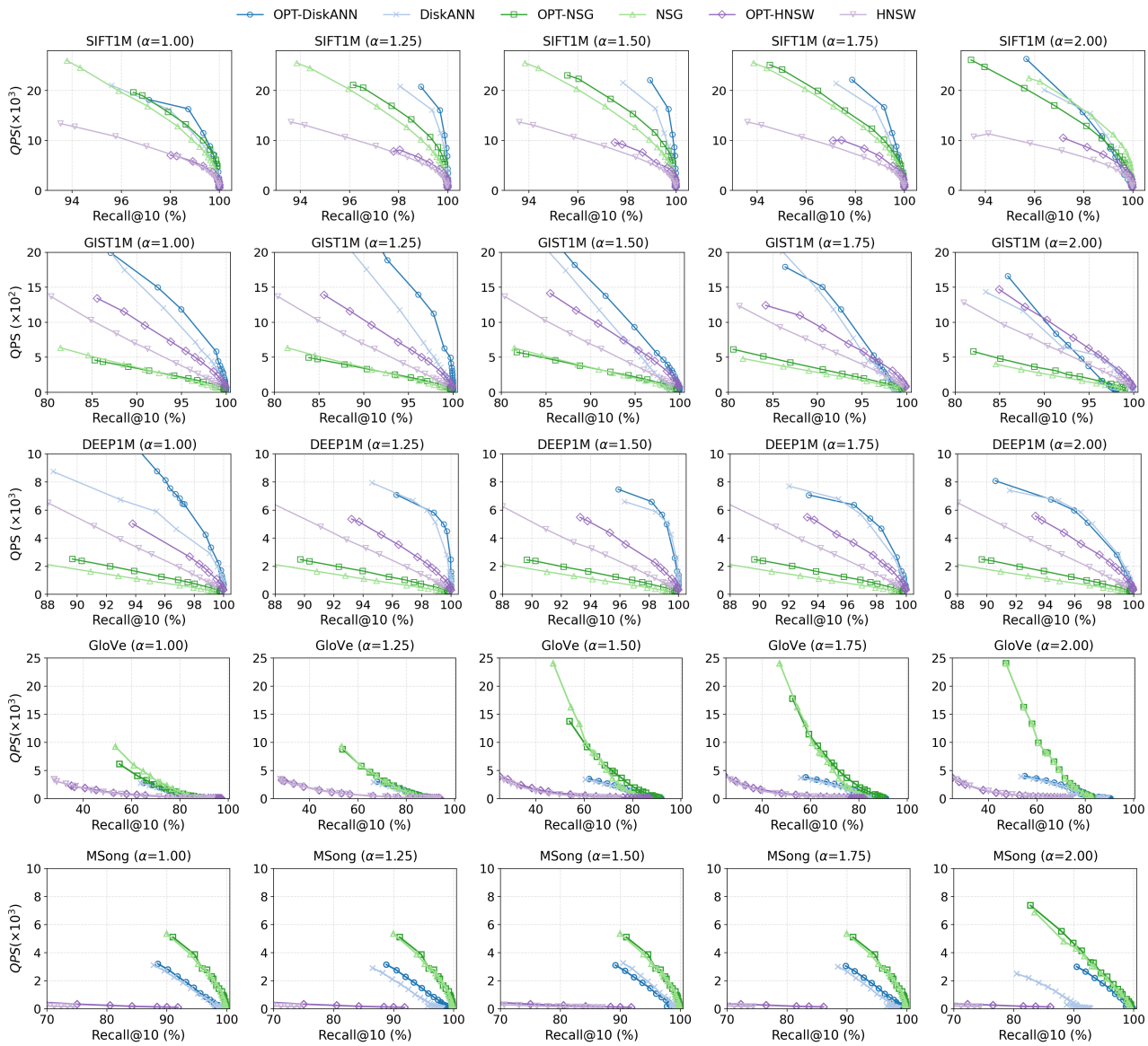


Figure 3: Query performance across varying α values on real-world datasets

Figure 4 shows that the UNIFORM dataset consistently yields lower throughput than SIFT1M across all evaluated algorithms at comparable recall levels. For example, at 99% Recall@10, Vamana on SIFT1M achieves approximately 7× higher QPS than on UNIFORM. Even under this challenging distribution, OPT consistently achieves higher QPS than the baseline at the same recall levels, demonstrating that our optimization remains effective and preserves query efficiency under uniform data without favorable geometric structure.

Figure 6 further demonstrates that UNIFORM produces systematically larger average node degrees than SIFT1M. This behavior precisely characterizes a *worst-case* regime for SNG construction. Our theoretical degree bound of $O(n^{2/3+\epsilon})$ (Theorem 1) is derived under such distributions, since the uniform distribution represents the least favorable case, this bound extends naturally to other distributions with more structured density. Empirically, even without truncation, the maximum degree observed on UNIFORM is 3873, which remains well within the theoretical bound.

These observations are consistent with our theoretical analysis. Uniformly distributed points lack the intrinsic clustering structure commonly present in real-world datasets, which reduces opportunities for aggressive pruning and geometric shortcutting. As a result, the graph construction must introduce **more edges** to capture neighborhood relationships, leading to increased pruning iterations, higher node degrees, and reduced search efficiency.

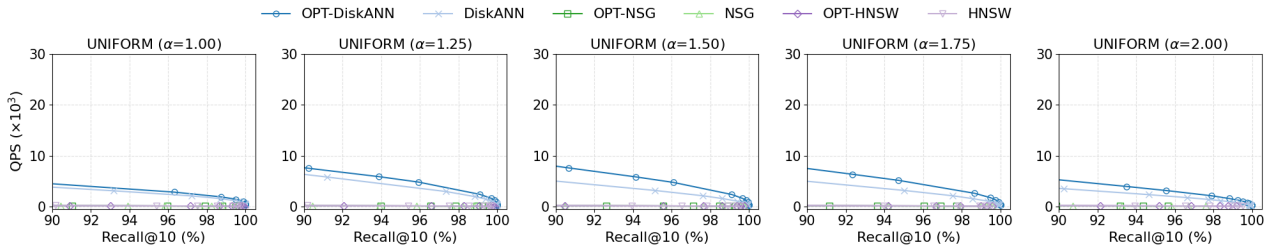


Figure 4: Query performance across varying α values on UNIFORM

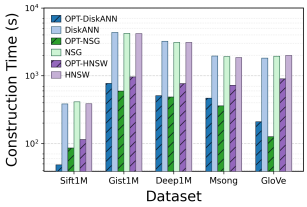


Figure 5: Index construction time comparison

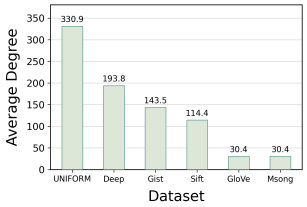


Figure 6: Average out-degree comparison across datasets

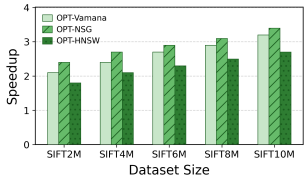


Figure 7: Scalability of index construction speedup on SIFT10M subsets

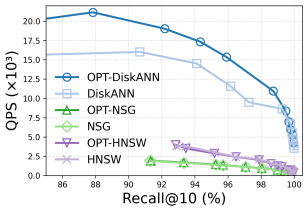


Figure 8: Query performance on SIFT10M

7.5 Effect of Pruning Parameter α

Figure 3 illustrates the recall versus QPS curves under different α settings. Across almost all datasets, increasing the pruning parameter α exhibits a consistent trend: for a fixed recall level, the achievable QPS initially increases as α grows, and then either saturates or slightly degrades when α becomes too large. We observe that $\alpha = 1.50$ provides the best overall balance across datasets, achieving the highest or near-highest QPS at comparable recall levels in most cases.

The sensitivity to α varies across datasets. High-dimensional datasets such as GIST1M and DEEP1M exhibit more pronounced performance degradation at large α values, as overly sparse graphs struggle to preserve navigability in high-dimensional spaces. In contrast, lower- or medium-dimensional datasets such as SIFT1M, GloVe, and MSong remain robust across a wider range of α values, showing only mild performance variation.

7.6 Scalability Study

Finally, we evaluate the scalability of our approach by varying the dataset size and reporting the index construction time. In addition, we conduct experiments on the SIFT10M dataset by sampling subsets of size 2M, 4M, 6M, and 8M points. As shown in Figure 7, the construction speedups achieved by the OPT variants are consistently preserved across all sampled subsets, and the benefits become increasingly pronounced as the dataset size grows.

Furthermore, as illustrated in Figure 8, the recall–QPS curves produced by OPT consistently match or outperform those obtained via parameter sweeping across all three algorithms, without compromising search quality. These results demonstrate that our optimization strategy scales favorably to large datasets and remains effective in both construction efficiency and query performance at scale.

7.7 Summary

The experimental results provide evidence supporting the effectiveness of the proposed analytical parameter optimization strategy. Across all evaluated datasets and SNG-based algorithms, our method substantially reduces end-to-end index construction time by eliminating expensive iterative parameter sweeping. In particular, tuning overhead is reduced by more than an order of magnitude, leading to overall construction speedups of up to 15.4× and an average speedup of 5.9× across datasets.

Crucially, this acceleration does not come at the cost of query quality. The analytically determined parameters consistently preserve—and in many cases improve the recall and QPS compared to baseline parameter sweeping. Recall@10 is matched or improved across all datasets and algorithms at representative latency targets. The results further demonstrate that the proposed method remains robust under the UNIFORM dataset, which represents a worst-case construction regime aligned with our theoretical assumptions.

Scalability experiments on subsets of SIFT10M confirm that the benefits of analytical tuning persist—and become more pronounced—as dataset size increases, while query-time performance remains competitive or superior to parameter sweeping. Taken together, these results validate that our theory-driven approach provides a principled, scalable, and practical alternative to heuristic tuning pipelines for SNG-based ANNS systems.

8 RELATED WORK

8.1 Approximate Nearest Neighbor Search

Over the years, ANNS has evolved to address diverse scenarios, such as multi-metric spaces [68], hybrid queries [45], privacy-preserving federated settings [67], and retrieval-augmented generation [33], which enhances large language models by efficiently querying external knowledge. Methodologically, ANNS techniques are broadly

categorized into tree-based [54, 56], hashing-based [15, 32, 61], quantization-based [19], and graph-based methods [36, 57, 69], with graph-based approaches standing out due to their superior real-world performance in high-dimensional and large-scale datasets [10, 69]. Additionally, hybrid systems that combine graph-based and quantization-based approaches, such as DiskANN [52] and SPANN [10], have been introduced to handle billion-scale datasets efficiently. Tree-based methods, such as k-d trees [6], partition the space hierarchically but suffer from the curse of dimensionality in high-dimensional settings. Hashing-based approaches, exemplified by locality-sensitive hashing (LSH) [15], map similar points to the same bucket with high probability but often require substantial memory overhead. Quantization-based techniques, including product quantization (PQ) [30] and optimized product quantization (OPQ) [23], compress vectors to enable efficient distance computation at the cost of some precision loss.

Among these paradigms, graph-based approaches have emerged as the dominant methodology due to their superior performance on high-dimensional, large-scale datasets [10, 58, 69]. Wang et al. [58] provide a comprehensive survey and experimental comparison of 13 representative graph-based ANNS algorithms, establishing a unified taxonomy based on seed selection, neighbor selection, and search strategies.

8.2 Graph-Based ANNS Framework

Graph-based ANNS methods have gained significant attention in recent years due to their outstanding performance in high-dimensional spaces and their scalability to large datasets. The state-of-the-art graph-based systems, such as HNSW [38], NSG [21], KGraph [17], DiskANN [52], DPG [35], and SPTAG [12] construct graphs with sparse connections to optimize search efficiency. Extensions like Filtered-DiskANN [24] and FreshDiskANN [50] further adapt the framework for different large-scale scenarios. In recent years, graph-based ANNS has also been extended to diverse application settings, including cross-modal retrieval [11], and filtered search [1, 45, 63].

Existing theoretical studies have primarily focused on analyzing fundamental structural properties such as degree bounds, query complexity, and search path length, often under simplified distributional assumptions.

For instance, Laarhoven [47] establishes sublinear query complexity for threshold-based nearest neighbor graphs under spherical data models. In such graphs, two nodes are connected by an edge if and only if the distance between them falls below a specified threshold value, ensuring that each node maintains connections to all sufficiently close neighbors. This structural property enables efficient greedy routing when the graph captures adequate local connectivity.

Along similar lines, Prokhorenkova and Shekhovtsov [16] analyze the degree distribution of navigable graphs constructed under a binomial distribution. A graph is said to be navigable if a greedy search algorithm can successfully find a path from any starting vertex to any target vertex in the dataset without getting trapped in local minima. Their analysis demonstrates that navigable graphs can achieve average degree $O(n^{\beta})$ for $\beta < 1/2$ while preserving the navigability property, revealing a fundamental trade-off between graph sparsity and search reliability.

Beyond degree bounds, another important line of work concerns structural properties that guarantee efficient search convergence. The Monotonic Search Network (MSNET) [21] is a navigable graph with the additional constraint that the search path between any two nodes follows a monotonically decreasing distance trajectory toward the target. This monotonicity property ensures that greedy search makes consistent progress at each step, thereby guaranteeing logarithmic search path length.

More recently, [64] proposes a new construction framework incorporating a self-iterative strategy for edge selection. This approach accelerates RNG construction while preserving competitive k-ANN search performance, demonstrating that careful algorithmic design can significantly reduce preprocessing overhead without sacrificing retrieval quality.

8.3 Parameter Tuning for ANNS Systems

Selecting optimal parameters for ANNS systems remains a practical challenge. Traditional approaches rely on parameter sweeping over the parameter space, which can be computationally expensive due to repeated index construction and evaluation. Recent work on automated tuning, such as VDTuner [65], addresses this challenge through learning-based approaches that predict optimal configurations. Our work takes a complementary approach by deriving analytical expressions for parameter selection based on theoretical insights into the pruning process, thereby eliminating the need for iterative search.

9 CONCLUSIONS

This paper revisits SNG-based graph indexing for approximate nearest neighbor search from a theoretical and optimization perspective. We develop a martingale-based probabilistic framework, OPT-SNG, to model the stochastic pruning dynamics in SNG construction, which enables a principled analysis of the graph structure generated by widely used systems. We establish that the constructed SNG admits a maximum out-degree of $O(n^{2/3+\epsilon})$ (for any $\epsilon > 0$) and that *GreedySearch* converges in an expected $O(\log n)$ number of steps, providing an expected-case, cardinality-based complement to existing analyses.

Building on these guarantees, we further derive a closed-form rule for selecting the truncation parameter R as a function of α and n, and propose an efficient calibration procedure that estimates the scaling constant from a lightweight reference construction. Experiments on multiple real-world benchmarks and a synthetic UNIFORM dataset demonstrate that our analytical parameter selection eliminates expensive parameter sweeping, achieving an average 5.9× reduction in construction time across datasets while maintaining or improving search performance.

We hope this work helps bridge the gap between the empirical success of SNG-based ANNS and its theoretical understanding, and encourages future studies on distribution-aware analyses and principled parameter optimization for large-scale vector search systems.

REFERENCES

[1] Anas Ait Aomar, Karima Echihabi, Marco Arnaboldi, Ioannis Alagiannis, Damien Hilloulin, and Manal Cherkaoui. 2025. RWalks: Random Walks as Attribute Diffusers for Filtered Vector Search. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[2] Sunil Arya and David M. Mount. 1993. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, USA, 271–280.

[3] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based Language Models and Applications. *ACL Tutorial* (2023).

[4] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2023. Elpis: Graph-Based Similarity Search for Scalable Data Science. *Proc. VLDB Endow.* 16, 6 (2023), 1548–1559.

[5] Artem Babenko and Victor S. Lempitsky. 2016. Efficient Indexing of Billion-Scale Datasets of Deep Descriptors. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2055–2063.

[6] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[7] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*. University of Miami, 591–596.

[8] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. 1999. Lower bounds for high dimensional nearest neighbor search and related problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. 312–321.

[9] Deng Cai. 2021. A Revisit of Hashing Algorithms for Approximate Nearest Neighbor Search. *IEEE Trans. Knowl. Data Eng.* 33, 6 (2021), 2337–2348.

[10] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. 2024. RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 17, 11 (2024), 2735–2749.

[11] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X Sean Wang. 2024. Roargraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. *arXiv preprint arXiv:2408.08933* (2024).

[12] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. Spann: Highly-efficient billion-scale approximate nearest neighborhood search. *Advances in Neural Information Processing Systems* 34 (2021), 5199–5212.

[13] Romain Couillet and Zhenyu Liao. 2022. *Random Matrix Methods for Machine Learning*. Cambridge University Press.

[14] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[15] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*. ACM, 253–262.

[16] Haya Diwan, Jinrui Gou, Cameron Musco, Christopher Musco, and Torsten Suel. 2024. Navigable Graphs for High-Dimensional Nearest Neighbor Search: Constructions and Limits. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

[17] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*. 577–586.

[18] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2020. Scalable Machine Learning on High-Dimensional Vectors: From Data Series to Deep Network Embeddings. In *International Conference on Web Intelligence, Mining and Semantics WIMS*. 1–6.

[19] Jingya Fan, Yang Wang, Wenwen Song, and Zhibin Pan. 2024. Flexible product quantization for fast approximate nearest neighbor search. *Multim. Tools Appl.* 83, 18 (2024), 53243–53261.

[20] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 8 (2022), 4139–4150.

[21] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.

[22] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3 (2024), 167.

[23] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* 36, 4 (2013), 744–755.

[24] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, et al. 2023. Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters. In *Proceedings of the ACM Web Conference 2023*. 3406–3416.

[25] Alexander Hinneburg, Charu C Aggarwal, and Daniel A Keim. 2000. What is the nearest neighbor in high dimensional spaces?. In *26th Internat. Conference on Very Large Databases*. 506–515.

[26] Qiang Huang and Anthony Kum Hoe Tung. 2023. Lightweight-Yet-Efficient: Revitalizing Ball-Tree for Point-to-Hyperplane Nearest Neighbor Search. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 436–449.

[27] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[28] Piotr Indyk and Haike Xu. 2023. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. *Advances in Neural Information Processing Systems* 36 (2023), 66239–66256.

[29] Shikhar Jaiswal, Ravishankar Krishnaswamy, Ankit Garg, Harsha Vardhan Simhadri, and Sheshansh Agrawal. 2022. Ood-diskann: Efficient and scalable graph anns for out-of-distribution queries. *arXiv preprint arXiv:2211.12850* (2022).

[30] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[31] Joseph Lauer and Nicholas Wormald. 2007. Large independent sets in regular graphs of large girth. *Journal of Combinatorial Theory, Series B* 97, 6 (2007), 999–1009.

[32] Yifan Lei, Qiang Huang, Mohan S. Kankanhalli, and Anthony K. H. Tung. 2020. Locality-Sensitive Hashing Scheme based on Longest Circular Co-Substring. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2589–2599.

[33] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[34] Shengqiao Li. 2010. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics & Statistics* 4, 1 (2010), 66–70.

[35] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2019), 1475–1488.

[36] Kejing Lu, Chuan Xiao, and Yoshiharu Ishikawa. 2024. Probabilistic Routing for Graph-Based Approximate Nearest Neighbor Search. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

[37] Xinran Ma, Zhaoqi Zhou, Chuan Zhou, Qi Meng, Zaijiu Shang, Guoliang Li, and Zhiming Ma. 2025. Graph-Based Approximate Nearest Neighbor Search Revisited: Theoretical Analysis and Optimization. *arXiv preprint arXiv:2509.15531* (2025). arXiv:2509.15531 [cs.DS]

[38] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[39] Michael Mitzenmacher and Eli Upfal. 2017. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press. 66–75 pages.

[40] Cun Matthew Mu, Jun Raymond Zhao, Guang Yang, Binwei Yang, and Zheng John Yan. 2019. Fast and Exact Nearest Neighbor Search in Hamming Space on Full-Text Search Engines. In *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings (Lecture Notes in Computer Science)*, Vol. 11807. Springer, 49–56.

[41] Riley Murray, James Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. 2023. Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software. arXiv:2302.11474

[42] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based News Recommendation for Millions of Users. In *SIGKDD*. ACM, 1933–1942.

[43] Themis Palpanas. 2015. Data Series Management: The Road to Big Sequence Analytics. *SIGMOD Record* (2015).

[44] Themis Palpanas and Volker Beckmann. 48(3), 2019. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGREC* (48(3), 2019).

[45] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proc. ACM Manag. Data* 2, 3 (2024), 120.

[46] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 1532–1543.

[47] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. 2020. Graph-based Nearest Neighbor Search: From Practice to Theory. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 7803–7813.

[48] M. M. Mahabubur Rahman and Jelena Tesic. 2022. Evaluating Hybrid Approximate Nearest Neighbor Indexing and Search (HANNIS) for High-dimensional Image Feature Search. In *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*. IEEE, 6802–6804.

[49] Susav Shrestha, Narasimha Reddy, and Zongwang Li. 2024. ESPN: Memory-Efficient Multi-vector Information Retrieval. In *Proceedings of the 2024 ACM SIGPLAN International Symposium on Memory Management, ISMM 2024, Copenhagen, Denmark, 25 June 2024*. ACM, 95–107.

[50] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *CoRR* abs/2105.09613 (2021). arXiv:2105.09613 https://arxiv.org/abs/2105.09613

[51] Sunil Srinivasa and Martin Haenggi. 2010. Distance Distributions in Finite Uniformly Random Networks: Theory and Applications. *IEEE Trans. Veh. Technol.* 59, 2 (2010), 940–949.

[52] Suhas Jayaram Subramanya, Devvrit Fnu, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

[53] Yukihiro Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 455–464.

[54] Yufei Tao, Ke Yi, Cheng Sheng, and Panos Kalnis. 2009. Quality and efficiency in high dimensional nearest neighbor search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*. ACM, 563–576.

[55] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1064–1072.

[56] 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 Trans. Pattern Anal. Mach. Intell.* 36, 2 (2014), 388–403.

[57] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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

[59] Lutz Warnke. 2019. On Wormald's differential equation method. *arXiv preprint arXiv:1905.08928* (2019).

[60] Roger Weber, Hans-Jörg Schek, and Stephen Blott. 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, Vol. 98. 194–205.

[61] Jiuqi Wei, Botao Peng, Xiaodong Lee, and Themis Palpanas. 2024. DET-LSH: A Locality-Sensitive Hashing Scheme with Dynamic Encoding Tree for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 17, 9 (2024), 2241–2254.

[62] Nicholas Wormald. 1995. Differential equations for random processes and random graphs. *The annals of applied probability* (1995), 1217–1235.

[63] Wei Wu, Junlin He, Yu Qiao, Guoheng Fu, Li Liu, and Jin Yu. 2022. HQANN: Efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4580–4584.

[64] Shuo Yang, Jiadong Xie, Yingfan Liu, Jeffrey Xu Yu, Xiyue Gao, Qianru Wang, Yanguo Peng, and Jiangtao Cui. 2025. Revisiting the Index Construction of Proximity Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 18, 6 (2025), 1825–1838.

[65] Tiannuo Yang, Wen Hu, Wangqi Peng, Yusen Li, Jianguo Li, Gang Wang, and Xiaoguang Liu. 2024. Vdtuner: Automated performance tuning for vector data management systems. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 4357–4369.

[66] Qiang Yue, Xiaoliang Xu, Yuxiang Wang, Yikun Tao, and Xuliyuan Luo. 2024. Routing-Guided Learned Product Quantization for Graph-Based Approximate Nearest Neighbor Search. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4870–4883.

[67] Xinyi Zhang, Qichen Wang, Cheng Xu, Yun Peng, and Jianliang Xu. 2024. FedKNN: Secure Federated k-Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 1 (2024), V2mod011:1–V2mod011:26.

[68] Yifan Zhu, Lu Chen, Yunjun Gao, Ruiyao Ma, Baihua Zheng, and Jingwen Zhao. 2024. HJG: An Effective Hierarchical Joint Graph for ANNS in Multi-Metric Spaces. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4275–4287.

[69] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 1 (2024), 69:1–69:26.

A SUPPLEMENTARY LEMMAS

We first introduce a commonly used lemma in probability theory, which determines the probability of the limsup of events by analyzing the convergence of the series.

Lemma 5 (Borel–Cantelli lemma). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, where: Ω is the set of all possible outcomes, $\mathcal{F}$ is a sigma-algebra that contains all the information about events, and $\mathbb{P}$ is the associated probability measure. Given a sequence of events $A_1, A_2, \ldots \in \mathcal{F}$ such that $\sum_{n=1}^{\infty} \mathbb{P}(A_n) < \infty$, it follows that:*

$$\mathbb{P}\left(\bigcap_{N=1}^{\infty}\left(\bigcup_{n\geq N} A_n\right)\right) = 0.$$

Here, $\bigcap_{N=1}^{\infty}\left(\bigcup_{n\geq N} A_n\right)$ is also commonly denoted as $\limsup A_n$.

The following Chernoff bound provides a probabilistic guarantee on the deviation of a binomial random variable from its expected value.

Lemma 6 (Chernoff Bound of binomial variable). *Let $X \sim \mathrm{Bin}(n, p)$ and let $\mu = \mathbb{E}[X]$. For any $0 < \delta < 1$:*

$$\mathbb{P}(X \leq (1-\delta)\mu) \leq \exp\left(-\frac{\delta^2\mu}{2}\right).$$

The frame of Wormald's differential equation method (DEM) was developed by Wormald in 1990s [62] as a powerful tool for analyzing the discrete-time randomized graph processes and algorithms. Given a discrete-time stochastic process, in particular, consider a submartingale, if the one-step differences are bounded and the expected differences are well-approximated by a Lipschitz function, it follows that the process closely aligns with the corresponding differential equation with high probability.

Lemma 7 (Differential Equation method [62]). *Given integers $n \geq 1$, a bounded domain $\mathcal{D} \subseteq \mathbb{R}^{a+1}$, a function $F : \mathcal{D} \to \mathbb{R}$, and a sequence of sigma algebras $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots$. Define random variables $Y(i)$ such that $Y(i)$ is $\mathcal{F}_i$-measurable for $i \geq 0$.*

Assume that for all $i \geq 0$, when $(i/n, Y(i)/n) \in \mathcal{D}$, the following conditions are satisfied. Additionally, let $\mathcal{D}$ contain the closure of the set $\{(0, z_0) : \mathbb{P}(Y(0) = z_0 n) \neq 0\}$ for some n, meaning that $\mathcal{D}$ includes all possible starting points.

If $Y(t)$ is a submartingale and the following three conditions hold:
***(1) (Boundedness)** For some $\beta = \beta(n) \geq 1$ and $\gamma = \gamma(n)$, the probability that $|Y(i+1) - Y(i)| \leq \beta$ given $\mathcal{F}_i$ is at least $1-\gamma$. This ensures that the change in $Y(i)$ over one step is controlled.*
***(2) (Trend)** For some $\lambda_1 = \lambda_1(n) = o(1)$, the gap between the conditional expectation of the change and the normalized value of the function F is insignificant:*

$$\left|\mathbb{E}(Y(i+1) - Y(i) \mid \mathcal{F}_i) - F\left(\frac{i}{n}, \frac{Y(i)}{n}\right)\right| \leq \lambda_1.$$

***(3) (Lipschitz)** F is continuous and Lipschitz with constant L on D.*
Then we have:
(a) For $(0, \hat{z}_0) \in \mathcal{D}$, the differential equation

$$\frac{dz}{dx} = F(x, z)$$

has a unique solution in $\mathcal{D}$ with the initial condition $z(0) = \hat{z}_0$. This solution can be extended close to the boundary of $\mathcal{D}$.
Let $\lambda > \lambda_1 + C_0 n\gamma$, where $\lambda = o(1)$. Then, with probability

$$1 - O\left(n\gamma + \frac{\beta}{\lambda}\exp\left(-\frac{n\lambda^3}{\beta^3}\right)\right),$$

the following holds:

$$Y(i) = nz\left(\frac{i}{n}\right) + O(\lambda n),$$

where $z(x)$ is the solution from part (a) with the initial condition $\hat{z}_0 = \frac{1}{n}Y(0)$. This result is valid uniformly for $0 \leq i \leq \sigma n$, where $\sigma = \sigma(n)$ is the supremum of x such that the solution $z(x)$ remains within $\mathcal{D}$.

We will also use Jensen's inequality in the proof. It establishes a fundamental relationship between the expectation of a convex transformation of a random variable and the transformation of its expectation.

Lemma 8 (Jensen's Inequality). *Suppose g is convex and X and $g(X)$ are both integrable. Then*

$$g(\mathbb{E}X) \leq \mathbb{E}g(X)$$

B OMITTED PROOFS

B.1 Proof of Lemma 2

The following is the complete proof of Lemma 2.

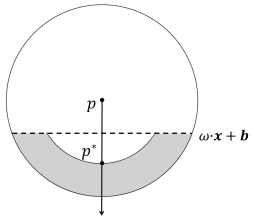


Figure 9: Given that p^* is the nearest point to p in t-th iteration and $\|p - p^*\| = \rho_t$. The light grey region represents all p' satisfied pruning condition $\|p - p'\| > \|p^* - p'\|$.

Proof of Lemma 2. Assuming a uniform distribution over the search space, the probability that a randomly sampled point falls within a region is proportional to the region's volume. Without loss of generality, we place the center point p at the origin and align the vector $\overrightarrow{pp^*}$ with the last coordinate axis, as shown in Figure 9. Under this coordinate system, the nearest neighbor p^* lies at the position $(0, 0, \ldots, 0, \rho_t) \in \mathbb{R}^d$. Let p^* denote the nearest point to p in t-th iteration and $\|p - p^*\| = \rho_t$, then the hyperplane is defined by:

$$\omega \cdot x + b = 0, \quad \text{where } \omega = (0, \ldots, 0, 1), \quad b = -\frac{\rho_t}{2}.$$

The pruned region at iteration t, denoted by $D(t)$, is defined as the intersection of the ball $B(p, \rho_0)$ and the half-space $\omega \cdot x > \frac{\rho_t}{2}$. Let $V(\cdot)$ denote the Lebesgue volume in the appropriate dimension. For example, $V(B_d(r))$ is the volume of a d-dimensional ball of radius r, and $V(B_{d-1}(r))$ is the volume of a $(d-1)$-dimensional

ball of radius r. In particular, $V(B_d(1))$ denotes the volume of the d-dimensional unit ball:

$$V(B_d(1)) = \frac{\pi^{d/2}}{\Gamma(1+d/2)}.$$

To compute $V(D(t))$, we slice the ball orthogonally along the last coordinate x_d, and integrate over the $(d-1)$-dimensional cross-sectional volumes. The volume of the pruned region is given by:

$$V(D(t)) = \int_{x_d=\frac{\rho_t}{2}}^{\rho_0} V\left(B_{d-1}\left(\sqrt{\rho_0^2 - x_d^2}\right)\right) dx_d,$$

where $B_{d-1}(\sqrt{\rho_0^2 - x_d^2})$ is the $(d-1)$-dimensional ball of radius $\sqrt{\rho_0^2 - x_d^2}$, corresponding to the horizontal slice at height x_d. Using the change of variables $t = x_d/\rho_0$, we have $x_d = \rho_0 t$ and $dx_d = \rho_0\, dt$. Substituting this into the integral yields:

$$V(D(t)) = V(B_{d-1}(1)) \cdot \rho_0^d \int_{\frac{\rho_t}{2\rho_0}}^{1} (1-t^2)^{\frac{d-1}{2}}\, dt.$$

Next, we apply the substitution $z = 1 - t^2$ to convert the integral into a standard form. Under this change of variables, we obtain:

$$\int_{\frac{\rho_t}{2\rho_0}}^{1} (1-t^2)^{\frac{d-1}{2}}\, dt = \frac{1}{2}\int_0^{1-\left(\frac{\rho_t}{2\rho_0}\right)^2} z^{\frac{d-1}{2}}(1-z)^{-1/2}\, dz.$$

This integral corresponds to the incomplete Beta function. After normalization, it becomes proportional to the *regularized incomplete Beta function*, also known as the cumulative distribution function of the Beta distribution, denoted by $I_x(a,b)$. Thus, we have:

$$\int_{\frac{\rho_t}{2\rho_0}}^{1} (1-t^2)^{\frac{d-1}{2}}\, dt = \frac{\Gamma\left(\frac{d+1}{2}\right)\Gamma\left(\frac{1}{2}\right)}{2\Gamma\left(\frac{d}{2}+1\right)} \cdot I_{1-\left(\frac{\rho_t}{2\rho_0}\right)^2}\left(\frac{d+1}{2}, \frac{1}{2}\right),$$

where $\Gamma(\cdot)$ denotes the Gamma function. Hence, the volume of the pruned region is:

$$V(D(t)) = \frac{\Gamma(\frac{d+1}{2})\Gamma(\frac{1}{2})}{2\Gamma(\frac{d}{2}+1)} V(B_{d-1}(1)) \cdot \rho_0^d \cdot I_{1-(\frac{\rho_t}{2\rho_0})^2}\left(\frac{d+1}{2}, \frac{1}{2}\right).$$

However, since this includes points within the inner ball $B(\rho_t)$, and we subtract the volume within $B(\rho_t)$, denoted V_{inner}, is:

$$V_{\text{inner}} = \frac{\Gamma(\frac{d+1}{2})\Gamma(\frac{1}{2})}{2\Gamma(\frac{d}{2}+1)} V(B_{d-1}(1)) \cdot \rho_t^d \cdot I_{3/4}\left(\frac{d+1}{2}, \frac{1}{2}\right),$$

where $3/4 = 1 - (1/2)^2$ accounts for the halfway-positioned hyperplane. The probability that a randomly sampled point is pruned in iteration t is:

$$\pi_t = \frac{V(D(t)) - V_{\text{inner}}}{V(B_d(\rho_0)) - V(B_d(\rho_t))}.$$

Substituting the expressions above and simplifying using $V(B_d(1)) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$, we get:

$$\pi_t = \frac{I_{1-(\frac{\rho_t}{2\rho_0})^2} - (\frac{\rho_t}{\rho_0})^d \cdot I_{3/4}}{2\left(1 - (\frac{\rho_t}{\rho_0})^d\right)}.$$

Since $\frac{\rho_t}{\rho_0} \in (0,1)$ and $1 - (\frac{\rho_t}{2\rho_0})^2 \in (3/4, 1)$, the ratio is always decreases with respect to $\frac{\rho_t}{\rho_0}$. Specifically, noting that $I_{3/4}(a,b) < 1$, for large d, we derive:

$$\frac{1}{2} I_{3/4}\left(\frac{d+1}{2}, \frac{1}{2}\right) < \pi_t < \frac{1}{2}.$$

2-dimensional case. In the two-dimensional case, we can leverage geometric relationships to derive a lower bound of $\frac{1}{3}$ for the pruning probability. This bound corresponds to the scenario where the pruning probability decreases as ρ_t/ρ_0 increases, reaching its minimum when $\rho_t/\rho_0 = 1$—that is, when the distance from the center point p to its nearest neighbor p^* approaches the radius of the disk. In this situation, the pruning decision boundary can be approximated by the perpendicular bisector of the segment connecting p and p^*. Geometrically, this bisector divides the circular region into two sectors, and the prunable region corresponds to the sector opposite p. As the angle subtended by this region exceeds $\frac{2\pi}{3}$, it follows that the fraction of the total area being pruned is at least $\frac{1}{3}$. Thus, the pruning probability is bounded below by $\frac{1}{3}$ in the two-dimensional setting. □

B.2 Conservative Probability Estimation

Applicability of Pruning Probability to General Points. Our probability calculations are specifically designed for the scenarios where the reference point is located at the center of the sphere, ensuring uniform density in all directions. Under the assumption of a uniform distribution, each point in the dataset can be locally approximated as having uniform density in its neighborhood. Consequently, the pruning probability derived in Lemma 2 extends naturally to arbitrary points in the dataset, justifying its applicability throughout the SNG construction process.

Overlap. We also note that in later iterations, the pruning region at step t may overlap with those from earlier steps. This geometric overlap does not invalidate our degree bound—in fact, it tends to have faster elimination of candidates and thus requires fewer steps (i.e., fewer edges). In the proof of Lemma 2, any point that lies within the overlapping regions of multiple pruning areas remains eligible for removal, effectively undergoing repeated pruning. Therefore, our earlier volume-based estimate, which assumes disjoint pruning regions and ignores such overlaps, is conservative.

B.3 Proof of Theorem 1

Proof of Theorem 1. Let T denote the total number of iterations. Recall from Lemma 1 (3) that the maximum out-degree is T. Then our goal is to show $T = O(n^{2/3+\epsilon})$ with probability $1 - o(1)$. To analyze this, we divide the construction process into two phases based on the number of processed points $|S'_t|$. Specifically, we define the first phase as $t \le t_1$, during which $|S'_t| < n - n^{1-\nu}$, and the second phase as $t_1 < t \le T$, where $|S'_t| \ge n - n^{1-\nu}$, for any $\nu \in (2/3, 1)$.

First Phase ($t \le t_1$): From Lemma 4 (part ii), for any $\nu \in (0,1)$, with probability one, the SNG construction reaches the $(n - n^{1-\nu})$-$O(n^\nu)$ level. Thus, the number of iterations t_1 to process $n - n^{1-\nu}$ points is:

$$t_1 = O(n^\nu).$$

Second Phase ($t_1 < t \le T$): When $|S'_t| \ge n - n^{1-\nu}$, the remaining points are $|S_t| = n - |S'_t| \le n^{1-\nu}$. We denote the number of iterations

in the second phase as $T_2 = T - t_1$, and aim to estimate how long it takes to complete the construction, i.e., to reach $|S'_T| = n - 1$.

Since at each step at least one point (the nearest neighbor) is added to the processed set, we trivially have $|S_i| \leq n^{1-\nu}$. However, to obtain a tighter estimate, we analyze the dynamics more precisely.

During this actual plateau phase, most iterations do not satisfy the pruning condition, and only remove the nearest neighbor. Significant pruning occurs in only a small fraction of steps. Intuitively, this is because the candidate set becomes small and the pruning probability π_t decays with it. Applying Markov's inequality, we can bound the probability that more than k points are pruned at iteration t:

$$\Pr(\Delta S_i \geq k) \leq \frac{\mathbb{E}[\Delta S_i]}{k} = \frac{\pi_i(|S_i| - 1)}{k} \leq \frac{n\pi_i}{k} = O(1),$$

for constant $k \in \mathbb{N}$. This justifies that $\pi_t = O(1/n)$ in this phase. The update rule for the processed set is:

$$|S'_{t+1}| - |S'_t| = \Delta S_{t+1} + 1, \quad E[|S'_{t+1}| - |S'_t| \mid \mathcal{F}_t] = 1 + \pi_t(|S_t| - 1),$$

where $\mathcal{F}_t$ is the natural filtration up to iteration t (defined in Section 3). Let $z(t/n) := |S'_t|/n$ be the normalized process, and approximate the expected increment by:

$$f(t, z) = 1 + \pi_t \cdot n(1 - z), \quad \text{with} \quad \pi_t \cdot n = O(1).$$

Thus, f is Lipschitz in z, with constant $\theta = O(1)$, satisfying:

$$|f(t, z_1) - f(t, z_2)| \leq \theta |z_1 - z_2|.$$

This verifies the Lipschitz condition for the application of Lemma 7. The corresponding differential equation is:

$$\frac{dz}{dx} = 1 + \theta(1 - z), \quad z(0) = 1 - n^{-\nu},$$

whose solution is:

$$z(x) = 1 + \frac{1}{\theta} - \left(\frac{1}{\theta} + n^{-\nu}\right) e^{-\theta x}.$$

To determine when $|S'_t| = n - 1$, set $z(t/n) = 1 - \frac{1}{n}$ and solve:

$$1 - \frac{1}{n} = 1 + \frac{1}{\theta} - \left(\frac{1}{\theta} + n^{-\nu}\right) e^{-\theta \cdot t/n}.$$

which leads to:

$$e^{-\theta \cdot t/n} = \frac{\frac{1}{\theta} + \frac{1}{n}}{\frac{1}{\theta} + n^{-\nu}}, \quad \Rightarrow \quad t = \frac{n}{\theta} \ln\left(1 + \frac{n^{-\nu} - \frac{1}{n}}{\frac{1}{\theta} + \frac{1}{n}}\right).$$

Using the approximation $\ln(1 + x) \approx x$ for small x, we conclude:

$$t \approx \frac{n}{\theta} \cdot \frac{n^{-\nu} - \frac{1}{n}}{\frac{1}{\theta} + \frac{1}{n}} = O(n^{1-\nu}).$$

Thus:

$$T_2 = O(n^{1-\nu}).$$

Lemma 7 guarantees that this approximation holds with high probability. Specifically, for $\lambda = O(n^{-m})$, where $m \in (0, 3\nu - 2)$, the probability is:

$$1 - O\left(\frac{n^{1-\nu} + 1}{n^{-m}} \exp\{-n^{-3m-2+3\nu}\}\right) \to 1 \text{ as } n \to \infty.$$

Total Iterations: Combining both phases:

$$T = t_1 + T_2 = O(n^{\nu}) + O(n^{1-\nu}) = O(n^{\nu}),$$

for $\nu \in (2/3, 1)$, thus for any $\epsilon > 0$, we conclude:

$$T = O(n^{2/3+\epsilon}).$$

with probability 1.

Case of $\alpha > 1$. When $\|p - p'\| > \alpha \cdot \|p^* - p'\|$, the pruning region is no longer a simple intersection of a hyperplane and a sphere, but in fact corresponds to the intersection of two spheres. The resulting volume is given by:

$$\frac{1}{2} \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2} + 1\right)} r^n I_{1-h^2/(\rho_0)^2}\left(\frac{n+1}{2}, \frac{1}{2}\right),$$

where ρ_0 is the radius of the ball, h is the distance between centers, and $I_x(a, b)$ denotes the regularized incomplete Beta function, with the detailed derivation provided in [34]. Due to the computability of this volume, the pruning probability can be updated and **remains bounded**. The proof technique can be adapted accordingly, thus preserving the conclusions regarding the degree bound. □

B.4 Full Proof to Theorem 2

Proof of Theorem 2. We analyze the expected length k of a *GreedySearch* path from p to the query v_k $\{p = v_0, v_1, \ldots, v_k\}$, where each node selects its neighbor closest to the query point. Let η denote the out-degree (number of neighbors) of v_k, and assume that points in P are independently and uniformly distributed in a d-dimensional ball of radius ρ_0.

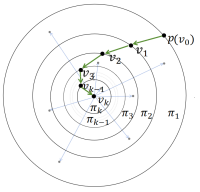


Figure 10: Path and neighbor distribution diagram: Blue lines represent edges from v_k to its neighbors, and green lines represent the search path. π_i denotes the number of neighbors in the i-th annular layer from the outermost to the innermost.

Define $\triangle r$ as the minimum distinguishable pairwise distance difference among all point triples:

$$\triangle r = \min\{|\|a - b\| - \|a - c\||, |\|a - b\| - \|b - c\||, |\|a - c\| - \|b - c\|| \mid a, b, c \in P, \text{distinct}\}$$

as introduced in [21]. Let v_k be the final node of the search path. We construct concentric balls centered at v_k with radii $\|v_k - v_i\|$ for $i = 0, 1, ..., k - 1$, and define the annular layer $A_i = B(v_k, \|v_k - v_{i-1}\|) \setminus B(v_k, \|v_k - v_i\|)$. Let η_i be the number of neighbors in each layer A_i. Then $\eta_i \sim \text{Bin}\left(\eta, \frac{V(A_i)}{V(B(v_k, \rho_0))}\right)$, and the expected number in each layer is proportional to the volume difference:

$$\mathbb{E}[\eta_i] = \eta \cdot \frac{V(B(v_k, \|v_k - v_{i-1}\|)) - V(B(v_k, \|v_k - v_i\|))}{V(B(v_k, \rho_0))}.$$

Summing over all k layers, we have:

$$E\left[\sum_{i=1}^{k}\eta\cdot\frac{V(B(v_k,\|v_k-v_{i-1}\|))-V(B(v_k,\|v_k-v_i\|))}{V(B(v_k,R))}\right] \tag{22}$$

$$\leq n\cdot\frac{V(B(v_k,\|p-v_k\|))}{V(B(v_k,R))}. \tag{23}$$

This upper bound is based on the fact that all points in the ball may serve as neighbors. Using $V(B(v\_k,r))\propto r^d$, we obtain:

$$E\left[\sum_{i=1}^{k}\eta\cdot\frac{\|v_k-v_{i-1}\|^d-\|v_k-v_i\|^d}{\rho_0^d}\right]\leq n\cdot E\left[\frac{\|p-v_k\|^d}{\rho_0^d}\right]. \tag{24}$$

Now, note that

$$\|v_k-v_{i-1}\|^d-\|v_k-v_i\|^d \tag{25}$$

$$=(\|v_k-v_{i-1}\|-\|v_k-v_i\|)\sum_{j=0}^{d-1}\|v_k-v_{i-1}\|^j\|v_k-v_i\|^{d-1-j}. \tag{26}$$

Since $\|v_k-v_{i-1}\|-\|v_k-v_i\|\geq\triangle r$ and $\|v_k-v_i\|\geq\|v_k-v_{k-1}\|$ for all i, we have:

$$\|v_k-v_{i-1}\|^d-\|v_k-v_i\|^d\geq d\cdot\triangle r\cdot\|v_k-v_{k-1}\|^{d-1}.$$

Substituting into Equation 24, we obtain:

$$\mathbb{E}[k\cdot\eta\cdot d\cdot\triangle r\cdot\|v_k-v_{k-1}\|^{d-1}]\leq n\cdot\mathbb{E}[\|p-v_k\|^d].$$

Assuming independence between path length and distances, we rearrange:

$$E[k\cdot\eta]\leq n\cdot\frac{E[\|p-v_k\|^d]}{\triangle r\cdot d\cdot E[\|v_k-v_{k-1}\|^{d-1}]}. \tag{27}$$

Next, we upper bound $\|p-v_k\|$. Since the search path is monotonic:

$$\|p-v_k\|=\|v_0-v_k\|\leq\|v_0-v_k\|\cdot\prod_{i=1}^{k-1}\frac{\|v_i-v_k\|}{\|v_{i-1}-v_k\|}\cdot\frac{\|p-v_k\|}{\|v_{k-1}-v_k\|}.$$

Because $\frac{\|v_i-v_k\|}{\|v_{i-1}-v_k\|}\leq\frac{\rho_0-\triangle r}{\rho_0}$ and $\frac{\|p-v_k\|}{\|v_{k-1}-v_k\|}\leq\frac{\rho_0}{\triangle r}$, we have:

$$\|p-v_k\|^d\leq\frac{\rho_0^{2d}}{\triangle r^d}\cdot\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d(k-1)}.$$

Substituting this into Equation 27, we obtain:

$$\mathbb{E}[k\cdot\eta]\leq\frac{n\cdot\rho_0^{2d}\cdot\mathbb{E}\left[\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d(k-1)}\right]}{d\cdot\triangle r^{d+1}\cdot\mathbb{E}[\|v_k-v_{k-1}\|^{d-1}]}.$$

Using the bound $\mathbb{E}[\|v_k-v_{k-1}\|]\geq\frac{\rho_0}{(n+1)^{1/d}}$ from [51] and Jensen's inequality(Lemma 8), we find:

$$\mathbb{E}[\|v_k-v_{k-1}\|^{d-1}]\geq\left(\frac{\rho_0}{(n+1)^{1/d}}\right)^{d-1}.$$

Thus:

$$E[k\cdot\eta]\leq\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}\cdot E\left[\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d\cdot(k-1)}\right]}{\triangle r^{d+1}\cdot d}$$

$$\leq\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}\cdot E\left[-\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d\cdot(k-1)}+2\right]}{\triangle r^{d+1}\cdot d}.$$

Applying Jensen's inequality again:

$$E[k\cdot\eta]\leq\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}\cdot(-\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d\cdot\mathbb{E}[k-1]}+2)}{\triangle r^{d+1}\cdot d}$$

Define the function:

$$g(x):=\eta\cdot x-\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}}{d\cdot\triangle r^{d+1}}\left[-\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d(x-1)}+2\right].$$

We have $E(k)<0$. Evaluating:

$$g(0)<0,$$

$$g'(x)=\eta+\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}}{\triangle r^{d+1}\cdot d}\cdot d\cdot\ln\left(\frac{R}{R-\triangle r}\right)\cdot\left(\frac{R-\triangle r}{R}\right)^{d\cdot x}>0,$$

$$g(\log n)=\eta\log n+\frac{n(n+1)^{(d-1)/d}\rho_0^{d+1}}{\triangle r^{d+1}\cdot d}\left[\left(\frac{R-\triangle r}{R}\right)^{d\cdot(\log n-1)}-2\right].$$

we have $\left(\frac{\rho_0-\triangle r}{\rho_0}\right)^{d\cdot\log n}=e^{d\cdot\log n\cdot\ln\frac{\rho_0-\triangle r}{\rho_0}}=n^{C'\cdot d\cdot\ln\frac{\rho_0-\triangle r}{\rho_0}}>2.$ Thus:

$$g(\log n)>0.$$

Since $g(x)$ is monotonically increasing and transitions from negative to positive, we conclude that $g(\log n)>0$ and $g(E(k))<0$. Hence,$E(k)<\log n$, leading to:

$$E[k]=O(\log n).$$

□

C COMPLEXITY ANALYSIS DETAILS

Below is a detailed complexity analysis of each component in the SNG construction algorithm. **Complexity of GreedySearch.** The GreedySearch algorithm iteratively selects the closest neighbor to the query and follows a path to an approximate nearest neighbor, evaluating all neighbors at each step. According to [21], the complexity of GreedySearch is determined by the product of the average path length and the average degree. Theorem 2 establishes that the path length is $O(\log n)$. Thus, for a graph with degree K, the search complexity is $O(K\cdot\log n)$.

Complexity of Pruning. In the SNG pruning algorithm, the outer loop checks if the candidate set S is empty. If not, it adds an edge to the current nearest neighbor. The inner loop evaluates all remaining points in S, determining whether to prune them based on the pruning rule. In the non-truncated SNG pruning process, the outer loop iterates at most $O(n^{2/3+\epsilon})$ times, as established by Theorem 1. The inner loop evaluates all points in the candidate set S, excluding the nearest neighbor, checking up to $n-2$ points per iteration. This yields a complexity upper bound of $O(n^{5/3+\epsilon})$, tighter than the $O(n^2)$ estimate in [52]. For the truncated variant, SNG-Prune, the outer loop runs at most R times. The inner loop evaluates up to $n-2$ points, resulting in a worst-case complexity of $O(R\cdot n)$.

Total Complexity of Construction od SNG. To begin with, the graph is randomly initialized as an R-regular graph. R is the maximum degree specified by the parameter sweep. To reduce the indexing time, for each point p in the dataset, GreedySearch is first performed starting from the centroid s to obtain the set of volunteer points, which are then used as candidate points for pruning, the complexity can be approximate by $O(R\cdot logn)$ and

written as $C_1 \cdot R \log n + b_1$ [21]. Next, we apply SNG-tPrune to prune the randomized graph. This process runs for at most R iterations, and in each iteration, the number of points to be checked is approximately $O(R \log n)$. Considering the parameter α, as α increases, the graph has higher degree, hence the complexity can be regarded as inversely proportional to α. Therefore, the algorithm's complexity is estimated to be $O(R \cdot R \log n/\alpha)$ and can be expressed as $C_2(R \cdot R \log n/\alpha) + b_2$. To enhance GreedySearch convergence during the search phase, we add reverse edges for each neighbor of point p. This may cause a degree overrun beyond R. In such cases, SNG-tPrune is reapplied. For each point p with at most R possible neighbors, we add reverse edges and reconstruct the graph. The candidate set size does not exceed $R + 1$, and the inner loop iterates at most R times. A larger α increases the number of neighbors, raising the probability of a degree overrun, which we model as a proportional relationship, so the complexity is expressed as $C_3(\alpha \cdot R \cdot R^2) + b_3$.

D DISCUSSION

Point Processes and Uniformity: A New Perspective on Assumptions. Dataset point distributions can often be modeled by a *spatial point process*, which is a random process used to describe the distribution of points in a given space. Formally, a spatial point process is a measurable mapping from a probability space into the space of locally finite point configurations in $\mathbb{R}^d$.