Enhancing Graph-based Approximate Maximum Inner Product Search via Norm-Adaptive Partitioning

XI ZHAO, The Hong Kong University of Science and Technology, Hong Kong SAR, China
ZHOUJIN TIAN, The Hong Kong University of Science and Technology, Hong Kong SAR, China
KAI HUANG*, East China Normal University, China
YAO TIAN, The Hong Kong University of Science and Technology, Hong Kong SAR, China
XIAOKUI XIAO, National University of Singapore, Singapore
BOLONG ZHENG, Wuhan University of Technology, China
XIAOFANG ZHOU, The Hong Kong University of Science and Technology, Hong Kong SAR, China

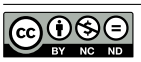
The rapid development of vector databases and large language models has significantly increased the importance of *Approximate Maximum Inner Product Search (AMIPS)* problem. Over the past decade, various approaches have been proposed to efficiently address the AMIPS problems. Compared to other methods, such as those based on Locality Sensitive Hashing (LSH), proximity graph-based methods have shown superior query performance on AMIPS. However, the performance of graph-based methods for AMIPS is still not fully optimized due to the norm bias issue, while graph-based methods have demonstrated strong effectiveness for ANNS. In this paper, we theoretically analyze the norm bias in MIPS and identify a *norm domination* phenomenon: results are consistently dominated by a handful of large-norm vectors, with negligible contributions from small-norm ones. Building on this observation, we propose the Norm-Adaptive Partitioning (NAP) scheme, which splits the vectors in a dataset into Head, Body, and Tail based on vector norms. The Head contains few large-norm vectors for exhaustive search; the Tail includes small-norm ones that can be safely pruned under accuracy bounds; and the Body, characterized by concentrated norms, is well-suited for conventional proximity graphs. The main challenge of NAP strategy is to balance the query accuracy and extra cost for NAP by efficiently determining the size of Tail, Head, and Body. To address it, we further design a practical NAP algorithm that minimizes search cost while ensuring bounded accuracy. To demonstrate NAP, we introduce a hybrid index combining a lightweight structure (e.g., a hash table) for the Head and a proximity graph for the Body. Experimental results show that NAP reduces the index size of existing proximity graphs by over 50% and the NAP-based hybrid index enables more than 2x speedup over state-of-the-art graph-based AMIPS methods at the same recall level.

CCS Concepts: • **Information systems → Top-k retrieval in databases**.

Additional Key Words and Phrases: Maximum Inner Product Search, Graph based Method

*Corresponding author.

Authors' Contact Information: Xi Zhao, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong SAR, China, xzhaoca@cse.ust.hk; Zhoujin Tian, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong SAR, China, ztianaf@cse.ust.hk; Kai Huang, East China Normal University, Shanghai, China, khuang@dase.ecnu.edu.cn; Yao Tian, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong SAR, China, ytianbc@cse.ust.hk; Xiaokui Xiao, National University of Singapore, Singapore, Singapore, xkxiao@nus.edu.sg; Bolong Zheng, Wuhan University of Technology, Wuhan, China, zblchris@gmail.com; Xiaofang Zhou, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong SAR, China, zxf@cse.ust.hk.



ACM 2836-6573/2026/6-ART174
https://doi.org/10.1145/3802051

ACM Reference Format:
Xi Zhao, Zhoujin Tian, Kai Huang, Yao Tian, Xiaokui Xiao, Bolong Zheng, and Xiaofang Zhou. 2026. Enhancing Graph-based Approximate Maximum Inner Product Search via Norm-Adaptive Partitioning. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 174 (June 2026), 26 pages. https://doi.org/10.1145/3802051

1 Introduction

The ***Maximum Inner Product Search (MIPS)*** problem seeks to retrieve, from a dataset of high-dimensional vectors, the vector (*i.e.,* an object representation) that maximizes its inner product with a given query vector. MIPS serves as a core primitive in many large-scale data management and retrieval systems, including vector databases such as Milvus [43] and PgVector [21], information retrieval systems [14, 20] (e.g., Google's REALM [7]), as well as industrial recommendation systems [3, 5, 26]. In recommendation workloads, MIPS underpins collaborative filtering models, where users and items are embedded into a latent space and recommendation reduces to identifying the item vector with the largest inner product to a user query [3]. Empirical evidence from large-scale production systems further highlights the importance of the inner-product-based retrieval; for example, Microsoft Research reports that inner product is a central operation in their Xbox recommendation system and outperforms nearest-neighbor-search (*NNS*, *i.e.,* retrieving the vector with the smallest Euclidean distance to a query) based alternatives in terms of accuracy [6]. Despite its importance, computing exact MIPS is expensive at scale. As both dataset cardinality and vector dimensionality increase, exhaustive computation becomes impractical due to "curse of dimensionality" [2, 10, 18, 24]. This challenge has motivated extensive research on the ***Approximate Maximum Inner Product Search (AMIPS)***, which is widely adopted in modern systems to significantly reduce query latency and resource consumption while delivering high-quality approximate results.

Existing solutions to AMIPS problem mainly consist of locality-sensitive hashing (LSH)-based methods [4, 17, 18] and graph-based approaches [16, 22, 29]. LSH-based methods are renowned for their solid theoretical guarantees and low indexing costs. By projecting high-dimensional vectors into low-dimensional hash spaces using a set of independent LSH functions, high-quality candidates for the query q can be identified with high probability by inspecting only the points in the corresponding hash buckets [4, 18]. However, LSH-based methods suffer from the *hash-boundary issue* [17, 40, 46], where points that are close in original space may be mapped into distinct hash buckets, thereby reducing query accuracy.

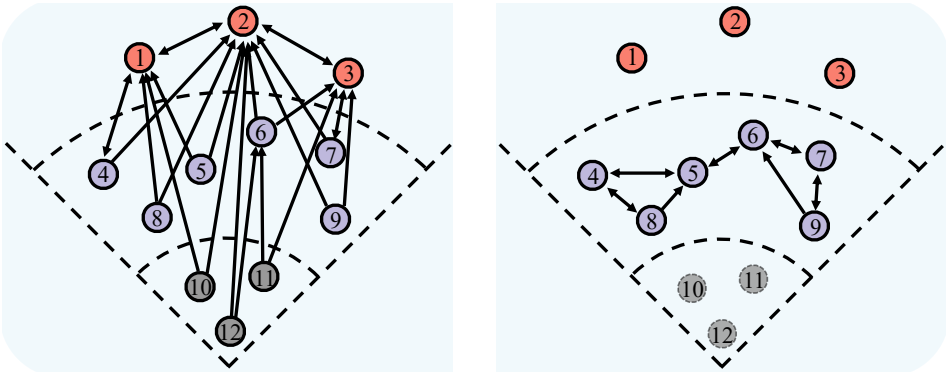
In contrast, graph-based methods have gained widespread adoption for AMIPS due to their superior query efficiency and retrieval accuracy. These methods first construct an **approximate proximity graph (APG)** over a vector dataset by connecting pairs of points (*i.e.,* vectors) with high inner products [11, 30, 37]. To perform a search, the algorithm starts from an entry point and traverses the graph greedily. At each step, it computes the inner products between the query and all neighbors of the current node, then moves to the neighbor with the largest inner product. This process repeats until a stopping criterion is met—typically when no "closer" neighbor (*i.e.,* one with a higher inner product) can be found. Despite outperforming LSH-based methods in query speed and accuracy, graph-based methods exhibit notable performance bottlenecks. Our evaluation on datasets like MNIST (Figure 12 and 13) shows their index size and efficiency remain suboptimal.

This naturally raises a critical question: *can we pinpoint and overcome the fundamental reason for the inadequate performance of graph-based methods in AMIPS?* We answer this affirmatively by uncovering a key insight—the norm domination phenomenon, where search results are overwhelmingly monopolized by a few large-norm vectors, while small-norm vectors contribute negligibly. This phenomenon fundamentally hinders proximity graphs, which are highly effective for approximate nearest neighbor search (ANNS) [16, 29, 46], from achieving comparable performance on

AMIPS. To understand this limitation, consider a dataset whose vectors have nearly uniform norms. In this case, maximizing the inner product is equivalent to minimizing the Euclidean distance, and graph-based methods such as APG can naturally perform AMIPS as effectively as ANNS. However, real-world datasets typically exhibit a wide range of norm distributions, leading to strong norm bias: points with extremely large or small norms distort the neighbor structure, degrading graph connectivity and search accuracy. Consider nodes x_1 and x_5 (*i.e.,* ① and ⑤) in an APG depicted in Figure 1(a) where their norms $r_1 = \|x_1\|$ and $r_5 = \|x_5\|$ satisfies $r_1 > r_5$. For any query q on the APG, it is apparent that x_1 is more likely to yield a larger inner product with q than x_5, making the other points (*e.g.,* x_5) hard to be found during a greedy search.

To tackle this bottleneck, we propose the Norm-Adaptive Partitioning (NAP) framework, which manages data points with varied norms and enables efficient and accurate AMIPS search. NAP strategically partitions the dataset into three norm-based segments— Head, Body, and Tail—in descending order of norms. The Head contains a small number of large-norm vectors that are highly likely to appear in MIPS results. Since they are few but critical, we search them exhaustively using lightweight methods such as linear scan or LSH, rather than indexing them in a proximity graph. The Tail consists of small-norm vectors that almost never appear in MIPS results and can be safely pruned to save computation and space. The Body , which forms the majority, has moderately varied norms and thus serves as the ideal candidate set for proximity graph construction. Building an APG on the Body notably alleviates the adverse effect of norm domination and preserves high search efficiency. Figure 1(b) depicts the APG for Body after applying NAP, where nodes x_4–x_9 share a similar norm. As a result, queries executed on the APG are free from norm bias. The main challenge of NAP strategy is to balance the query accuracy and extra cost for NAP by efficiently determining the size of Tail, Head, and Body. To address it, we further design a practical NAP algorithm that minimizes search cost while ensuring bounded accuracy. The Elbow method is employed to determine a suitable size for Head, owing to its low computational overhead. Finally, we demonstrate the effectiveness of NAP through a hybrid index that integrates a lightweight hash table for the Head and a proximity graph for the Body. Experimental results demonstrate that NAP reduces the index size of existing proximity graphs by over 50%, and that the NAP-based hybrid index achieves more than 2x speedup over state-of-the-art graph-based AMIPS methods to reach the same recall. In summary, this paper makes the following contributions.

- Critical Analysis of Norm Domination. We conduct a critical analysis of the limitations inherent in graph-based methods for Maximum Inner Product Search (MIPS), identifying the norm domination phenomenon and clarifying its detrimental impact on both the distribution of MIPS results and the query performance of proximity graphs.
- Novel Partitioning Strategy. We introduce a novel Norm-Adaptive Partitioning (NAP) strategy, which segments the dataset into three distinct regions—Head, Body, and Tail—based on vector norms, effectively countering the adverse effects of norm bias. Accompanying this strategy, we develop a practical partitioning algorithm that maximizes the identification of Tail points while rigorously bounding accuracy loss within a predefined error threshold.
- Customized Hybrid Index. We design hybrid index structures customized to the characteristics of the Head and Body partitions, leveraging their respective sizes and norm distributions to enable efficient and accurate retrieval of the most relevant MIPS candidates.
- Significant Performance Gains. We conduct extensive experiments on real-world datasets to validate the effectiveness of NAP. The results demonstrate that NAP reduces the query time and memory overhead of state-of-the-art graph-based methods by more than 50% when the dataset exhibits a wide norm distribution.



(a) APG for a whole dataset (b) APG for Body following NAP

Fig. 1. The effect of NAP for APGs in inner product spaces.

The rest of the paper is organized as follows. Section 2 reviews related work on MIPS problem. Section 3 presents the problem definition and background. Section 4 provides an in-depth analysis of the norm domination phenomenon across datasets. Section 5 details the NAP strategy. Section 6 describes NAP-based index structure and the corresponding AMIPS query processing algorithm. Section 7 provides complexity analysis. Section 8 reports extensive experimental evaluations, and Section 9 concludes the paper.

2 Related Work

The AMIPS problem has attracted extensive attention and has been widely studied. Existing AMIPS methods can be categorized into three main groups: LSH-based methods [6, 23, 25, 34, 35, 45, 47], graph-based methods [11, 27, 30, 36–38], and other methods such as those based on learning, sampling, and quantization [13, 15, 33].

2.1 LSH-based methods

The foundation of LSH-based methods lies in LSH functions, which ensure that the collision probability of two points increases with their similarity in the metric space (*i.e.,* it decreases with their distance). However, LSH functions are not directly applicable in inner product spaces, as a point does not necessarily have the highest inner product with itself, despite always exhibiting the highest collision probability with itself. To address this inherent limitation, transformation-based techniques have been proposed to convert the AMIPS problem into ANNS problem, enabling it to be tackled via LSH-based methods [17, 18]. Such transformations typically involve two functions, $P(x)$ and $Q(q)$, which map data points and query vectors from the original inner product space $(\mathbb{R}^d, ip)$ into a new metric space $(\mathbb{R}^m, dist)$. Under these mappings, maximizing $q^\top x$ becomes equivalent to minimizing $dist(Q(q), P(x))$, allowing the AMIPS problem to be answered by performing ANNS on $Q(q)$ in the transformed space. Subsequently, we can address the AMIPS problem by utilizing the LSH functions specifically designed for the metric space [6, 34, 35]. These methods are straightforward to deploy and incur an extremely low building cost, as they only require the computation of hash values. Nevertheless, these transformations often neglect norm bias, resulting in distance distortions [23]. That is, points with distinct inner products relative to the query may end up at similar distances from the query in the new space, rendering the transformed ANNS ineffective. To mitigate this issue, several works [23, 45, 47] propose partitioning the dataset into blocks based on vector norms,

ensuring that points within each block share similar norms. Consequently, ANNS queries within a single block are less affected by distortion. However, this design necessitates searching across multiple blocks, which increases query costs compared to querying the entire dataset at once, thereby creating a trade-off between accuracy and efficiency. Although LSH based methods are easy to implement and have low indexing costs, their query performance for AMIPS is generally inferior to graph-based methods due to the distance distortion [11] and the inherent hash boundary issue [40].

2.2 Graph-based methods

Given the proven effectiveness of graph-based methods such as HNSW [29] and DiskANN [36] in ANNS, applying them to AMIPS is a promising direction. A straightforward strategy involves constructing APGs directly in the inner product space. This can be achieved by modifying existing frameworks (e.g., replacing the metric function in HNSW with the inner product, known as ip-NSW [30]), a practice commonly adopted in industrial systems due to its simplicity. However, since the inner product is non-metric and the MIPS neighbor of a point x's MIPS neighbor is not x's the MIPS neighbor again, greedy search in APGs for AMIPS suffers from degraded performance, primarily due to norm bias. To address this, several studies [11, 37, 38, 48] propose improved edge selection strategies that prioritize large-norm points. While beneficial in some cases, these methods often fail when the dataset exhibits a wide norm distribution. In such cases, large-norm points disproportionately attract incoming edges, creating "traps" that hinder search efficiency. Moreover, compared to LSH-based methods, graph-based methods require much higher indexing costs due to the complexity of graph construction, where numerous inner product computations are needed to find neighbors for each point. Therefore, it becomes crucial to design graph-based methods that can effectively mitigate the norm bias issue while maintaining reasonable indexing costs.

2.3 Other Methods

Other methods, such as learning, sampling, and quantization based methods, have also been proposed for AMIPS [13, 15, 33]. Shen et al. [33] propose a binary code learning framework that preserves the inner product among raw data vectors. BanditMIPS [41] estimates the inner product for each point by subsampling coordinates and adaptively evaluates more coordinates for more promising points. These methods can effectively reduce the computational cost of inner product by using less bits or fewer dimensions to represent vectors. However, the incurred estimation errors lead to lower query accuracy, leading to still a high query complexity to achieve high query accuracy, limiting their scalability to large-scale datasets.

3 Preliminaries

In this paper, we focus on the AMIPS problem in memory. Let $\mathbb{R}^d$ be a d-dimensional vector space, $x_1^\top x_2$ represents the inner product between points $x_1, x_2 \in \mathbb{R}^d$, and $\langle x_1, x_2 \rangle$ denotes the angle between x_1 and x_2.

3.1 Problem Definition

Definition 3.1 ((c,k)-Approximate Maximum Inner Product Search). Given a dataset $\mathcal{D}$, a query point $q \in \mathbb{R}^d$, an approximation ratio $0 < c < 1$, and a positive integer k, a (c,k)-approximate maximum inner product search (AMIPS) returns k points $x_1, \ldots, x_k \in \mathcal{D}$, sorted in descending order according to their inner products with the query point q. For each $i = 1, \ldots, k$, the following condition holds: $q^\top x_i \geq c \cdot q^\top x_i^*$, where $q^\top x_i^*$ represents the i-th maximum inner product in the dataset.

The ratio c denotes the approximation property of AMIPS and is not directly utilized in graph-based methods. Next, we describe some properties of the normal distribution, which are used to prove the correctness of our framework.

Definition 3.2 (i.i.d. d-dimensional normal random vector). A d-dimensional point $q = (q_1, \cdots, q_d)$ is an i.i.d. d-dimensional normal random vector if each of its entries q_i is independently normally distributed as $N(\mu, \sigma^2)$, where μ and σ^2 are the expectation and variance of q_i, respectively. We denote this as $q \sim N^d(\mu, \sigma^2)$.

Lemma 3.3 (Independence [19, 32]). *If X and Y are jointly normal and uncorrelated, then they are independent.*

Lemma 3.4 ([19, 42]). *The sum $S = \sum_{i=1}^{d} a_i X_i$ follows a normal distribution $N(0, \sum_{i=1}^{d} a_i^2)$, where $X_i \sim N(0,1)$ and $a_i \in \mathbb{R}$.*

3.2 Locality Sensitive Hashing based Methods

Definition 3.5 (Locality-sensitive hashing). Given a distance function $dist$, a distance threshold r, an approximation ratio $c > 1$, and two probability values p_1, p_2, where $p_1 > p_2$, a family $\mathcal{H} = \{h : \mathbb{R}^d \to U\}$ is called (r, cr, p_1, p_2)-locality sensitive, if for any $x_1, x_2 \in \mathbb{R}^d$, it satisfies the following two conditions:

(1) If $dist(x_1, x_2) \leq r$, then $\Pr[h(x_1) = h(x_2)] \geq p_1$;
(2) If $dist(x_1, x_2) \geq cr$, then $\Pr[h(x_1) = h(x_2)] \leq p_2$.

LSH index is widely used for solving the ANNS problem due to its simple structure. Since inner product is not a valid metric, no standard LSH function can be directly applied to AMIPS [23, 34]. Hence, we consider an LSH function based on cosine similarity, *i.e.,* angular distance, which is defined as follows [9, 34]:

$$h(x) = \text{sign}(a^\top x), \tag{1}$$

where a is a d-dimensional random vector with each entry independently chosen from the standard normal distribution $N(0,1)$.

Lemma 3.6. *The collision probability of $h(x)$ and $h(q)$ under the LSH function in Eq. 1 is [47]:*

$$\Pr[h(x) = h(q)] = 1 - \frac{\langle q, x \rangle}{\pi} \tag{2}$$

Lemma 3.6 shows that points with high cosine similarity are more likely to have the same hash value, which helps in identifying the required neighbors from the hash buckets containing q.

Based on Equation 1, we can construct multi-dimensional hash tables to map the data points [31, 34]. This approach enables us to efficiently retrieve approximate ANNS results by only inspecting the query point's buckets. As a result, the indexing and query times for LSH-based methods can be extremely fast, particularly when targeting lower accuracy. However, due to the inherent hash boundary issues [11, 40, 47], LSH based methods hardly achieve a higher accuracy compared to graph-based methods.

4 Motivation

Norm bias is a commonly discussed phenomenon in the MIPS problem [23, 45, 47], which indicates that points with larger norms are more likely to exhibit a larger inner product with a query than points with smaller norms. Consequently, it is more promising to identify the MIPS results from points with larger norms. However, few studies quantitatively analyze the probability that MIPS results originate from points with large norms, which makes it difficult to fully leverage the effect

of norm bias for MIPS queries. In this section, we first quantify the norm bias and reveal the Norm Domination phenomenon (Section 4.1). We then analyze the norm and MIPS result distributions in real datasets to demonstrate the dominance of high-norm points (Section 4.2). Finally, we explore how this bias adversely affects graph structure (Section 4.3), leading to the inefficiency of MIPS queries on APGs.

4.1 Norm Domination

To formalize norm bias, we begin by considering a simplified scenario with two points, x_1 and x_2, where $r_1 = \|x_1\|$ and $r_2 = \|x_2\|$, with $r_1 > r_2$. Based on the norm bias phenomenon, for any query q, it is reasonable to infer that x_1 is more likely to yield a larger inner product with q than x_2.

Lemma 4.1. *Consider two orthogonal points $x_1, x_2 \in \mathbb{R}^d$ with $r_1 = \|x_1\|, r_2 = \|x_2\|$, and a random point $q \sim N^d(0,1)$. If $q^\top x_1 > 0$ and $q^\top x_2 > 0$, then the probability that x_1 has a larger inner product with q than x_2 is $\frac{2}{\pi} \cdot \arctan\left(\frac{r_1}{r_2}\right)$.*

The proof of Lemma 4.1 is presented in the Appendix. Lemma 4.1 quantifies the norm bias phenomenon by revealing the probability that x_2 becomes the MIPS result of q when another point x_1 exists. To simplify the model, we assume that the maximum inner product is positive. Based on this lemma, it is evident that points like x_1, with a larger norm, are more likely to be the MIPS result, and the likelihood increases with the ratio of their norms, r_1/r_2. Thus, when the norms of the two points differ significantly, we can compute how small the probability of x_2 becoming the MIPS result can be.

Estimation of the domination probability. However, the dataset $\mathcal{D}$ typically contains multiple points, and it is essential to analyze the norm bias in this case. To simplify the analysis, we divide $\mathcal{D}$ into two parts and compute the probability that the MIPS result falls into one of them. Let us consider two sets of random non-zero points $X = \{x_1, x_2, \cdots, x_{n_1}\}$ and $Y = \{y_1, y_2, \cdots, y_{n_2}\}$ with $r_1 = \min_{x \in X} \|x\|$ and $r_2 = \max_{y \in Y} \|y\|$. For an i.i.d. d-dimensional normal random query point $q \sim N^d(0,1)$, let x^* be the MIPS neighbor of q in the set $X \cup Y$, we aim to estimate the probability that $x^* \in X$ when $\gamma = r_1/r_2 > 1$. Based on Lemma 3.4, we have $I_i = q^\top x_i \sim N(0, \|x_i\|^2)$. To simply estimate the cumulative density function (cdf) of $I_X^* = \max_{x \in X} q^\top x$, we ignore the correlations between any two variables I_i and I_j, then for any a number t,

$$\begin{aligned} &\Pr[I_X^* \le t] = \Pr[\max I_i \le t] \\ =&\Pr[I_1 \le t, I_2 \le t, \cdots, I_i \le t] \\ =&\prod_{i=1}^{n_1} \Pr[I_i \le t] = \prod_{i=1}^{n_1} \Phi(\frac{t}{\|x_i\|}) \end{aligned} \tag{3}$$

where $\Phi(t)$ is the cumulative distribution function (cdf) of the standard normal distribution. So, the cdf of I_X^* is $F_{I_X^*}(t) = \prod_{i=1}^{n_1} \Phi(\frac{t}{\|x_i\|})$. Similarly, for $I_Y^* = \max_{y \in Y} q^\top y$, its cdf is $F_{I_Y^*}(t) = \prod_{i=1}^{n_2} \Phi(\frac{t}{\|y_i\|})$. Assume I_X^* and I_Y^* are independent, we have

$$\begin{aligned} \Psi(X,Y) =& \Pr[I_X^* > I_Y^*] \\ =& \int_{-\infty}^{+\infty} \int_{t_1}^{+\infty} F'_{I_X^*}(t_2)\, \mathrm{d}t_2 \cdot F'_{I_Y^*}(t_1)\, \mathrm{d}t_1. \end{aligned} \tag{4}$$

$\Psi(X,Y)$ forms an estimator of the probability that $x^* \in X$.

Lemma 4.2. *Denote $\gamma = r_1/r_2$ where $r_1 = \min_{x \in X} \|x\|$ and $r_2 = \max_{y \in Y} \|y\|$, $\Psi(X,Y)$ is at least*

$$F(\gamma, n_1, n_2) = \int_0^{+\infty} \left[1 - \Phi^{n_1}\left(\frac{t}{\gamma}\right)\right] d\Phi^{n_2}(t). \tag{5}$$

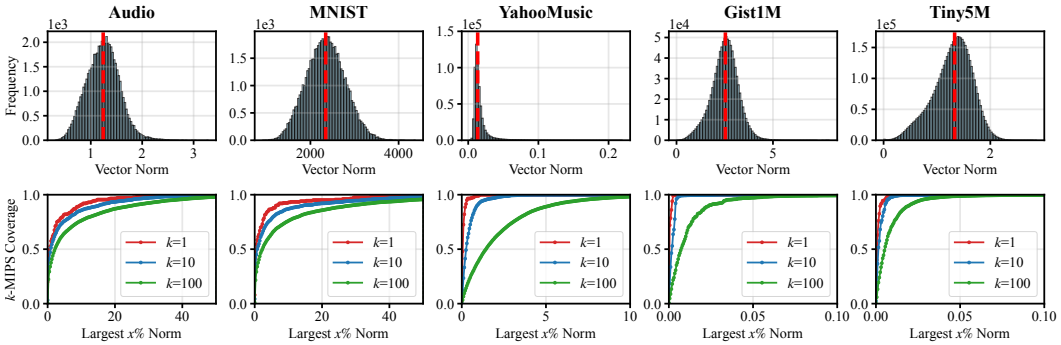


Fig. 2. Norm Distribution and MIPS Distribution in Real Datasets

The proof of Lemma 4.2 can be found in the Appendix. Lemma 4.2 provides strong evidence for the norm bias phenomenon between two sets of points, X and Y, when the norm of points in X is at least γ times larger than that in Y. It computes the lower bound of the probability that the MIPS result of a random query belongs to X rather than Y as $F(\gamma, n_1, n_2)$. $F(\gamma, n_1, n_2)$ converges rapidly to 1 with increasing n_1 or γ when $\gamma > 1$. Even if n_1 is only a few thousand, $F(\gamma, n_1, n_2)$ can approach 1 when $\gamma = 2.0$ and $n_2 = 1$ million. We refer to this phenomenon as *norm domination*.

Definition 4.3 (Norm Domination). Consider two random point sets, $X = \{x_1, x_2, \cdots, x_{n_1}\}$ and $Y = \{y_1, y_2, \cdots, y_{n_2}\}$, where $r_1 = \min_{x \in X} \|x\|$ and $r_2 = \max_{y \in Y} \|y\|$. When $\gamma = r_1/r_2$ is sufficiently large, the MIPS result of any query q in $X \cup Y$ will almost always appear in X, even if $|X| \ll |Y|$. We refer to this phenomenon as *norm domination*, where Y is dominated by X.

In contrast to norm bias, norm domination offers a more precise characterization: rather than simply stating that points with larger norms are more likely to appear among MIPS results, it asserts that once the norm gap exceeds a certain threshold, e.g., γ, the probability that the MIPS result falls within the large-norm subset approaches nearly 100%, even when that subset accounts for only about 1% of the points. This conclusion is striking, as it implies that with the presence of large-norm points, MIPS results can be found by searching only thousands of points, rather than the entire dataset of several million points, with high probability. In the following section, we present the distributions of MIPS results in real datasets to further illustrate norm domination.

4.2 Norm Domination in Real Datasets

We conduct an empirical analysis on five real datasets – Audio, MNIST, YahooMusic, Gist1M, and Tiny5M – to investigate whether these datasets exhibit skewed norm distributions and the relationship between MIPS results and the norm of points. Specifically, we first characterize the overall norm distributions to identify potential imbalances, and then quantify the proportion of k-MIPS results contributed by large-norm vectors. As shown in Figure 2, three key observations emerge: (1) all datasets display non-uniform, right-skewed norm distributions; (2) a small fraction of large-norm vectors dominate the top-k MIPS results; and (3) the degree of norm imbalance varies significantly across datasets. Detailed analyses of these findings are presented below.

Norm skewness across all datasets. The top row of Figure 2 shows the empirical histograms of vector norms for each dataset, with the median indicated by the *dashed red line*. First, all datasets exhibit highly non-uniform and asymmetric norm distributions, rather than flat or symmetric ones, aligning with our assumption on the norm of points. Even in relatively concentrated datasets such as Audio and MNIST, the ratios between the 99-th and 10-th percentiles ($2.06/0.83 \approx 2.5\times$

and 3378/1785 ≈ 1.9×, respectively) reveal persistent asymmetry. Second, the majority of vectors cluster around relatively small magnitudes, while a small fraction extend into a long right tail with substantially larger norms. For example, although the norms in Audio span from 0.3 to 3.1, over 75% of them fall below 1.45, and only the top 1% reach 2.06; a similar situation is even more pronounced in the YahooMusic dataset: norms in YahooMusic span from 0.0017–0.34, but 75% of them fall below 0.0178, and only the top 1% reach 0.08, indicating the highly concentrated and long-tail distribution of norms in real datasets. This disparity in real datasets demonstrates that, while the norm of most points lies in compact small regions, a small subset of large-norm vectors dominate the upper tail, creating the structural basis for norm domination in the MIPS problem.

Concentration of MIPS results on large-norm points. To quantify how the observed norm skewness affects MIPS retrieval, we examine the cumulative contribution of large-norm vectors to the top-k MIPS results of all query points. As shown in the bottom row of Figure 2, each curve depicts the fraction of retrieved top-k neighbors that originate from the top-x% of vectors ranked by their norms. A steeper curve indicates a stronger dominance of large-norm points in retrieval outcomes, reflecting how severely the query results are biased toward the large-norm region. The top-1 and top-10 curves rise sharply within the first few percent of the points, indicating that only a tiny portion of large-norm vectors contribute to retrieval outcomes. For instance, in Tiny5M and Gist, more than 99% of the top-10 MIPS neighbors are drawn from the top 0.05% of large-norm points, while even the broader top-100 set is almost entirely covered by the top 0.1%. Audio and MNIST exhibit a milder but still significant bias, where 70–80% of the top-10 results originate from the top 10% of large-norm points. YahooMusic lies between these extremes, showing strong skewness for top-10 retrieval (over 99% within the top 5%) but a smoother distribution for larger k.

Dataset-dependent skewness intensity. Although norm domination is universally observed, its severity varies markedly across datasets. YahooMusic, Gist, and Tiny5M exhibit highly skewed norm distributions, where over 90% of top-10 MIPS results arise from the top 1–2% of the largest-norm points. In contrast, Audio and MNIST display more moderate imbalance—approximately 70–80% of retrieved results are concentrated within the top 10% of vectors—showing that smaller-norm regions still contribute meaningfully to retrieval. This dataset-dependent disparity suggests that the number of dominating vectors varies with the underlying norm distribution, motivating the need for a norm-adaptive strategy that facilitates the efficient AMIPS query based on the specific norm distribution in the datasets.

Example 4.4. These experimental findings also validate the theoretical norm-domination model introduced in Lemma 4.2. Specifically, we verify this correspondence by splitting each dataset into two subsets according to norms: the top 10% large-norm vectors as X and the bottom 50% as Y, from which we estimate the norm ratio γ. In the highly skewed dataset YahooMusic, where $n_1 = 62{,}476$ with minimum norm $r_1 = 0.026$ and $n_2 = 312{,}380$ with maximum norm $r_2 = 0.013$, we obtain $\gamma = r_1/r_2 = 2.0$, the theoretical domination probability is $F(\gamma, n_1, n_2) \approx 0.999$. In contrast, for the relatively balanced dataset Audio ($n_1 = 5{,}418$, $r_1 = 1.63$; $n_2 = 27{,}093$, $r_2 = 1.24$), $\gamma = 1.31$ yields $F(\gamma, n_1, n_2) \approx 0.946$. This aligns closely with the empirical observation that more than 99% of top-10 MIPS neighbors originate from the top 1% of large-norm vectors in the YahooMusic dataset, while the proportion decreases to 70–80% in the Audio dataset. These results demonstrate that the observed empirical dominance ratios quantitatively conform to the theoretical prediction, reinforcing the validity of the proposed norm-domination model in practical MIPS scenarios.

4.3 The Effect of Norm Domination on APG

The theoretical analysis of norm domination, coupled with the empirical evidence of severe retrieval concentration in real datasets, demonstrates that MIPS results are overwhelmingly concentrated

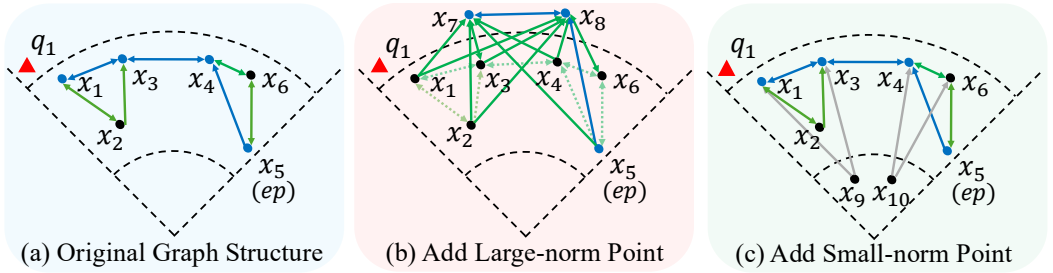


Fig. 3. Illustration of how vector norms influence graph structure (max out-degree is set to 2).

on a small subset of large-norm points. This phenomenon of norm domination not only biases retrieval results but also fundamentally alters the structure of the APG for MIPS. To illustrate this, we compare the changes in graph structures and query performance in APGs when large-norm or small-norm points are inserted, as shown in Figure 3.

Syphon effect of large-norm points in APGs. According to Lemma 4.1, large-norm points tend to produce high inner products with most other points. As a result, these points are frequently selected as neighbors and accumulate an exceptionally high number of in-edges, ultimately becoming *hub nodes*. As illustrated in Figure 3(b), the insertion of large-norm nodes x_7, x_8 traps almost all the in-edges in the APG. Many modest and small-norm nodes (e.g., x_1, x_2, x_3, x_4, x_6) replace their original local edges (such as $x_3 \leftrightarrow x_4$ or $x_1 \leftrightarrow x_2$) with edges directed toward x_7 or x_8 in order to maximize the inner product. Hub nodes x_7 and x_8 attract a large number of in-edges from x_1–x_6, resulting in a sharp spike in in-degrees at the head. In contrast, the in-degrees of x_1–x_6 decrease because their original in-edges are evicted due to the out-degree constraint of their neighbors, thus compromising the connectivity in the APG. These two phenomena collectively undermine the quality of the APG for MIPS queries. As a result, the greedy search for AMIPS is quickly "pulled" toward the hub region $\{x_7, x_8\}$ and struggles to escape due to the lack of connectivity, leading to low query accuracy for certain query points.

Futility of small-norm points in APGs. In contrast, small-norm vectors produce low inner products with other nodes, and thus receive few in-edges. As depicted in Figure 3(c), when small-norm points x_9, x_{10} are inserted, most of their out-edges point to higher-norm nodes (*e.g.,* x_1, x_3, x_4, and x_6), with no edges directed back to them. As a result, x_9 and x_{10} remain at the periphery of the graph, forming dead-end branches that are never visited during the search unless they are entry points. These small-norm nodes contribute little to the overall connectivity or navigability of the graph but still incur storage and construction overhead without improving retrieval accuracy.

Example 4.5. Figure 3 illustrates the AMIPS query under three scenarios: (a) a balanced APG, (b) the APG after inserting large-norm points, and (c) the APG after inserting small-norm points. Each node has an out-degree of 2. In (a), all nodes $\{x_1, \ldots, x_6\}$ have comparable norms, forming a well-connected structure. A greedy search beginning from x_5 follows the path $x_5 \rightarrow x_4 \rightarrow x_3 \rightarrow x_1$, successfully reaching the correct result. In (b), adding large-norm nodes x_7, x_8 causes all existing nodes to redistribute their edges toward the hub nodes x_7, x_8. The hubs, in turn, primarily connect to each other or to a few nearby nodes ($x_7 \rightarrow x_8$, $x_8 \rightarrow x_3$). The search path thus becomes $x_5 \rightarrow x_8 \rightarrow x_7$, unable to reach x_1, demonstrating how hub dominance traps the search in the head region. In (c), the insertion of small-norm nodes x_9, x_{10} results in isolated low-degree nodes without altering connectivity. Due to their small inner products with other points, no other nodes select them as neighbors, leaving them with zero in-degree. The search path remains unchanged. These nodes consume storage while contributing little to retrieval accuracy.

5 The Norm-Adaptive Partitioning Framework

As analyzed in Section 4, the issue of norm domination significantly impacts the MIPS problem. First, a substantial number of small-norm points are rarely returned as MIPS results for any query. Therefore, identifying and excluding these points from the index can enhance query efficiency. Second, large-norm points tend to attract a disproportionately high number of edges, resulting in an imbalanced edge distribution within the APGs. To address these challenges and improve MIPS query efficiency, we propose a norm-adaptive partitioning strategy that determines which data points should be stored in the APGs.

5.1 Norm-Adaptive Partitioning Scheme

Given a dataset $\mathcal{D} = \{x_1, x_2, \cdots, x_n\}$ sorted in non-increasing order of norms ($\|x_i\| \geq \|x_{i+1}\|$), the norm-adaptive partition (NAP) divides $\mathcal{D}$ into three segments—Head, Body, and Tail—such that:

- The Head consists of large-norm points $\{x_1, \cdots, x_{n_1}\}$. As shown in Figure 2, due to the long-tail distribution, certain points exhibit significantly larger norms than the rest. These points are highly likely to appear as MIPS results, making it reasonable to prioritize searching within this segment. Early identification of candidates from Head can either terminate the query promptly or guide subsequent exploration in other regions.
- The Body includes the medium-norm points from x_{n_1+1} to x_{n-n_2}. This segment forms the core of the dataset and typically exhibits a narrower norm range than Head and Tail (see Figure 2). Although their norms are smaller than those in Head , these points can still become MIPS results for some queries q, particularly if their cosine similarity with q is high. To capture such cases, we construct an APG over Body. By excluding large- and small-norm points, the resulting APG suffers less from norm domination, leading to improved search effectiveness.
- The Tail consists of small-norm points ranging from x_{n-n_2+1} to x_n. Its size n_2 is determined by the size of Head (n_1) and the norm ratio $\gamma = \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$. Based on Lemma 4.2, the probability that the true MIPS results fall outside Tail is at least $F(\gamma, n_1, n_2)$, since these points are likely dominated by those in Head .

Definition 5.1 (δ-NAP). Given a tolerance error δ, and a dataset $\mathcal{D} = \{x_1, x_2, \cdots, x_n\}$ sorted by non-increasing norm, the δ-NAP seeks the maximal value of n_2 such that there exists a subset of n_1 points satisfying $F(\gamma, n_1, n_2) > 1 - \delta$, where $\gamma = \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$.

However, computing the optimal n_2 for δ-NAP is computationally intensive, as the search space for (n_1, n_2) consists of $O(n^2)$ combinations, and evaluating $F(\gamma, n_1, n_2)$ for each is costly. To overcome this, we next describe practical algorithms that efficiently and effectively perform δ-NAP on real-world datasets.

5.2 Practical NAP Implementation

We consider two NAP algorithms to determine the boundaries of Head and Tail based on the estimated success probability $F(\gamma, n_1, n_2)$.

Greedy δ-NAP. Let $\psi(n_2)$ be the minimal n_1 such that $F(\gamma, n_1, n_2) > 1 - \delta$. Since $F(\gamma, n_1, n_2)$ increases with n_1 and decreases with n_2 for a fixed γ, it is reasonable to assume that $\psi(n_2)$ increases with n_2, given that $\gamma = \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$ changes gradually. We can then adopt a greedy strategy to find the appropriate $\psi(n_2)$ for each n_1. Algorithm 1 outlines the steps of the greedy δ-NAP algorithm. It begins by initializing values for n_1 and n_2 (lines 1). Specifically, n_1 is set to 2048, and n_2 is initialized to 100. We then check whether $F(\gamma, n_1, n_2) > 1 - \delta$ holds in the initial case (line 4). If not, it indicates that all points in $\mathcal{D}$ have nearly identical norms, rendering the NAP unnecessary. In this case, we

set $n_1 = n_2 = 0$. Otherwise, we attempt to increase n_2 until $F(\gamma, n_1, n_2) > 1 - \delta$ no longer holds (lines 6 to 9), thereby finding the maximum n_2 that satisfies the tolerance condition. Next, n_1 is incremented (line 6). If $F(\gamma, n_1, n_2)$ decreases with increasing n_1, the algorithm terminates, as this suggests that the norm reduction from x_{n_1-1} to x_{n_1} (line 11) is too large. Otherwise, we decrease n_1 until $F(\gamma, n_1, n_2) > 1 - \delta$ is satisfied (lines 10 to 15). This process continues until every point is classified as either a Head or Tail point.

Algorithm 1: δ-NAP in $\mathcal{D}$

Input: The sorted dataset $\mathcal{D} = \{x_i\}_{i=1}^n$, parameters δ
Output: The size of Head n_1 and Tail n_2

1 $n_1 \leftarrow 2048$, $n_2 \leftarrow 100$;
2 $old_F \leftarrow 0$;
3 $\gamma \leftarrow \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$;
4 **if** $F(\gamma, n_1, n_2) < 1 - \delta$ **then**
5 **return** 0,0;
6 **while** $n_1 + n_2 < n$ **do**
7 **while** $F(\gamma, n_1, n_2) > 1 - \delta$ **do**
8 $n_2 \leftarrow n_2 + 1$;
9 $\gamma \leftarrow \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$;
10 **while** $F(\gamma, n_1, n_2) < 1 - \delta$ **do**
11 **if** $F(\gamma, n_1, n_2) < old_F$ **then**
12 **Break**;
13 $old_F \leftarrow F(\gamma, n_1, n_2)$;
14 $n_1 \leftarrow n_1 + 1$;
15 $\gamma \leftarrow \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$;
16 **return** $n_1, n_2 - 1$;

Heuristic Two-Step δ-NAP. While the greedy δ-NAP algorithm efficiently identifies reasonable values for n_1 and n_2, it approximates the norm of points in Head as $\|x_{n_1}\|$ and the norm of points in Tail as $\|x_{n-n_2-1}\|$ when computing the domination probability p. This approach allows for a quick estimation of p, but it tends to significantly underestimate it, especially when there is substantial variation in the norms of the points in Head .

To improve this estimation, we consider the more accurate domination probability p based on the Eq. 4, given by:

$$p = \int_{-\infty}^{+\infty} F_{I_Y^*}(t) \cdot F'_{I_X^*}(t)\, \mathrm{d}t. \tag{6}$$

Here, $F_{I_X^*}(t)$ represents the probability that $\max_{x \in \text{Head}} q^\top x \le t$, and $F_{I_Y^*}(t)$ represents the probability that $\max_{x \in \text{Tail}} q^\top x \le t$, which can be computed as

$$F_{I_X^*}(t) = \prod_{i=1}^{n_1} \Phi\left(\frac{t}{\|x_i\|}\right), F_{I_Y^*}(t) = \prod_{i=n-n_2}^{n} \Phi\left(\frac{t}{\|x_i\|}\right) \tag{7}$$

The value of p increases monotonically with n_1 and decreases with n_2, regardless of the norm distribution of $\mathcal{D}$. Thus, the optimal n_2 is found when $p > 1-\delta$, but $p < 1-\delta$. The computation of p is significantly more complex than that of $F(\gamma, n_1, n_2)$. To reduce computational cost, we recursively

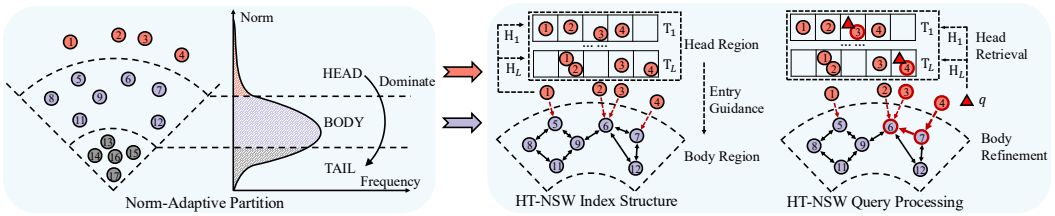


Fig. 4. Construction and AMIPS query of **HT-NSW**.

update $F_{I_Y^*}(t)$, $F_{I_X^*}(t)$, and precompute the derivative of $F_{I_X^*}(t)$, expressed as $F'_{I_X^*}(t) = F_{I_X^*}(t) \cdot \varphi(t)$, where

$$\varphi(t) = \sum_{i=1}^{n_1} \frac{f\left(\frac{t}{\|x_i\|}\right)}{\|x_i\| \cdot \Phi\left(\frac{t}{\|x_i\|}\right)} \tag{8}$$

After determining n_2, we select a proper n_1 using the elbow method [8, 12], based on the norm ratio $\gamma = \frac{\|x_{n_1}\|}{\|x_{n-n_2+1}\|}$. The elbow method is employed because it is easy to implement and offers a clear indication of where norm domination begins to diminish, helping us choose a balanced value for n_1 that maximizes efficiency without selecting an excessive number of points as Head points. This technique enables us to identify the appropriate n_1 by examining the point where the norm ratio exhibits a sharp change, signaling the optimal trade-off between query accuracy and the high cost of intensive queries in the Head region.

6 NAP-based AMIPS Query

After partitioning the dataset $\mathcal{D}$ using NAP, we construct separate indexes for Head and Body. The suitable indexes for Head and Body differ due to their distinct norm distributions and point densities. The Body region consists of a large number of points with similar norms after the NAP partition, making it ideal to utilize APG as the index. This facilitates the AMIPS query and mitigates norm domination. In contrast, the Head region contains fewer points with varied norms. For this reason, it is recommended to use Linscan [1, 39] or LSH-based indexes [23, 47] rather than APGs, for the following reasons: Firstly, the dispersed norm distribution in Head makes it extremely difficult to avoid the issue of norm domination. Secondly, the small size of Head makes it challenging to find a sufficient number of effective MIPS when compared to the entire dataset. On the other hand, due to its data-independent structure and low construction cost, LSH is sufficient to efficiently support AMIPS queries in Head . Therefore, we propose a practical NAP-based hybrid index, HT-NSW, which integrates LSH [34] and HNSW [30] structures to efficiently support the AMIPS query on top of NAP. Figure 4 illustrates the overall framework of HT-NSW, which comprises three major stages: *Norm-Adaptive Partition*, *Index Construction*, and *Query Processing*. We now describe the construction of HT-NSW and its efficient AMIPS query processing.

6.1 A NAP-based Hybrid Index: HT-NSW

Building Hash Tables for Head . We choose the LSH functions defined in Eq. 1 to map the points in Head. An LSH function $h(x)$ alone is insufficient for finding nearest neighbors under cosine similarity effectively, as $\Pr[h(x) = h(q)]$ can still be very high even when $\langle q, x \rangle$ is large, e.g., $\pi/2$. Therefore, a common strategy in LSH-based methods is to construct several multi-dimensional hash tables [18, 40]. In our HT-NSW method, $\mathcal{I}_H$ consists of L K-dimensional hash tables. We randomly choose K independent hash functions as defined in Eq. 1 for each hash table [34, 40].

Algorithm 2: k-AMIPS Query

Input: The sorted dataset $\mathcal{D} = \{x_i\}_{i=1}^n$, the query point q, parameters k, indexes $\mathcal{I}_H$ for Head and $\mathcal{I}_B$ for Body
Output: k-AMIPS of q

1 $R \leftarrow k$-AMIPS neighbors of q found in $\mathcal{I}_H$;
2 $R_c \leftarrow k$-ANNS angular neighbors of q found in $\mathcal{I}_H$;
3 **if** $\|q\| \cdot \|x_{n_1+1}\| < \min_{x \in R} q^\top x$ **then**
4 **return** R;
5 $EPs \leftarrow R_c$'s MIPS neighbors in Body ;
6 Search in $\mathcal{I}_H$ with EPs as the entry points;
7 $R_B \leftarrow k$-AMIPS neighbors of q found in $\mathcal{I}_H$;
8 Update R with R_B;
9 **return** R;

Algorithm 3: k-AMIPS Query in $\mathcal{I}_H$

Input: The query q, parameters k, δ_L, indexes $\mathcal{I}_H$ for Head
Output: Two sets of points

1 $R \leftarrow \emptyset$; ▷ **Store the found best k-AMIPS of q**
2 $R_c \leftarrow \emptyset$; ▷ **Store the found best angular k-ANNS of q**
3 **for** $l \leftarrow 0 : K$ **do**
4 **for each hash table** T_i **do**
5 **if** $\|g_i(x), g_i(q)\|_H = l - 1$ **then**
6 Update R with x;
7 Update R_c with x;
8 **if** $\|q\| \cdot M_j \cos\theta_l(\delta_L) \leq \min_{x \in R} q^\top x$ **then**
9 **return** R,R_c;
10 **return** R,R_c;

Let $g_i(x) = (h_{i,1}(x), \cdots, h_{i,K}(x))$ be the hash keys of point x in hash table T_i. We store the pairs $\{g_i(x), x\}$ in T_i.

Building HNSW for Body. Given that HNSW [29, 44] achieves superior query performance on ANNS, we build the HNSW index for Body . However, several studies [44, 46] show that the multi-layer structure of HNSW contributes little to query performance, as it typically generates only one point from the upper layer as the entry point for the search in the base layer. To refine the entry point quality and reduce the index size of $\mathcal{I}_B$, we adopt a single-layer HNSW structure for Body. We then select the entry points based on the k-AMIPS results found in Head. We construct the required single-layer HNSW for Body as outlined in Algorithm 1 of [29]. Specifically, we first create an empty APG. Points are then inserted into the APG one by one. For each insertion, we find the at most M MIPS neighbors of point x and mutually connect x to its neighbor y. When the out-degree of point y exceeds $2M$ (indicating that y has been selected as a neighbor for other points), we select M new neighbors for y from its old neighbor set.

6.2 NAP-based AMIPS Query Algorithm

After indexing Head with $\mathcal{I}_H$ and Body with $\mathcal{I}_B$, we perform the AMIPS query for q as shown in Algorithm 2. We first identify the k-AMIPS neighbors of q in $\mathcal{I}_H$ (line 1). Next, we examine whether

Algorithm 4: k-AMIPS in $\mathcal{I}_B$

Input: A query point q, the APG $\mathcal{I}_B$, the entry point set EPs, the parameters k and ef
Output: k points in $\mathcal{I}_B$

1 $R \leftarrow \emptyset$;
2 **while** $|EPs| > 0$ **do**
3 $ep \leftarrow$ extract q's best MIPS results from EPs;
4 **if** $|R| \geq ef$ *and* $q^\top ep < q^\top x_f$ **then**
5 **break**;
6 **for** $x \in ep$*'s neighbor set* **do**
7 **if** x *is unvisited* **then**
8 $EPs \leftarrow EPs \cup \{x\}$, $R \leftarrow R \cup \{x\}$;
9 **while** $|R| > ef$ **do**
10 Remove q's worst MIPS results in R;
11 $x_f \leftarrow \arg\min_{x \in R} q^\top x$;
12 $R \leftarrow$ The best k MIPS results to q in R;
13 **return** R;

better k-AMIPS neighbors exist in Body (line 3). Since x_{n_1+1} is the point with the largest norm in Body, the product $\|q\| \cdot \|x_{n_1+1}\|$ serves as the upper bound for the inner product between q and any point in Body. Thus, if $\|q\| \cdot \|x_{n_1+1}\| < \min_{x \in R} q^\top x$, we can safely terminate the query without further searching in $\mathcal{I}_B$. Otherwise, we use the results obtained from $\mathcal{I}_H$ to select appropriate entry points for the search in $\mathcal{I}_B$ and then generate the final k-AMIPS results for q from both $\mathcal{I}_H$ and $\mathcal{I}_B$ (line 7). We now describe the query algorithms in $\mathcal{I}_H$ and $\mathcal{I}_B$, including the entry point selection for the search in $\mathcal{I}_B$.

AMIPS & ANNS Query in $\mathcal{I}_H$. Unlike traditional LSH-based methods for ANNS, $\mathcal{I}_H$ is built over angular distance rather than inner product. The conventional LSH-based query strategy, which probes candidates only from q's hash buckets, typically achieves limited query accuracy. To ensure an accurate and exhaustive query in $\mathcal{I}_H$, we adopt the more precise multi-probing strategy [28] for the search in $\mathcal{I}_H$, and we terminate the query adaptively by estimating the angle between q and x based on their hash values. Algorithm 3 describes our search strategy in $\mathcal{I}_H$. Specifically, we conduct at most $K+1$ rounds of search in $\mathcal{I}_H$ (lines 3-9). Since $g_i(x)$ is a K-dimensional binary vector, the Hamming distance $\|g_i(x), g_i(q)\|_H$ lies within the interval $[0, K]$, with a larger $\|g_i(x), g_i(q)\|_H$ corresponding to a larger angle $\langle q, x \rangle$, i.e., lower cosine similarity. Thus, at the l-th round of search, we probe the buckets $h_i(x)$ in T_i where $\|g_i(x), g_i(q)\|_H = l - 1$ (line 5). For each found point x when probing a bucket, we compute the angular distance $\langle q, x \rangle$ and inner product $q^\top x$, updating the results into R_c and R, respectively. R_c is used for generating high-quality entry points for the search in $\mathcal{I}_B$. Computing $\langle q, x \rangle$ incurs no additional cost since it can be derived from $q^\top x$. Finally, we consider adaptively terminating the query based on the properties of the LSH functions.

Lemma 6.1. *Given a tolerant error δ_L, let $\theta_l(\delta_L)$ be the solution to the following equation on θ,*

$$[1 - \sum_{j=0}^{l} \binom{K}{j} \cdot (1 - \frac{\theta}{\pi})^{K-j} \cdot (\frac{\theta}{\pi})^j]^L = \delta_L. \tag{9}$$

If the point x satisfies $\langle q, x \rangle \leq \theta_l(\delta_L)$, the probability that it is found during the first $l+1$ rounds of search is at least $1 - \delta_L$.

The proof of Lemma 6.1 can be found in the Appendix. Based on this lemma, we terminate the search in $\mathcal{I}_H$ after $l+1$ rounds of search if $\|q\| \cdot M_j \cos\theta_l(\delta)$ is smaller than the best k results found so far (line 8), where M_j represents the largest norm of the unseen points in Head . When this condition is met, the probability of finding a better MIPS result for q from other hash buckets becomes extremely small.

AMIPS Query in $\mathcal{I}_B$. The k-AMIPS query for q in $\mathcal{I}_B$ follows the greedy search approach shown in Algorithm 4. In each search epoch, the point ep that has the maximum inner product with q in EPs is selected as the next entry point (Line 3). For each neighbor x of ep in the APG, we compute its inner product with q and update the results into the sets EP and R. The result set R stores the best ef points found so far. The search procedure repeats until no point in EPs has a larger inner product with q than any of the best ef points in R (Line 4). At that point, the search terminates, and the top k points in R are returned as the AMIPS results for q. The parameter ef is set to balance query efficiency and accuracy. A higher ef makes termination more difficult (Line 4), allowing us to explore more points to refine the query results, but it increases query cost.

Head-Guided Entry Point Generation. To enhance the query efficiency of the search in $\mathcal{I}_B$, we generate high-quality entry points based on the k-AMIPS results found in Head. First, for each point x in Head, we retrieve its AMIPS results in Body ahead and store them in the index during the indexing phase. Then, during the query phase, after the search in $\mathcal{I}_H$, we have found q's angular neighbors in Head , i.e., R_c (Algorithm 3, line 7). For each point x in R_c, we add its MIPS neighbors y in $\mathcal{I}_B$ to the entry point set EPs. Although the inner product is not a metric, it is expected that point y will have a large inner product with q, because the MIPS neighbor of an angular neighbor is likely to also be an MIPS neighbor [27]. Finally, we illustrate the overall AMIPS query process in HT-NSW through the example in Figure 4.

Example 6.2. Given a query point q (the red triangle), the right subgraph in Figure 4 illustrates the complete k-AMIPS query in HT-NSW with $k = 1$. The query first probes the Head index $\mathcal{I}_H$ round by round. The retrieved candidates from $\mathcal{I}_H$ are marked with a red face in the figure, i.e., point x_3 and x_4 where x_4 is the MIPS result of q in Head. Assume the inner product between x_4 and q is less than $\|q\| \cdot M$, where M is the largest norm of points in Body, indicating that the points in Body are not dominated by x_4 and the query needs to continue in $\mathcal{I}_B$. Then, we generate the entry point in Body based on the nearest neighbors of q in Head under cosine similarity, i.e., x_4. Finally, we conduct the search in the APG to find the AMIPS of q in Body , with x_7 as the entry point, since it is the MIPS neighbor of x_4 in Body . The search terminates after accessing x_7, as its inner product with q is larger than its neighbor in the APG, *i.e.,* x_6.

7 Complexity Analysis

We analyze the construction cost, space consumption, and query cost of HT-NSW in this subsection. The construction cost of HT-NSW is determined by the cost of NAP and the costs associated with constructing $\mathcal{I}_H$ and $\mathcal{I}_B$. The NAP process can be completed in at most n iterations, with each iteration having a constant cost of $O(1)$, resulting in an overall NAP cost of $O(n)$. The structure of $\mathcal{I}_H$ consists of L K-dimensional hash tables. Typically, we set $L = O(1)$ and $K = O(\log n_1)$ to ensure that, on average, each bucket contains exactly one point. Consequently, the time required to compute the hash values is $O(n_1 LK) = O(n_1 \log n_1)$, and the time to construct the hash tables is $O(n)$. The complexity of $\mathcal{I}_B$ is governed by HNSW, whose empirical query cost is $O(\log N)$, where N represents the number of points in HNSW [29]. The construction of HNSW in Body involves $n - n_1 - n_2$ insertions, with the cost of these insertions dominated by the query cost in the APG. Let $\beta = \frac{n-n_1-n_2}{n}$, the construction cost of $\mathcal{I}_B$ is then $O(\beta n \log(\beta n)) = O(\beta n \log(n))$. Thus, the total construction cost of HT-NSW is $O(n + n_1 \log n_1 + \beta n \log(n))$. Since $n_1 \ll n$, the construction cost

Table 1. Summary of Datasets

Datasets	Card.	Dim.	Attribute Type
Audio	54,187	192	Audio embedding
MNIST	60,000	784	Raw image
YahooMusic	625,000	300	Rating decomposition
Gist1M	1,000,000	960	GIST descriptor
Tiny5M	5,000,000	384	Image embedding

of HT-NSW can be simplified to $O(\beta n \log(n))$. Similarly, the space consumption of HT-NSW is determined by the sizes of $\mathcal{I}_H$ and $\mathcal{I}_B$, both of which are linear in the number of indexed points. Therefore, the space consumption of HT-NSW is $O(n)$. The worst-case query cost of HT-NSW occurs when all points in $\mathcal{I}_H$ are accessed, and the search continues within $\mathcal{I}_H$. In this scenario, the query cost is $O(\log n + n_1) = O(\log n)$, provided that n_1 remains sufficiently small. In summary, HT-NSW can be constructed within $O(\beta n \log(n))$ time and requires $O(n)$ space, where β is the ratio of the number of points in Body. HT-NSW answers AMIPS queries in $O(\log n)$ time.

8 Experiments

In this section, we present a series of extensive experiments on real-world datasets to provide a comprehensive analysis of the NAP framework and HT-NSW. The HT-NSW implementation[1] and its competitors are developed in C++. All experiments are conducted on an Ubuntu 24.04 LTS system equipped with 4 Intel(R) Xeon(R) Gold 6218 CPUs (160 threads) and 1.5 TB of RAM.

8.1 Experimental Settings

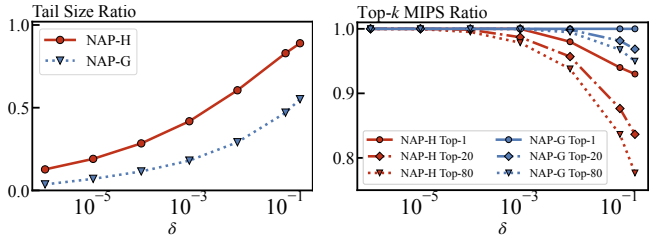
Datasets. We employ five real-world datasets that are widely used in Approximate Nearest Neighbor Search (ANNS) and Approximate Maximum Inner Product Search (AMIPS) [27, 30, 47], the details of which are summarized in Table 1 in ascending order of their sizes. We present their norm distribution and MIPS distribution in Figure 2. For each dataset, we randomly select 10K query points from its query set to evaluate the AMIPS query performance.

Competitors. We compare HT-NSW with several state-of-the-art graph-based methods for AMIPS.

- HNSW[2] [29] is a well-known vector search method that supports ANNS and AMIPS. Here we adopt the implement from Faiss and name it as **HNSW-F**.
- **ip-NSW**[3] **[30]** is a library for HNSW that supports AMIPS.
- **NAPG**[4] **[37]** enhances ip-NSW by introducing an adaptive edge selection strategy.
- **PSP**[5] **[11]** is a graph based methods that convert the AMIPS to ANNS by scaling the query point.
- FARGO[6] [47] is the state-of-the-art LSH methods to AMIPS.

Parameter Settings. In our default configuration, we set $k = 10$ for all algorithms and adopt fixed parameters for each method. For HT-NSW, we use the heuristic NAP strategy with $\delta = 0.001$ to partition the dataset. Additionally, we set $L = 2$ and $K = 10$ for $\mathcal{I}_H$, and $M = 16$ and $efC = 80$ for $\mathcal{I}_B$. For both HNSW-F, ip-NSW and NAPG, we use the same settings, i.e., $M = 16$ and $efC = 80$. For PSP, we configure $R = 40$, $\alpha = 60^{\circ}$, and $S = 50$ for all datasets. For FARGO, we set $N_0 = 4096$, $c = 0.8$, $L = 5$ and $K = 12$.

[1] https://github.com/SIGMOD2026-NAP/SIGMOD2026-NAP
[2] https://github.com/facebookresearch/faiss
[3] https://github.com/stanis-morozov/ip-nsw
[4] https://github.com/ThrunGroup/BanditMIPS/tree/camera_ready/algorithms/napg
[5] https://github.com/ZJU-DAILY/PSP
[6] https://github.com/Jacyhust/FARGO



(a) Tail size ratio on MNIST (b) Top-k MIPS ratio on MNIST

Fig. 5. Comparison between greedy and heuristic NAP.

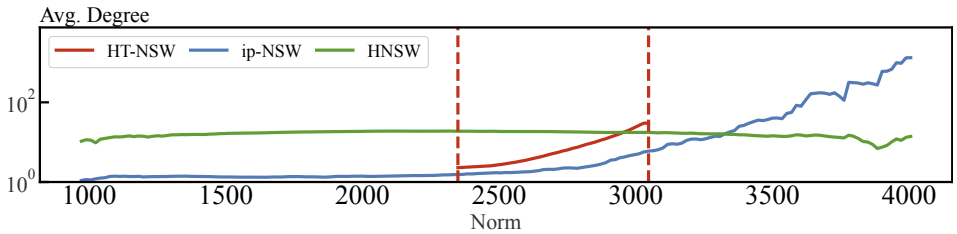


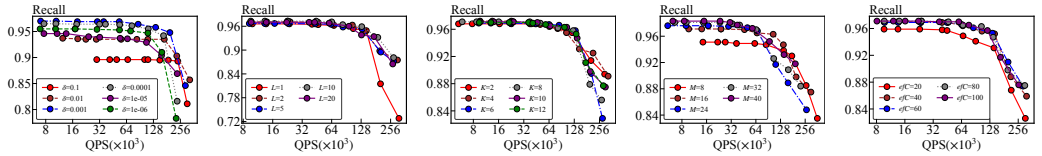
Fig. 6. Degree-Norm Distribution.

Evaluation Metrics. The query performance of each algorithm is evaluated using *QPS* (queries per second) and recall@k with a total of 160 threads. The *recall@k* is defined as follows [11], *Recall* = $\frac{|R \cap R^*|}{k}$ where R denotes the top-k AMIPS results returned by the algorithm, and R^* represents the exact top-k MIPS results for query q. Furthermore, we assess the indexing performance of each algorithm using *index size* and *indexing time* under 160 threads.

8.2 Self Evaluation of NAP

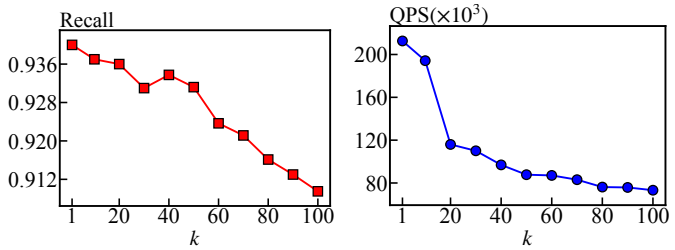
In this subsection, we conduct self evaluation to study the performance of NAP. Firstly, by varying the tolerance error δ, we compare greedy NAP and the heuristic two-step NAP in terms of their tail size and the accuracy loss incurred during partitioning. Secondly, we analyze the impact of NAP on the APG structure by examining the edge distribution of HT-NSW, HNSW (for ANNS), and ip-NSW (for AMIPS).

8.2.1 ***Evaluation of NAP Strategies.*** In Figure 5 we compare the greedy and heuristic two-step variants of NAP under varying tolerance errors δ (from 10^{-6} to 0.2). Figure 5(a) reports the ratio of Tailsize, defined as n_2/n, which reflects the fraction of data classified as tail points under each setting. Figure 5(b) reports the proportion of Top-k MIPS results retained within the Head and Body after NAP, where higher retention implies lower accuracy loss. Increasing δ results in a clear and monotonic growth in the Tail size for both NAP variants. For example, NAP-H increases the Tail ratio from about 10%–15% at $\delta = 10^{-6}$ to over 40% at $\delta = 10^{-3}$, and exceeds 85% when $\delta = 0.2$. This trend shows that both NAP strategies expand the partition boundary with increasing δ. Compared to the tail size, the Top-k MIPS ratio decreases only slightly as δ increases. For example, at $\delta = 10^{-5}$, the Top-1 MIPS ratio of NAP-H strategy remains nearly 100%. When δ increases to 10^{-3}, the Top-1 ratio still stays above 98% for both NAP variants. This observation aligns with the norm domination phenomenon discussed in Section 4, where large-norm vectors dominate MIPS results.



(a) δ on MNIST (b) L on MNIST (c) K on MNIST (d) M on MNIST (e) efC on MNIST

Fig. 7. Effect of parameters (δ, L, K, M, efC) on query performance on MNIST.

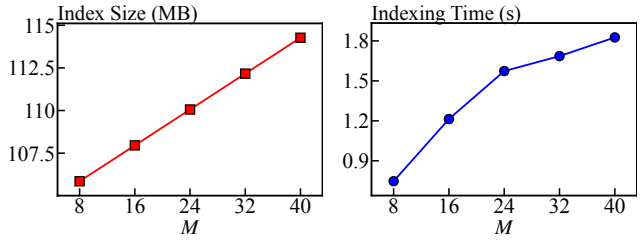


(a) Recall on MNIST (b) QPS on MNIST

Fig. 8. Effect of k on query performance.

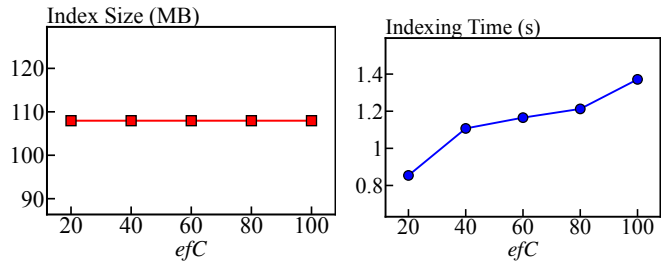
By adjusting the tolerance error δ, NAP adaptively identifies the dominant points and minimizes accuracy loss during pruning the unqualified Tail points. Compared to NAP-G and NAP-H, NAP-H identifies a larger tail region than NAP-G, achieving stronger compression at the cost of higher—but controlled—accuracy loss. For example, at $\delta = 0.001$, NAP-H classifies 41.8% of the data as tail points, compared to 18.0% for NAP-G, while both methods retain perfect 100% Top-1 accuracy. The performance difference between the two strategies stems from their underlying estimation mechanisms. The heuristic two-step NAP explicitly re-estimates the domination probability p based on the precise norm distribution, leading to more aggressive pruning when the tolerance δ is small.

8.2.2 ***Effect of NAP on APG Structure.*** To further investigate how NAP and norm domination affect the underlying APG structure, we compare the in-edge distribution across norm regions in HT-NSW, HNSW (under Euclidean space), and ip-NSW (under inner product space). The results are shown in Figure 6. HNSW maintains a nearly uniform degree distribution across all norm ranges: the in-degree concentrated within a narrow range of 12–18 (more than 70% points), with a mean degree of 14.95 and a standard deviation of 6.04. ip-NSW exhibits a strong positive correlation between node norm and in-degree. For example, small-norm nodes (norm [919, 2349]) have very low in-degree (mean 1.34, max 1.67), whereas large-norm nodes (norm [3047, 4267]) become extreme hubs with average in-degree 80.5 and a maximum of 1504. This skewed distribution confirms the norm domination effect (see Section 4.3), resulting in an imbalanced topology that harms global navigability and query performance. HT-NSW indexes only the Body points (*e.g.,* norm range in [2349, 3047] on MNIST), thereby alleviating the norm domination issue. Although the in-degree distribution of HT-NSW (mean 2.98 with standard deviation 6.04) is not as uniform as that of HNSW, it is significantly more balanced compared to ip-NSW (mean 72.02 with standard deviation 372.78). This suggests that NAP effectively redistributes connections, preventing hub overload and ensuring efficient navigation.



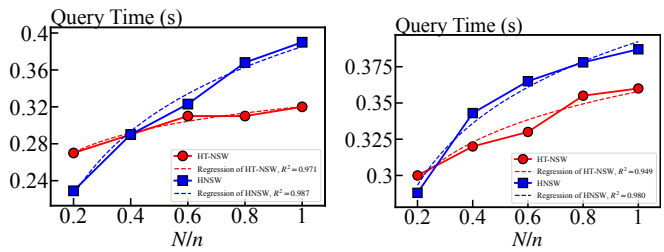
(a) Index Size on MNIST (b) Indexing Time on MNIST

Fig. 9. Effect of M on indexing performance.



(a) Index Size on MNIST (b) Indexing Time on MNIST

Fig. 10. Effect of efC on indexing performance.



(a) Query Time on Gist1M (b) Query Time on Tiny5M

Fig. 11. Query Complexity.

8.3 Self Evaluation of HT-NSW on AMIPS

In this subsection, we investigate the effects of the parameters δ, L, K, M, and efC in HT-NSW on both the indexing and query performance for AMIPS. Due to space limitations, we present experimental results on the MNIST dataset in this section.

8.3.1 ***The Effect of*** k. Figure 8 illustrates the impact of varying the number of returned results k on the query performance of HT-NSW on the MNIST dataset. We vary $k \in \{1, 10, 20, \ldots, 100\}$ while keeping all other parameters at their default settings. As k increases, the QPS of HT-NSW exhibits a gradual decline, while the recall remains remarkably stable. For instance, the recall stays approximately at 0.91–0.94 across the entire range from $k = 1$ to $k = 100$, whereas the QPS

decreases from about 210,000 at $k = 1$ to around 80,000 at $k = 100$, representing a reduction of roughly 60%. This trend is expected, as retrieving more results requires additional computation to terminate the search process.

8.3.2 ***The Effect of*** δ. We first investigate the effect of the tolerance error δ on the query performance of HT-NSW using Recall–QPS curves. As shown in Figure 7(a), HT-NSW achieves the highest query performance at $\delta = 0.001$. Both overly small and overly large values of δ degrade the query performance. For example, at a QPS of 64,000, HT-NSW achieves a recall of 0.97 at $\delta = 0.001$ but only has a recall of 0.9 at $\delta = 0.1$. Therefore, we set $\delta = 0.001$ as the default value for NAP. It indicates the balance between the NAP error and the pruning ratio on Tail (as shown in Figure 5).

8.3.3 ***The Effect of*** L ***and*** K. We further evaluate the impact of the parameters L and K in the Head index $\mathcal{I}_H$ on the performance of HT-NSW. Since there are only a small number of points in the Head (512 on MNIST), the index size and indexing time of the Head are relatively small compared to the entire dataset; therefore, we focus only on the query performance here. As shown in Figures 7(b) and 7(c), the query performance of HT-NSW remains stable as L and K increase across almost the entire QPS range (e.g., 8,000–128,000). This indicates that most of the required points can be found in $\mathcal{I}_H$ (recall of 0.97) with a small cost (about 128,000 QPS), and thus there is no need to enlarge the search region. Moreover, small values of L and K ($L = 2$, $K = 10$) are sufficient to achieve good query performance.

8.3.4 ***The Effect of*** M ***and*** efC. We analyze the effects of the parameters M and efC in the Body index $\mathcal{I}_B$ on both the indexing and query performance of HT-NSW. As shown in Figure 7(d), the query performance of HT-NSW improves as M increases. When M increases from 8 to 16, the recall at QPS = 32,000 improves from 0.95 to 0.98. This is because a larger M allows each node to connect to more neighbors, which in turn improves query performance. Similarly, as shown in Figure 7(e), increasing efC also leads to better query performance. Furthermore, we evaluate the impact of M and efC on the indexing performance of HT-NSW. As shown in Figures 9 and 10, both the index size and indexing time of HT-NSW increase with larger values of M, while increasing efC leads only to longer indexing time. This is expected, since a larger M introduces more edges per node, increasing the overall index size, while a higher efC requires more computations during graph construction. As most exact top-k MIPS results can be found in the Head, the improved quality of the Body index $\mathcal{I}_B$ has a limited effect on the overall query performance of HT-NSW. We can thus choose moderate values for M and efC to balance indexing efficiency and query effectiveness.

8.4 The Complexity of HT-NSW and HNSW

We further investigate the query time complexity of HT-NSW and HNSW by varying the dataset cardinality N to empirically demonstrate its logarithmic query complexity, as discussed in Section 7. We conduct experiments on two datasets, Gist1M and Tiny5M, by sampling subsets of different sizes (20%–100%) from the original datasets, as shown in Figure 11. Moreover, we adopt logarithmic regression to fit the query time with respect to different values of N, which yields high R^2 values (HT-NSW: 0.971 on Gist1M and 0.949 on Tiny5M, HNSW: 0.980 on Gist1M and Tiny5M). These results indicate that the query time of HT-NSW and HNSW exhibits a clear logarithmic relationship with N on both datasets. Moreover, the complexity of HT-NSW grows slightly slower than that of HNSW due to the reduced search space after applying NAP.

8.5 Comparison of HT-NSW with Baselines

8.5.1 ***Overview of Indexing Performance.*** The indexing results for all algorithms are presented in Figure 13, comparing them in terms of index size and indexing time. It is evident that HT-NSW

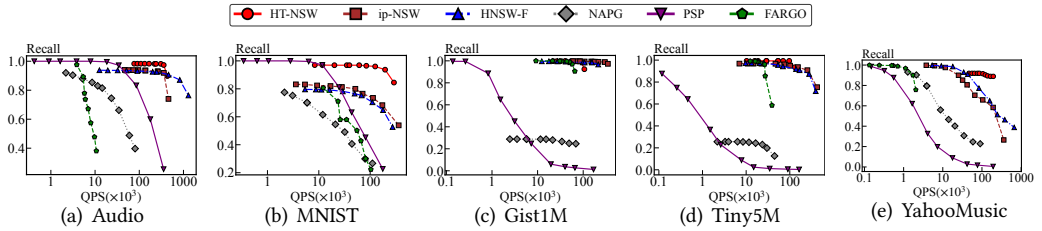


Fig. 12. Recall-QPS Curves

Table 2. Sizes of Head, Body, and Tail across all datasets.

Dataset	Head (MB)	Body (MB)	Tail (MB)
Audio	0.38 (0.97%)	7.13 (18.50%)	31.05 (80.53%)
MNIST	1.53 (0.88%)	64.46 (36.99%)	108.26 (62.13%)
Yahoo	2.34 (0.33%)	4.95 (0.69%)	707.69 (98.98%)
Gist1M	7.50 (0.20%)	481.78 (13.16%)	3172.09 (86.64%)
Tiny5M	3.00 (0.04%)	1004.12 (13.71%)	6316.80 (86.25%)

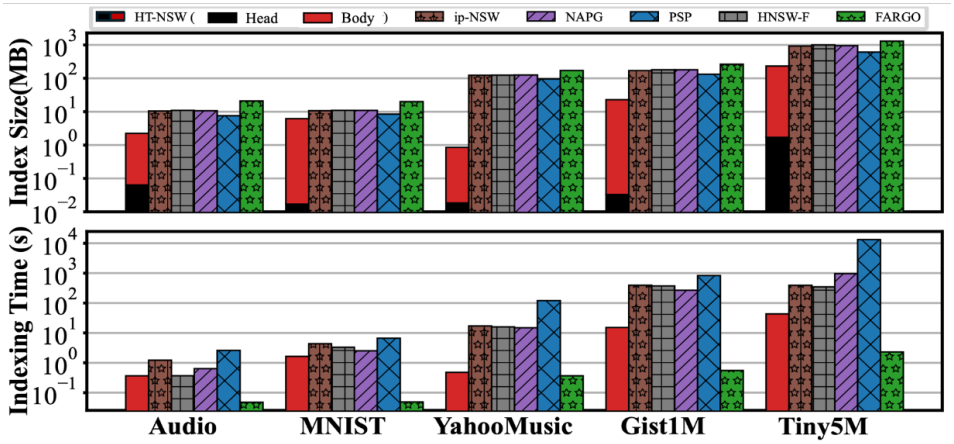
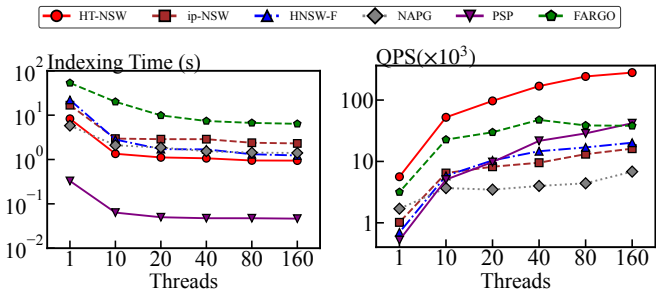


Fig. 13. Index Size and Indexing Time

achieves significantly smaller index sizes and shorter indexing times compared to the other methods. Notably, on the Yahoo dataset (i.e., YahooMusic), the index size of HT-NSW is only about 1% of that of ip-NSW. Given that HT-NSW adopts the same graph structure as ip-NSW, this performance improvement can be attributed to NAP. In HT-NSW, the index size and indexing time of $\mathcal{I}_H$ are less than 1% of those of $\mathcal{I}_B$, due to the small number of points in the Head and the low construction cost of hash indexes. The indexing time of $\mathcal{I}_H$ is even less than 0.01 s, which can be safely ignored. Table 2 provides a detailed breakdown of the sizes of the Head , Body, and Tail components of HT-NSW across all datasets, where the Tail of YahooMusic accounts for nearly 99% of the entire dataset. This observation aligns with the norm distribution depicted in Figure 2(c), where a significant portion of points exhibit small norms. By effectively pruning these Tail points, HT-NSW substantially reduces both the index size and indexing time. As for the baselines, all of them exhibit similar index sizes on the same dataset. This similarity arises because the size of the APG is largely determined by the



(a) Indexing Time on MNIST (b) QPS on MNIST

Fig. 14. Effect of thread numbers.

maximum degree, and these methods share comparable degrees. However, despite similar index sizes, NAPG exhibits significantly longer indexing times than ip-NSW on YahooMusic, Gist1M, and Tiny5M, but shorter indexing times on Audio and MNIST. The performance of NAPG, which enhances ip-NSW through a new edge selection strategy, indicates that its improvements are not universally effective.

8.5.2 ***Recall-QPS Curves.*** We assess the query performance of all algorithms using Recall–QPS curves, as shown in Figure 12. For a given QPS, an algorithm with a higher recall is considered a better AMIPS method. To obtain different QPS and recall values for these graph-based methods, we adjust the value of ef in Algorithm 4. From the figures, we observe the following: (1) Among all algorithms, HT-NSW achieves a much higher recall than the others at the same QPS, indicating superior query performance. For instance, HT-NSW attains a recall of 0.95 at a QPS of 100,000 on MNIST, while the other methods achieve recalls of less than 0.8. Furthermore, the recall of HT-NSW remains very high across all datasets. This is because HT-NSW identifies most of the AMIPS results within the Head, allowing it to achieve extremely high recall even when searching the Body at minimal cost. (2) The recall of HNSW and NAPG does not reach the level of HT-NSW, even when searching a larger number of points. This provides strong evidence of norm domination, consistent with the findings in Section 8.2.2. The recall of NAPG is particularly low on Gist1M and Tiny5M, highlighting the limitations of its improvements over HNSW. (3) PSP can mitigate the norm domination issue on Audio and MNIST, but its recall is lower than that of other algorithms at the same QPS. To achieve higher recall, PSP incurs a much larger query cost compared to HT-NSW. This demonstrates that simply increasing the number of queried points is insufficient to effectively address AMIPS, due to the out-of-distribution issue.

8.5.3 ***The Effect of Thread Numbers.*** We further investigate the effect of the number of threads on the indexing time and QPS of HT-NSW. As shown in Figure 14(a), the indexing time of HT-NSW improves as the number of threads increases from 1 to 40 (9s $\rightarrow$ 1s), but remains nearly unchanged when the number of threads further increases from 40 to 160. This is because lock contention occurs when too many threads concurrently insert points into the shared Body index $\mathcal{I}_B$. As for QPS, it improves significantly as the number of threads increases from 1 to 160, since each query can be processed independently. For the other methods, we observe similar trends regarding the effect of the number of threads on indexing time and QPS, which demonstrates that these methods can effectively leverage multi-threading to enhance both indexing and query efficiency.

9 Conclusion

This paper addresses the critical challenge of norm bias in graph-based Approximate Maximum Inner Product Search (AMIPS). We identify the norm domination phenomenon as a fundamental limitation and propose the Norm-Adaptive Partitioning (NAP) framework to mitigate it. NAP strategically partitions data into Head, Body, and Tail segments, applying tailored search strategies to each: nearly exhaustive search for Head, safe pruning for Tail, and graph-based search for Body. The main challenge of NAP strategy is to balance the query accuracy and extra cost for NAP by efficiently determining the size of Tail, Head, and Body. To address it, we further design a practical NAP algorithm that minimizes search cost while ensuring bounded accuracy. The Elbow method is employed to determine a suitable size for Head, owing to its low computational overhead. This approach led to a hybrid index, HT-NSW, that reduces the memory overhead of existing graph structures by more than 50% and achieves more than 2x query speedup while maintaining target recall. Our work demonstrates that NAP effectively unlocks the full potential of proximity graphs for efficient and scalable AMIPS. In the future, we will improve the structure of HT-NSW based on the data distribution and consider efficient index maintenance for it, making HT-NSW more suitable for modern vector databases.

Acknowledgments

The research work described in this paper was supported by Hong Kong Research Grants Council (Grants No. 16210625, T43-513/23-N), HKUST-MetaX Joint Laboratory for Advanced AI Computing (Grants No. METAX24EG01) and Natural Science Foundation of China (Grants No. 62461146205). It was partially conducted in JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Firas Abuzaid, Geet Sethi, Peter Bailis, and Matei Zaharia. 2019. To Index or Not to Index: Optimizing Exact Maximum Inner Product Search. In *ICDE*. IEEE, 1250–1261.

[2] Thomas Dybdahl Ahle, Rasmus Pagh, Ilya P. Razenshteyn, and Francesco Silvestri. 2016. On the Complexity of Inner Product Similarity Join. In *PODS*. 151–164.

[3] Daichi Amagata and Takahiro Hara. 2021. Reverse Maximum Inner Product Search: How to efficiently find users who would like to buy my item?. In *RecSys*. ACM, 273–281.

[4] Alexandr Andoni and Piotr Indyk. 2016. *LSH Algorithm and Implementation (E2LSH).* https://www.mit.edu/~andoni/LSH

[5] Imad Aouali, Amine Benhalloum, Martin Bompaire, Achraf Ait Sidi Hammou, Sergey Ivanov, Benjamin Heymann, David Rohde, Otmane Sakhi, Flavian Vasile, and Maxime Vono. 2022. Reward Optimizing Recommendation using Deep Learning and Fast Maximum Inner Product Search. In *KDD*. ACM, 4772–4773.

[6] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the Xbox recommender system using a Euclidean transformation for inner-product spaces. In *RecSys*. 257–264.

[7] Vidhisha Balachandran, Ashish Vaswani, Yulia Tsvetkov, and Niki Parmar. 2021. Simple and Efficient ways to Improve REALM. *CoRR* abs/2104.08710 (2021).

[8] Purnima Bholowalia and Arvind Kumar. 2014. EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications* 105, 9 (2014).

[9] Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*. 380–388.

[10] Lijie Chen. 2018. On The Hardness of Approximate and Exact (Bichromatic) Maximum Inner Product. *Electronic Colloquium on Computational Complexity (ECCC)* 25 (2018), 26.

[11] Tingyang Chen, Cong Fu, Kun Wang, Xiangyu Ke, Yunjun Gao, Wenchao Zhou, Yabo Ni, and Anxiang Zeng. 2025. Maximum Inner Product is Query-Scaled Nearest Neighbor. *Proc. VLDB Endow.* 18, 6 (2025), 1770–1783.

[12] Mengyao Cui et al. 2020. Introduction to the k-means clustering algorithm based on the elbow method. *Accounting, Auditing and Finance* 1, 1 (2020), 5–8.

[13] Xinyan Dai, Xiao Yan, Kelvin Kai Wing Ng, Jiu Liu, and James Cheng. 2020. Norm-Explicit Quantization: Improving Vector Quantization for Maximum Inner Product Search. In *AAAI*. 51–58.

[14] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *KDD*. ACM, 6491–6501.
[15] Marco Fraccaro, Ulrich Paquet, and Ole Winther. 2016. Indexable Probabilistic Matrix Factorization for Maximum Inner Product Search. In *AAAI*. 1554–1560.
[16] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *PVLDB* 12, 5 (2019), 461–474.
[17] Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. 2012. Locality-sensitive hashing scheme based on dynamic collision counting. In *SIGMOD*. 541–552.
[18] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB*. 518–529.
[19] Allan Gut. 2009. The multivariate normal distribution. In *An Intermediate Course in Probability*. Springer, 117–145.
[20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Augmented Language Model Pre-Training. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3929–3938.
[21] Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge. *CoRR* abs/2310.11703 (2023).
[22] Ben Harwood and Tom Drummond. 2016. FANNG: Fast Approximate Nearest Neighbour Graphs. In *CVPR*. IEEE Computer Society, 5713–5722.
[23] Qiang Huang, Guihong Ma, Jianlin Feng, Qiong Fang, and Anthony K. H. Tung. 2018. Accurate and Fast Asymmetric Locality-Sensitive Hashing Scheme for Maximum Inner Product Search. In *KDD*. 1561–1570.
[24] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*. 604–613.
[25] Alexis Joly and Olivier Buisson. 2011. Random maximum margin hashing. In *CVPR*. IEEE Computer Society, 873–880.
[26] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: Fast and Exact Inner Product Retrieval in Recommender Systems. In *SIGMOD Conference*. ACM, 835–850.
[27] Jie Liu, Xiao Yan, Xinyan Dai, Zhirong Li, James Cheng, and Ming-Chang Yang. 2020. Understanding and Improving Proximity Graph Based Maximum Inner Product Search. In *AAAI*. 139–146.
[28] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. 2007. Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search. In *VLDB*. 950–961.
[29] 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.
[30] Stanislav Morozov and Artem Babenko. 2018. Non-metric Similarity Graphs for Maximum Inner Product Search. In *NeurIPS*. 4726–4735.
[31] Behnam Neyshabur and Nathan Srebro. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *ICML*, Vol. 37. 1926–1934.
[32] Hossein Pishro-Nik. 2014. *Introduction to probability, statistics, and random processes*. Kappa Research, LLC Blue Bell, PA, USA.
[33] Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning Binary Codes for Maximum Inner Product Search. In *ICCV*. 4148–4156.
[34] Anshumali Shrivastava. 2015. Improved Asymmetric Locality Sensitive Hashing (ALSH) for Maximum Inner Product Search (MIPS). In *UAI*. AUAI Press, 812–821.
[35] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NIPS*. 2321–2329.
[36] Suhas Jayaram Subramanya, Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, and Rohan Kadekodi. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS*. 13748–13758.
[37] Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, and Ping Li. 2021. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In *KDD*. ACM, 1552–1560.
[38] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. 2019. On Efficient Retrieval of Top Similarity Vectors. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 5235–5245.
[39] Christina Teflioudi and Rainer Gemulla. 2017. Exact and Approximate Maximum Inner Product Search with LEMP. *ACM Trans. Database Syst.* 42, 1 (2017), 5:1–5:49.
[40] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2022. DB-LSH: Locality-Sensitive Hashing with Query-based Dynamic Bucketing. In *ICDE*. 2250–2262.
[41] Mo Tiwari, Ryan Kang, Jaeyong Lee, Donghyun Lee, Christopher Piech, Sebastian Thrun, Ilan Shomorony, and Martin Jinye Zhang. 2024. Faster Maximum Inner Product Search in High Dimensions. In *ICML*. OpenReview.net.
[42] Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press.

[43] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD*. ACM, 2614–2627.

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

[45] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. 2018. Norm-Ranging LSH for Maximum Inner Product Search. In *NeurIPS*. 2956–2965.

[46] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. 2023. Towards Efficient Index Construction and Approximate Nearest Neighbor Search in High-Dimensional Spaces. *Proc. VLDB Endow.* 16, 8 (2023), 1979–1991.

[47] Xi Zhao, Bolong Zheng, Xiaomeng Yi, Xiaofan Luan, Charles Xie, Xiaofang Zhou, and Christian S. Jensen. 2023. FARGO: Fast Maximum Inner Product Search via Global Multi-Probing. *Proc. VLDB Endow.* 16, 5 (2023), 1100–1112.

[48] Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. 2019. Möbius Transformation for Fast Inner Product Search on Graph. In *NeurIPS*. 8216–8227.

Received October 2025; revised January 2026; accepted February 2026