Maximum Inner Product is Query-Scaled Nearest Neighbor

Tingyang Chen
Zhejiang University
chenty@zju.edu.cn

Cong Fu
Shopee Pte. Ltd.
fc731097343@gmail.com

Kun Wang
Shopee Pte. Ltd.
wk1135256721@gmail.com

Xiangyu Ke
Zhejiang University
xiangyu.ke@zju.edu.cn

Yunjun Gao
Zhejiang University
gaoyj@zju.edu.cn

Wenchao Zhou
Alibaba Group
zwc231487@alibaba-inc.com

Yabo Ni
Nanyang Technological University
yabo001@e.ntu.edu.sg

Anxiang Zeng
Nanyang Technological University
zeng0118@ntu.edu.sg

ABSTRACT

Maximum Inner Product Search (MIPS) for high-dimensional vectors is pivotal across databases, information retrieval, and artificial intelligence. Existing methods either reduce MIPS to Nearest Neighbor Search (NNS) while suffer from harmful vector space transformations, or attempt to directly tackle MIPS but struggle to mitigate redundant computations due to the absence of the triangle inequality. This paper presents a novel theoretical framework that equates MIPS with NNS without requiring space transformation, thereby allowing us to leverage advanced graph-based indices for NNS and efficient edge pruning strategies, significantly reducing unnecessary computations. Despite a strong baseline set by our theoretical analysis, we identify and address two persistent challenges to further refine our method: the introduction of the **P**roximity Graph with **S**pherical **P**athway (PSP), designed to mitigate the issue of MIPS solutions clustering around large-norm vectors, and the implementation of **A**daptive **E**arly **T**ermination (AET), which efficiently curtails the excessive exploration once an accuracy bottleneck is reached. Extensive experiments reveal the superiority of our method over existing state-of-the-art techniques in search efficiency, scalability, and practical applicability, achieving an average 35% speed-up in query processing and a 3× reduction in index size compared with state-of-the-art graph based methods. Notably, our approach has been validated and deployed in search engines of Shopee, a shopping APP. Our code and an industrial-scale dataset for offline evaluation will also be released to address the absence of e-commerce data in public benchmarks.

PVLDB Reference Format:
Tingyang Chen, Cong Fu, Kun Wang, Xiangyu Ke, Yunjun Gao, Wenchao Zhou, Yabo Ni, and Anxiang Zeng. Maximum Inner Product is Query-Scaled Nearest Neighbor. PVLDB, 18(1): XXX-XXX, 2025.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/TiyCHEN/PSP.

Proceedings of the VLDB Endowment, Vol. 18, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Maximum Inner Product Search (MIPS) is essential across various artificial intelligence and information retrieval applications [6, 34, 56, 83]. The growing demand for managing and retrieving large-scale, high-dimensional data — particularly fueled by advancements of large language models [80] and retrieval-augmented generation [6] — has garnered significant attention within the database community [24, 49, 75]. However, efficiently implementing exact MIPS remains a formidable challenge [13, 29, 65, 71]. In response, there has been a shift towards approximate MIPS, which trades minimum accuracy for substantial gain in speed.

For the approximate MIPS problem, two primary paradigms have emerged. The first focuses on the Inner Product (IP) metric, establishing specialized theoretical frameworks to tackle MIPS challenges [12, 23, 25, 39, 46, 62, 78]. However, the absence of triangle inequality in the IP space hinders them from efficiently reducing redundant computations [62], particularly in recent promising graph-based methods [39, 46]. This deficiency impacts query performance and increases memory usage, as these methods lack theory foundations to prune edges as effectively as advanced NNS graphs [18, 19, 44]. Specifically, the widely-used greedy graph search algorithm (Algorithm 1) necessitates checking all neighbors at each step to approach the query more closely during traversal, often leading to unnecessary checks that degrade efficiency [39, 62, 68].

The second paradigm addresses the MIPS problem by reformulating it as a NNS problem, enabling the use of established NNS methods [37, 47, 57, 58, 74, 81, 84] that exploit the triangle inequality for efficient computation pruning. These methods typically involve various vector space transformations, which, while effective to some extent, often rely on strong theoretical assumptions [84], can introduce potential structural distortions [46, 81], and exhibit inefficiencies in data updating (elaborated in §2). These limitations can significantly restrict their practical applicability.

Our research explores a fundamental question: *Can we leverage the efficient computation reduction properties without altering the vector space?* We begin with an intriguing observation derived from comparing the search routing behaviors of MIPS and NNS on the same Euclidean proximity graph, such as an MSNET [19]. Specifically, using Algorithm 1, the search iteratively moves from a node to its neighbor, minimizing the distance to the query under given metrics. As shown in Figure 1, the search paths under MIPS and NNS *initially overlap*. The main discrepancy arises when MIPS *moves beyond the query into outer spherical areas of larger norms*. According to the homogeneity of the IP metric, i.e., $\langle \mu q, p \rangle = \mu \langle q, p \rangle$,

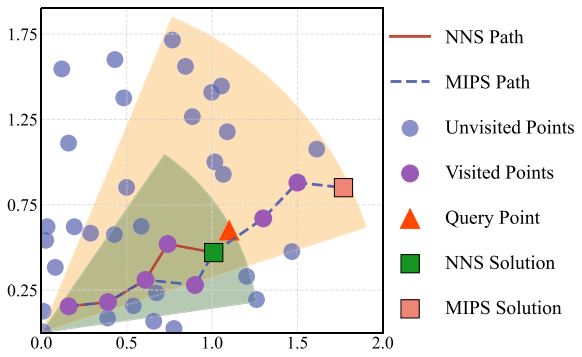

Figure 1: An illustration of the overlaps and discrepancies between the paths of NNS and MIPS on a Euclidean proximity graph. The overlaps hold mainly before the norms of explored nodes exceed the norm of the query.

their paths and solutions coincide when a scalar scales up q until its norm exceeds the MIPS solution.

We theoretically verify the above observations by showing that MIPS for query q can be equated to NNS for a scaled query $q' = \mu q$ on an MSNET, via introducing a proper scalar μ. This equality enables using advanced NNS graph indices for MIPS without altering the vector space, thus avoiding the associated drawbacks. Additionally, this breakthrough lays the groundwork for the first theoretical analysis of search complexity in the graph-based MIPS context. Our experiments on real-world datasets further validate this theory (§5). Despite these advancements, two practical challenges persist: the tendency of MIPS solutions to congregate in large norm regions and the excessive exploration that occurs once a precision bottleneck is reached. We propose an innovative index, Proximity Graph with Spherical Pathway (PSP), to handle the biased distribution of MIPS solutions. Additionally, we introduce a novel Adaptive Early Termination (AET) mechanism to mitigate excessive explorations.

Contributions. Our contributions are highlighted as follows:

(1) Theoretical Foundation: This work pioneers in establishing comprehensive theoretical foundations for graph-based MIPS.

(2) New Practical Solution: We introduce PSP, a scalable graph index with efficient spherical pathways and an adaptive entry point selection mechanism to address the biased distribution of MIPS solutions. We propose an adaptive early termination mechanism to mitigate the excessive exploration problem.

(3) Extensive Experiments on eight real-world datasets validate the theoretical soundness, efficiency, and scalability of this work, highlighting an average 35% speed-up in query process and a 3× reduction in index size over leading graph-based methods. Our method is the only one reaching 95% recall under 5ms and thus has been deployed in the large-scale search engines of Shopee, which showcases the application viability.

(4) New Industry-scale Dataset: We release a new dataset derived from industry-scale traffic data collected from the Shopee search scenario. Our dataset fills the absence of e-commercial modality in public benchmarks, facilitating research in MIPS and NNS.

Roadmap. §2 introduces the background of the MIPS problem and two main paradigms; §3 presents our theoretical foundations; §4 introduces practical solutions (PSP and AET); §5 outlines the research questions and provides a comprehensive analysis based on the experimental outcomes; §?? revisits the study, discussing its limitations and potential topics in future work; §6 reviews previous related works for broader interests and §7 concludes this paper.

2 PRELIMINARIE AND BACKGROUND

The Inner Product (IP) serves as a fundamental and effective similarity metric, widely used in artificial intelligence and machine learning [6, 53]. With the proliferation of data represented and stored in high-dimensional vectors [24, 49, 80], the importance of Maximum Inner Product Search (MIPS) has become increasingly pronounced [25, 46, 81]. The MIPS problem is formally defined as:

Definition 1 (Maximum Inner Product Search (MIPS)). *Given a query $q \in \mathbb{R}^d$, and a dataset $\mathcal{D} \subset \mathbb{R}^d$, the* MIPS *problem aims to find a vector $p \in \mathcal{D}$ that maximizes the inner product with q:*

$$p = \arg\max_{p \in \mathcal{D}} \langle p, q \rangle \tag{1}$$

Several approaches have been proposed to solve the MIPS problem in sublinear time [5, 36, 64]. However, these methods often encounter practical limitations due to inefficiencies in query processing. Consequently, researchers have explored approximate MIPS methods, which trade a minimum accuracy loss for significantly improved retrieval efficiency. Formally, we have:

Definition 2 (ϵ-Maximum Inner Product Search (ϵMIPS)). *Given a query $q \in \mathbb{R}^d$, a dataset $\mathcal{D} \subset \mathbb{R}^d$, and an approximation ratio $\epsilon \in (0, 1)$, let $p^* \in \mathcal{D}$ be the* MIPS *solution of q, the* ϵMIPS *problem aims to find a vector $p \in \mathcal{D}$ satisfying $\langle p, q \rangle \geq \epsilon \cdot \langle p^*, q \rangle$.*

While extending these definitions to k-MIPS and k-ϵMIPS to find the top k solutions is straightforward, we omit these details here for brevity. Note that the main difference between MIPS and a similar field, Nearest Neighbor Search (NNS), lies in the metric used [49] (inner product and distance metric like Euclidean distance, respectively)[1]. Recent efforts aim to address both ϵNNS and ϵMIPS problems with similar strategies: reducing the cost of distance calculations and pruning the search space [21, 25, 81, 82]. To better motivate our study, we propose revisiting the ϵMIPS techniques from a novel perspective: *whether they transform the vector space.*

Non-transformation-based methods adopt the Inner Product (IP) metric for both index structures and search algorithms [23, 25, 39, 46, 62, 78]. While these methods preserve the integrity of information in the original space, they often suffer from large indices and inferior search performance due to their inability to reduce redundant computations effectively, unlike ϵNNS methods. This inefficiency primarily stems from the absence of properties analogous to the triangle inequality in the Euclidean metric. In contrast, the triangle inequality significantly reduces the search space in ϵNNS under the Euclidean metric [70].

[1] In real-world applications, MIPS is often used in recommendation systems and information retrieval, where relevance is key [27]. NNS is prevalent in tasks like image recognition and clustering, where spatial proximity matters [51].

Table 1: Table of notations

Symbol	Meaning
$\mathbb{R}^d$	A d-dimension space
$G = (V, E)$	A directed graph with nodes V and edges E
$\delta(\cdot,\cdot)$	l_2 distance between two vectors
$\delta_\perp(p, H)$	l_2 distance from a point p to a hyperplane H
$\langle p, q\rangle$	The inner product between p and q
$\|\cdot\|$	The l_2 norm of a vector
θ	The angle between two certain vectors
μ	A fixed scaling scalar
$sup(\cdot)$	The upper bound value of a set

Transformation-based methods, inspired by the successes in NNS, attempt to reframe the MIPS problem into an NNS problem through two typical space transformation techniques:

Möbius transformation [84] aims to establish an isomorphism between the Delaunay graph for the IP metric [46] and that for the Euclidean metric [43] by normalizing the vectors:

Definition 3 (Möbius Transformation). *Given a database $\mathcal{D} \subset \mathbb{R}^d \backslash \{0\}$, the möbius transformation modifies $p \in \mathcal{D}$ by $p/\|p\|$.*

The isomorphism established by Möbius transformation comes with strong assumptions, such as requiring the origin point to be within $\mathcal{D}$'s convex hull and the data to be independently and identically distributed (IID), which are often violated in practice. The origin-contained assumption could easily be violated with one-hot or multi-hot representations. Verifying that the origin is not contained in such convex hull is straightforward. Additionally, the IID assumption is often violated because the representations generated by ML models are usually correlated in different dimensions [53, 55].

XBOX transformation [81] adopts an asymmetric approach by elevating both the dataset and query vectors to higher dimensions:

Definition 4 (XBOX Transformation). *Given a database $\mathcal{D} \subset \mathbb{R}^d$, and a query q. The XBOX transformation maps $\forall p \in \mathcal{D}$ to $p' = \left[p; \sqrt{M^2 - \|p\|}\right]$, and maps $\forall q \in \mathcal{R}^d$ to $q' = [q; 0]$: where $[;]$ denotes vector concatenation and $M \geq \max_{p\in\mathcal{D}} ||p||$.*

Thus, we have $\langle p, q\rangle = \frac{1}{2}\left(\|q\| + M^2 - \delta(p', q')\right)$, a conversion from IP to Euclidean with constant offset. Although this transformation is widely used [28, 32, 47, 50, 74], it has significant drawbacks like potential structural distortions [46] and difficulties in determining an appropriate M: A large M may reduce the distinguishability of different points because it contributes significantly to $\delta(P(p), Q(q))$. Conversely, a small M may cause inflexibility in data updates when the new points' norms exceed M.

Graph-based methods for MIPS is the focus of this paper. While previous approaches vary in their graph index structures, they predominantly employ a similar search algorithm, often referred to as a greedy walk (Algorithm 1). This algorithm iteratively moves to a neighbor of the current node that is closer to the query, with the search complexity dominated by the **walking steps and the graph's average out-degree [19]**. Notably, non-transformation graph-based methods have struggled with implementing efficient edge pruning strategies due to the absence of the triangle inequality, leading to graphs with high average degrees and subsequently increased search complexity. In contrast, advanced graph-based NNS methods [18, 19, 44] can dramatically reduce edge quantity while maintaining graph connectivity and short search paths, significantly enhancing efficiency. In the following section, this paper will explore how to leverage these properties for the MIPS problem by providing theoretical foundations to align graph-based MIPS with NNS *without vector space transformations*.

Algorithm 1: Greedy Search For Graphs[52]

Input: Dataset $\mathcal{D}$, Graph G, query q, candidate set size l_s, result set size k.
Output: Top k result set R.

```
R ← ∅; q' ← μq; Q ← ∅;
P ← random sample l_s nodes from G;
for each node p in P do
    Q.add(p,δ(p, q'));
Q.make_min_heap(); R.init_max_heap() ▷ compare distances
while Q.size() do
    p ← Q.pop()[0]; R.insert((p, δ(p, q')))
    if visited(p) then
        continue;
    N_p ← neighbors of p in G
    Q ← batch_insert(Q, N_p)
    Q.resize(l_s);R.resize(k)   ▷ delete larger-distance nodes
return R
```

3 THEORETICAL FOUNDATIONS

In this section, we aim to *equate* MIPS *and* NNS *theoretically on Euclidean proximity graphs*. This alignment allows us to capitalize on the beneficial features of established graph-based NNS methods, such as efficient edge pruning, strong connectivity, and robust theoretical guarantees, to enhance graph-based MIPS and analyze its search behavior. Formally, a proximity graph is defined as:

Definition 5 (Proximity Graph). *Given a dataset $\mathcal{D}$, a proximity graph G on $\mathcal{D}$ can be denoted by $G = (V, E)$, where V is the node set, and E is the edge set. Each node $v_i \in V$ represents a vector $p_i \in \mathcal{D}$, while $(v_i, v_j) \in E$ if and only if (v_i, v_j) satisfies a given criteria.*

Within our theoretical alignment, the specific type of proximity graph—whether a Delaunay graph [7], a Navigating Small World (NSW) graph [44], or a Monotonic Relative Neighborhood Graph (MRNG) [18]—is not fixed. *The choice of graph does not affect the validity of our theory.* Notably, part of our theory also is *not limited to graph-based context* (Theorem 1) and can be applied to any type of MIPS methods, such as hashing and quantization.

3.1 Equivalence under Scaled Query

First, we establish a mapping $q' = f(q)$, such that **(1)** the MIPS solutions for q' align with those for q, and **(2)** the NNS solutions for q' align with the MIPS solutions for q. This mapping enables the use of efficient NNS algorithms to solve the MIPS problem without changing the origin vector space. We leverage the homogeneity of IP metric, as formalized in the following lemma:

Lemma 1. *Given a vector database $\mathcal{D} \subset \mathbb{R}^d$, and a scalar $\mu > 0$, the MIPS solution for q is identical to those for $q' = \mu q$ within $\mathcal{D}$.*

Proof. Let $p^* \in \mathcal{D}$ be a MIPS solutions for q, satisfying $\forall p \in \mathcal{D}\backslash\{p^*\}, \langle p^*, q\rangle \geq \langle p, q\rangle$. Scaling q by $\mu > 0$, we have:

$$\langle p^*, \mu q\rangle = \mu\langle p^*, q\rangle \geq \mu\langle p, q\rangle = \langle p, \mu q\rangle, \forall p \in \mathcal{D}\backslash\{p^*\} \quad (2)$$

Therefore, $q' = \mu q$ and q share the same MIPS solutions. □

Lemma 1 reveals that non-negative constant scaling of the query vector preserves MIPS solutions. Next, we will align MIPS and NNS with this scaling mapping by identifying solution space for μ.

Theorem 1. *Given a vector database $\mathcal{D} \subset \mathbb{R}^d$ and $\forall q \in \mathbb{R}^d$, there exists a scalar $\bar{\mu}$ such that for $\forall \mu > \max(\bar{\mu}, 0)$, the nearest neighbor of $q' = \mu q$ in $\mathcal{D}$ aligns with the* MIPS *solution for q.*

Proof. We exclude cases where the zero-vector or multiple identical vectors in $\mathcal{D}$ complicate the analysis. The IP of the zero vector with any vector equals 0, which lacks practical meaning. Then, we break down the proof step by step.

Case 1: Assume there is a unique MIPS solution $p^* \in \mathcal{D}$ for q. By Lemma 1, p^* is also a MIPS solution for $q', \forall \mu > 0$

To ensure p^* is a nearest neighbor of $q' = \mu q$, we need to solve:

$$\|p^* - q'\| < \|p - q'\|, \quad \forall p \in \mathcal{D}, p \neq p^*$$

By simplifying and rearranging, we find the condition for μ:

$$\mu > \frac{\|p^*\|^2 - \|p\|^2}{2(\langle p^*, q\rangle - \langle p, q\rangle)}, \forall p \in \mathcal{D}, p \neq p^* \quad (3)$$

Let $\bar{\mu} = \sup\left(\left\{\frac{\|p^*\|^2 - \|p\|^2}{2(\langle p^*, q\rangle - \langle p, q\rangle)} \middle| \forall p \in \mathcal{D}, p \neq p^*\right\}\right)$. This space is bounded because $\mathcal{D}$ is a finite set; thus, $\bar{\mu}$ exists. Unifying conditions for μ, p^* is a nearest neighbor of q' when $\mu > \max(\bar{\mu}, 0)$.

Case 2: Considering there may exist multiple MIPS solutions for $q \in \mathcal{D}$, let $\mathcal{P}^* = \{p^* | \langle p^*, q\rangle \geq \langle p, q\rangle, \forall p \in \mathcal{D}, p \neq p^*\}$ denote this solution set. We can find a $p_n^* \in \mathcal{P}^*$ such that p_n^* is the nearest neighbor of $q' = \mu q$ with an appropriate μ. By following a similar procedure in Case 1, we can get the solution space for μ as:

$$\mu > \max(\bar{\mu}, 0) = \max(\sup\left(\left\{\frac{\|p^*\|^2 - \|p\|^2}{2(\langle p^*, q\rangle - 2\langle p, q\rangle)} | p \in \mathcal{D} \setminus \mathcal{P}^*\right\}\right), 0)$$

Summarizing Case 1 and 2, we can identify a scalar boundary $\bar{\mu}$ such that $\forall \mu > \max(\bar{\mu}, 0)$, there exists a point $p^* \in \mathcal{D}$ which is the nearest neighbor of $q' = \mu q$ and also a MIPS solution of q. □

By Theorem 1, we establish the existence of a scalar μ and a corresponding node p^* that *unifies the solutions for the* NNS *and* MIPS *problems* for a scaled query $q' = \mu q$. This identification alone does not ensure we can retrieve p^* under the IP metric as efficiently as NNS algorithms on a Euclidean proximity graph. To address this, we aim to demonstrate *the equivalence of search behavior* under IP and Euclidean metrics within the same graph structure:

Theorem 2. *Given a proximity graph $G = (V, E)$ and a query $q \in \mathbb{R}^d$, when using the standard Graph Nearest Neighbor Search (*GNNS*) [52] algorithm to decide the* MIPS *solution for q, there exists a scalar $\bar{\mu}$ such that for $\forall \mu > \max(\bar{\mu}, 0)$, we can identify a node $p^* \in \mathcal{N}_o$ that satisfies: $p^* \in \left\{\arg\max_{p \in \mathcal{N}_o} \langle p, q\rangle\right\} \cap \left\{\arg\min_{p \in \mathcal{N}_o} \delta(p, q')\right\}$. Here, o can be any node on the search path of* GNNS *regarding the query $q' = \mu q$, and $\mathcal{N}_o = \{p | (p, o) \in E\}$ denotes o's neighbor nodes.*

Proof Sketch. To align the search paths under both metrics as per Algorithm 1, we can unify the node-selection behavior by localizing the problem: solving for the optimal p^* among the neighbors at each greedy step, akin to the conditions established in Theorem 1. Let l be the number of steps in the path. Define $U = \{\bar{\mu}_i | 1 \leq i \leq l\}$. it's not hard to verify that there exists a scalar boundary $\sup(U \cup \{0\})$ forces the Algorithm 1 under both IP and Euclidean metrics to generate the same search path targeting the scaled query μq. The complete proof is in [4]. □

Remarks. From Theorem 1 and Theorem 2, we discern a compelling duality between the MIPS and NNS paradigms. These findings suggest that by appropriately scaling the query q, we can modulate the overlap in their search paths on a Euclidean proximity graph. *By choosing an appropriate μ, the search behavior becomes invariant to the metric used.* Our experimental validations further corroborate this theoretical insight on real-world datasets (§5.2).

Notably, in practical applications, it is unnecessary to calculate a specific μ to tackle the MIPS problem under the Euclidean metric (Lemma 1). Theorem 1 and 2 further demonstrate that IP metric can be directly applied on a Euclidean proximity graph to solve the MIPS problem. Consequently, the only modifications needed are to adjust the metric in Algorithm 1 to IP and adapt the algorithm to retain candidates and answers with the largest IP values to q.

3.2 Efficiency Analyses for Graph-Based MIPS

While our theoretical framework permits the use of any Euclidean Proximity Graph, only a few, such as MRNG [19] and SSG [18], come with robust theoretical guarantees. To overcome the high-degree issue prevalent in state-of-the-art non-transformation graph-based methods [39, 46], we aim to identify a sparse Euclidean graph that ensures a low amortized search complexity. In this paper, we focus on SSG for its sparsity and additional theoretical assurances for queries absent from the database $\mathcal{D}$.

1-MIPS Analysis. We begin with top-1 MIPS of SSG formally as:

Theorem 3. *Consider a vector database $\mathcal{D} \subset \mathbb{R}^d$ containing n points, where n is sufficiently large to ensure robust statistical properties. Let $\mathcal{G}$ be a SSG [18] defined on $\mathcal{D}$. Assume that the base and query vectors are independently and identically distributed (i.i.d.) and are drawn from the same Gaussian distribution, with each component having zero mean and a variance σ^2. The expected length of the search path L from any randomly selected start node p to a query $q \in \mathbb{R}^d$ can be bounded by $\mathbb{E}[L] < c_0 \frac{\log n + c_1 d}{\log R + c_2 d}$, where R is the max-degree of all possible $\mathcal{G}$, independent with n [18], and c_0, c_1, c_2 are constants.*

Proof Sketch. Given Theorem 1, 2, along with the monotonic search property of SSG [18], Algorithm 1 identifies a greedy search path where each step minimizes the Euclidean distance to μq while maximizes the IP distance with respect to q, i.e. $\langle r_i, q\rangle > \langle r_{i-1}, q\rangle$. The expected length of the search path can be calculated as:

$$\mathbb{E}[L] = \mathbb{E}_{p \in \mathcal{D}, q \in \mathbb{R}^d}\left[\frac{\langle r_{mip}, q\rangle - \langle p, q\rangle}{\mathbb{E}[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q]}\right] \quad (4)$$

where r_{mip} is the MIPS solution of q. The outer expectation cannot be simplified directly due to potential dependencies among p, r_i, and q. Given that the base and query vectors are finite and drawn from the same distribution, we can enclose them in a hypersphere

of diameter $2H$ [72], which is independent of p, r_i, and q. Since $\langle p, q\rangle < H^2$ and $\langle r_{mip}, q\rangle < H^2$, we can derive:

$$\mathbb{E}[L] < \frac{2H^2}{\mathbb{E}_{p\in\mathcal{D},q\in\mathbb{R}^d}\left[\mathbb{E}\left[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q\right]\right]} \tag{5}$$

Although the exact solution to this inequality is intractable, a practical approach involves approximating both the numerator and the denominator. By applying Extreme Value Theory (EVT) [59], we can derive a tight upper bound for this approximation, ensuring convergence to the bound as n becomes sufficiently large.

As H is the half-diameter, H^2 is the maximum squared norm of the dataset $\mathcal{D}$. We define $M_0 = H^2 = \max_1^n\{X_1^2, ..., X_n^2\}$, where X_i are element-wise i.i.d. samples from $\mathcal{N}(0, \sigma^2)$. Consequently, X^2 follows a chi-square distribution with d degrees of freedom. The Moment Generating Function (MGF) [15] of M_0 is given:

$$MGF(t)_{X^2} = (1 - 2\sigma^2 t)^{-\frac{d}{2}}, 0 < t < \frac{1}{2\sigma^2} \tag{6}$$

From Jensen's Inequality [45] with $\phi(x) = e^{t_0 x}, t_0 > 0$, we have:

$$\begin{aligned} e^{t_0\mathbb{E}[M_0]} &\le \mathbb{E}[e^{t_0 M_0}] = \mathbb{E}\max_{i=1}^{n} e^{t_0 X_i^2} \\ &\le \sum_1^n \mathbb{E}\left[e^{t_0 X_i^2}\right] = nMGF(t_0)_{X^2} \end{aligned} \tag{7}$$

$$\mathbb{E}[M_0] \le \frac{1}{t_0}\left(\log n + \frac{d}{2}\log(1 - 2\sigma^2 t_0)^{-1}\right) \tag{8}$$

Here, t_0 is a hyper-parameter that can be optimized within the range $(0, 0.5)$ to enhance the tightness of this bound.

A similar approach can be applied to the random variable XY for approximating the denominator of Eq. 5, where X and Y element-wise i.i.d. sampled from $\mathcal{N}(0, \sigma^2)$. In this case, XY follows a distribution characterized by a modified Bessel function of the second kind [10]. Then we can have:

$$\mathbb{E}[\langle r_i, q\rangle] \le \mathbb{E}[\langle r_{i-1}, q\rangle] + \frac{1}{t_1}\left(\log R + d\log(1 - \sigma^2 t_1^2)^{-1}\right) \tag{9}$$

Substituting them into Eq.(5), we have:

$$\begin{aligned} \mathbb{E}[L] &< \frac{2H^2}{\mathbb{E}_{p\in\mathcal{D},q\in\mathbb{R}^d}\left[\mathbb{E}\left[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q\right]\right]} \\ &\approx \frac{\frac{2}{t_0}\left(\log n + \frac{d}{2}\log(1 - 2\sigma^2 t_0)^{-1}\right)}{\frac{1}{t_1}\left(\log R + d\log(1 - \sigma^2 t_1^2)^{-1}\right)} \end{aligned} \tag{10}$$

where R is the maximum degree of $\mathcal{G}$, independent of n and choices of p, q, and r_i[18]. Thus, the outer expectation can be eliminated. When t_0 and t_1 are optimized for the tightest approximation, introducing constants c_0, c_1 simplifies the formula to:

$$\mathbb{E}[L] < c_0\frac{\log n + c_1 d}{\log R + c_2 d} \tag{11}$$

The complete proof is in [4]. □

While Theorem 3 provides an approximation for the expected search path length, the results align well with intuitive insights: the length L increases gradually with n while decreases with graph max-degree R. In SSG, the inter-edge angle α governs the graph sparsity [18]. A smaller α leads to higher R, enhancing graph connectivity and thereby reducing inter-node transition costs. Moreover, the overall search complexity for top-1 MIPS can be bounded by $O\left(c_0 R\frac{\log n + c_1 d}{\log R + c_2 d}\right)$, indicating that the complexity increases more rapidly with R than with n, highlighting the critical role of graph pruning for better searching efficiency.

k-MIPS Analysis. The preceding theory examines the validity and efficiency of tackling the 1-MIPS problem on an SSG. Next, We demonstrate that high-confidence k-MIPS solutions can be obtained by exploring the neighborhoods of the 1-MIPS solution.

Theorem 4. *Given a vector database $\mathcal{D} \subset \mathbb{R}^d$ consisting of n points, where n is sufficiently large to ensure robust statistical properties. For the ease of calculation, we assume base and query vectors are element-wise i.i.d. and are drawn from the same Gaussian distribution, parameterized with zero mean and a variance σ^2. Given a tunable parameter $s > 0$, we have $\langle r, q\rangle$ is lower bounded by $|\|p\| - s|\,\|q\|\cos\left(\theta_{pq} + 2\sin^{-1}\frac{s}{2\|p\|}\right)$ with probability $Q(s) = \frac{\gamma\left(\frac{d}{2}, \frac{s}{2\sigma^2}\right)}{\Gamma\left(\frac{d}{2}\right)}$, where $\Gamma(\cdot)$ is the gamma function and $\gamma(\cdot)$ is the lower incomplete gamma function.*

Proof. For any query q and any point $r \in \mathcal{D}$, this proof aims to establish their relationship regarding q's top-1 MIPS solution p, We begin by expand $\langle r, q\rangle$ for r and q as:

$$\langle r, q\rangle = \|r\|\|q\|\cos\theta_{rq} \tag{12}$$

, where θ_{rq} is the angle between vector r and q. Under the same norm of r, $\cos\theta_{rq}$ w.r.t. p is minimized when normalized r, p, and q fall on the same great circle of the unit sphere. According to spherical trigonometry cosine rule, above equation is bounded as:

$$\langle r, q\rangle \ge \|r\|\|q\|\cos\left(\theta_{rp} + \theta_{pq}\right) \tag{13}$$

With triangle inequality, we further have:

$$\langle r, q\rangle \ge \left|\|p\| - \Delta_{pr}\right|\|q\|\cos\left(\theta_{rp} + \theta_{pq}\right) \tag{14}$$

, where Δ_{pr} is the L_2 distance between point p and r. We can bound Δ_{pr} according to the distribution of Δ_{pr}. Specifically, let $Z = \|X - Y\|^2$ denote the distribution of Euclidean distance between X and Y, i.i.d. drawn from $\mathcal{D}$. Z follows a gamma distribution parameterized as :

$$Z = \|X - Y\|^2 \sim Gamma\left(\frac{d}{2}, 2\sigma^2\right) \tag{15}$$

The probability that $\mathbb{P}[Z < s]$ for a given threshold s can be derived from the Cumulative Density Function (CDF) as:

$$\mathbb{P}[Z < s] = Q(s) = \frac{\gamma\left(\frac{d}{2}, \frac{s}{2\sigma^2}\right)}{\Gamma\left(\frac{d}{2}\right)} \tag{16}$$

With a given Δ_{pr}, the maximum of θ_{rp} can also be calculated as $\max\theta_{rp} = 2\sin^{-1}\frac{s}{2\|p\|}$. Combining Equation (14) and (16), we derive the lower bound for $\langle r, q\rangle$ as :

$$\langle r, q\rangle > |\|p\| - s|\,\|q\|\cos\left(\theta_{pq} + 2\sin^{-1}\frac{s}{2\|p\|}\right) \tag{17}$$

, with probability $Q(s)$ in Equation (16). □

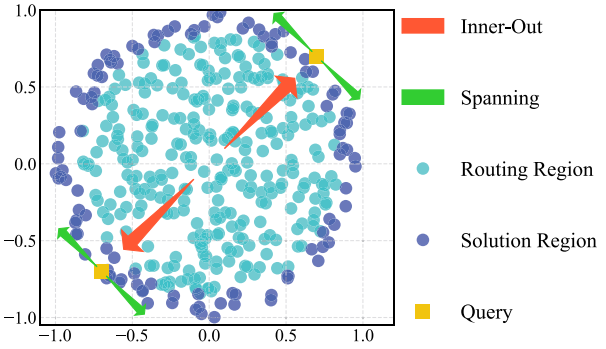

Figure 2: An illustration of biased solutions for MIPS problems. Spherical Pathways benefit the routing from the inner to the outer ring and the spherical spanning for k answers.

Algorithm 2: Graph Index Construction

Input: Dataset $\mathcal{D}$, candidate size L, maximum degree R, minimum angle α, refine edge quota S, navigation sample number n, kNNG neighbor number K.
Output: PSP index G

$G \leftarrow NSSG\_Build(\mathcal{D}, R, \alpha, L, K)$; ▷ refer to SSG paper [18]
for *each node* i *in* G **do**
  $G \leftarrow \mathrm{EF}(i, S, \alpha, G)$; ▷ refer to Alg. 2
$\mathrm{N} \leftarrow \mathrm{SN}(\mathcal{D}, n)$ ▷ refer to Alg. 3
$G \leftarrow \{G, N\}$
return G

According to Theorem 4, reducing s drives the convergence of $\langle r, q\rangle$ towards $\langle p, q\rangle$, while also decreasing the estimated number of nodes ($nQ(s)$) that maximize the IP values relative to q. Since s represents the upper bound of the Euclidean distance between r and q, a judicious choice of s can effectively identify about $nQ(s)$ points in $\mathcal{D}$ as both the top-k MIPS solutions for q and the range-s nearest neighbors of p. In most Euclidean proximity graphs, it is likely that neighbors of neighbors are also neighbors [52]. Thus, after locating the top-1 MIPS solution r_0 for query q, Algorithm 1 may require extra iterations to navigate among r_0's nearest neighbors for the remaining $k-1$ solutions. Due to the efficient pruning in sparse graphs like SSG, the nearest neighbors of r_0 are expected to be no more than $O(f(n)n^{1/d}\log n)$ hops away [19], where $f(n)$ is a slowly increasing function of n. Thus, the search complexity for top-k MIPS is bounded by $O\left(c_0\frac{\log n + c_1 d}{\log R + c_2 d} + c_3 f(n) k R n^{1/d}\log n\right)$, where c_3 is a scaling constant that integrates the terms linearly.

Summary. Previous transformation-based methods [37, 47, 57, 58, 74, 81, 84] sought to reduce the MIPS problem to the NNS problem using non-linear transformations. However, such transformations often impose strong theoretical assumptions [84], introduce potential structural distortions [46, 81], and complicate data updates (§2). Notably, no transformation-based graph methods provide theoretical guarantees on the reachability of the 1-MIPS solution. Our Theorems 1 and 2 demonstrate that this reachability can be achieved without space transformations. Furthermore, Theorems 3 and 4 validate the effectiveness of solving MIPS on an SSG, highlighting the crucial role of edge pruning in enhancing search performance

Algorithm 3: Edge Refinement (EF)

Input: Node n, new edge quota S, angle α, NSSG G.
Output: A refined graph G'

$E_n \leftarrow \emptyset$; $C \leftarrow$ 2-Hop neighbors of n on G; $G' \leftarrow G$
sort C in descending order of $\langle n, c_i\rangle$, $c_i \in C$;
E_n.add($(n, C[0])$); C.remove($C[0]$); $o \leftarrow$ zero vector;
while *C is not empty* **do**
  $p \leftarrow C[0]$; C.remove($C[0]$);
  if $cos\angle nop > \alpha$ **then**
    continue;
  E_n.add((n, p));
  if $E_n.size() \geq S$ **then**
    break;
for *each edge* e *in* E_n **do**
  G'.add(e) ▷ deduplicate if repeated
return G'

— a factor which is previously overlooked[39, 46]. Our extensive experiments in §5.2 substantiate this analysis.

4 PRACTICAL SOLUTION

In this section, we introduce a novel MIPS[2] framework tailored for practical applications. This framework comprises two key components: a new graph-based indexing structure named **Proximity Graph with Spherical Pathways (PSP)** and a novel search algorithm with **Adaptive Early Termination (AET)**. The PSP index is specifically designed to address the **Biased Solution** problem prevalent in MIPS approaches. Concurrently, the AET mechanism effectively mitigates the **Excessive Exploration** problem, enhancing search efficiency across diverse query distributions.

4.1 Proximity Graph with Spherical Pathway

Base Structure. Advanced theoretical models for nearest neighbor search often exhibit $O(N^2\log N)$ indexing time complexity [18, 19], which impedes practical application of our theoretical insights. To mitigate this, we employ an approximate Euclidean proximity graph, NSSG, a practical adaptation of its theoretical counterpart SSG [18]. This choice is driven by its efficiency and our theoretical analysis in §3. Please refer to SSG paper for details in NSSG indexing algorithm.

Proximity Graph with Spherical Pathways (PSP) addresses the biased solution problem by injecting new edges into the SSG, named as Spherical Pathways (SP), and introducing a lightweight extra structure for search entry management, named as Spherical Navigation (SN). These two components are introduced as follows.

Solving the MIPS problem often results in biased solutions towards points in regions of large norms, as highlighted in [39]. Traditional proximity graphs under the Euclidean metric typically do not inherently address this bias, as their edge selection criteria focus on local neighborhood connectivity, leading to limited sensory regions [68]. Empirical observations (Figure 2) reveal that MIPS routing typically involves *initially hopping towards a few answers from inner to outer rings, followed by a broader exploration process*

[2]For simplicity, we use MIPS instead of ϵMIPS and k-ϵMIPS for the rest of this paper.

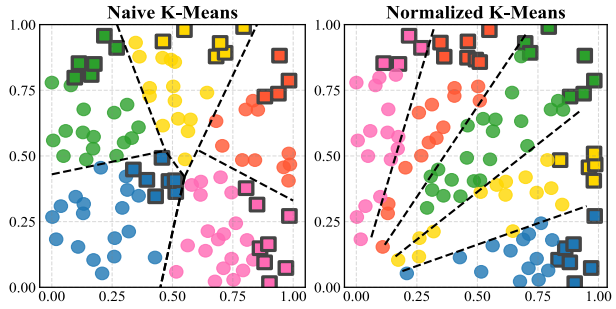

Figure 3: An illustration of k-means-based navigation node selection w/ and w/o normalization. Squared with dark edges are randomly selected navigation nodes with large norms.

Algorithm 4: SPHERICAL NAVIGATION (SN)

Input: Dataset $\mathcal{D}$, navigation sample number m.
Output: Inverted File N

$N \leftarrow \emptyset$;$\mathcal{D}' \leftarrow$ Sample($\mathcal{D}$) ▷ cluster on subset $\mathcal{D}'$ for big $\mathcal{D}$
$C \leftarrow$ Normalized k-means($\mathcal{D}, \mathcal{D}', c$); ▷ c is cluster number
$m_c \leftarrow m/c$;
for $i \in range(0, c)$ **do**
  $S_m \leftarrow$ sampling m_c nodes from $C[i]$ with Gaussian;
  $N[i]$.add(S_m);
return N

for the remaining answers. To enhance this process, we introduce **Spherical Pathways**, which benefit the MIPS routing in two aspects (Figure 2): **(1)** Linking each node to its MIPS neighbors in the outer ring to accelerate the inner-out routing; **(2)** Strengthening mutual connectivity within the "outer ring", thereby expanding sensory regions crucial for the spanning process in k-ϵMIPS.

Identifying MIPS neighbors for each node on a large scale is challenging; we address this with a heuristic that involves collecting k-hop neighbors and selecting up to S nodes with the largest IP distances as new neighbors. To widen the sensory region, we introduce a smallest-angle constraint inspired by the techniques used in NSSG indexing [18]. Specifically, any new MIPS neighbor m of node n (excluding the one with the largest IP distance), the angle $\angle nom \geq \alpha$. Empirically, we find that collecting 2-hop neighbors has significantly enhanced search performance by 8% (§5.2). This process, termed **Edge Refinement (EF)**, is detailed in Algorithm 3.

Spherical Navigation (SN). Navigation nodes (entry nodes of the search algorithm) are pivotal in optimizing search performance on proximity graphs [17, 44]. Given the bias in MIPS solutions towards large-norm vectors, selecting navigation nodes among them can be advantageous. To prevent over-concentration of navigation nodes, we cluster the base vectors' normalised projections on a unit sphere using k-means, prioritizing angular separation over Euclidean distance (see Figure 3). We then sample m/c navigation nodes from each of the c clusters based on their norm distribution in original space. While the norm distribution follows a long-tail pattern, the Gaussian distribution can serve as an effective approximation in practice. The resulting structure maintaining the selected navigation nodes is encapsulated as an Inverted File (IVF) [31]. Clusters are keyed by their centers, while navigation nodes are stored in corresponding inverted lists. The SN selection process is outlined in Algorithm 4. Notably, while SN is built on normalised vectors, it only generates the entry points' IDs. IP metric is used in the original space when searching on the graph, aligning with our theory.

To summarize, building a PSP (detailed in Algorithm 2) involves three steps: (1) Build an NSSG; (2) Apply EF; (3) Apply SN.

Algorithm 5: SEARCH ON PSP

Input: Dataset $\mathcal{D}$, PSP G, query q, adaptive early termination function $ET(\cdot)$, candidate set size l_s, result set size k, query scale factor μ.
Output: Top k result set R.

$R \leftarrow \emptyset$; $Q \leftarrow \emptyset$;
$C \leftarrow$ closest navigation cluster look-up;
$P \leftarrow$ random sample l_s nodes from C;
for *each node p in P* **do**
  Q.add(p,$\langle p, q\rangle$);
Q.make_max_heap(); R.init_min_heap() ▷ compare IP value
while $Q.size()$ **do**
  $p \leftarrow Q.pop()[0]$; $R.insert((p, \langle p, q\rangle))$
  $f_p \leftarrow$ get_features(p); ▷ calculate features for $ET(\cdot)$
  if $ET(f_p)$ **then**
    break; ▷ $ET(\cdot)$ is true for stop
  if *visited(p)* **then**
    continue;
  $N_p \leftarrow$ neighbors of p in G
  $Q \leftarrow$ batch_insert(Q, N_p)
  $Q.resize(l_s)$;$R.resize(k)$ ▷ delete small-IP value nodes
return R

4.2 Search On PSP

Standard GNNS [52] does not fully release the potential of PSP. To optimize its efficiency for MIPS and ensure effective alignment with PSP, we employ three key adaptations: **(1)** selecting navigation nodes based on SN; **(2)** altering the metric as inner product; and **(3)** integrating adaptive early termination (Algorithm 5).

Entry Points Selection. Unlike the random initialization used in GNNS, we generates navigation nodes from SN. For a query q, we first identify the closest cluster in the SN-IVF with cosine similarity, then randomly fills the initial pool from the corresponding inverted list. This is efficient because of the limited number of clusters, ensuring a swift and precise start to the search.

Altering the Metric. Leveraging Theorem 1 and 2, IP metric is enabled for GNNS on the PSP. Meanwhile, the max-heap and min-heap are interchanged adaptively due to the pursuing of answers with larger IP distances instead of smaller L_2 distances.

Adaptive Early Termination. Early termination is crucial to search efficiency [35, 40, 79]. The exploration process is highly sensitive to the query distribution, yet standard GNNS is unaware of different query distributions and the accuracy of the result set. As shown in Figure 4(a), the distribution of the number of nodes visited exhibits a long tail, with most queries reaching 99% recall with a small exploration depth. Instead of a uniform configuration

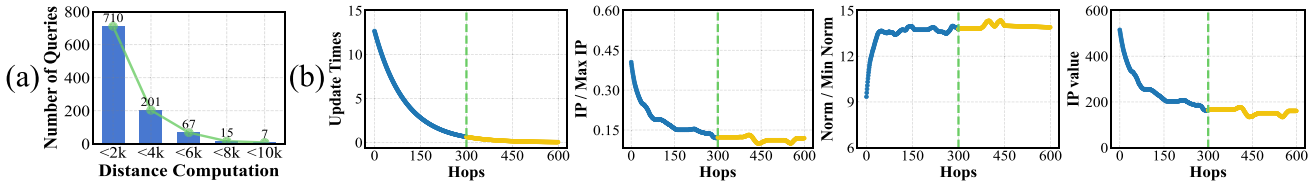

Figure 4: An illustration of early termination intuitions. Figure (a) shows the quantity distribution of visited nodes to reach 99% recall@100 on Text2Image1M. Figure (b) presents the changing trends of four selected features during search for a query. The green dotted lines indicate the optimal stop state. After that (yellow curves), the recall@100 stops increasing.

Table 2: Features used in adaptive early termination. EMA means exponential moving average.

Feature	Description
F1: IP value	EMA of (IP distance to q)
F2: Norm/Min Norm	EMA of (norm / historical min norm)
F3: IP/Max IP	EMA of (ip / historical max ip)
F4: Update Times	Top k update frequency (using EMA)

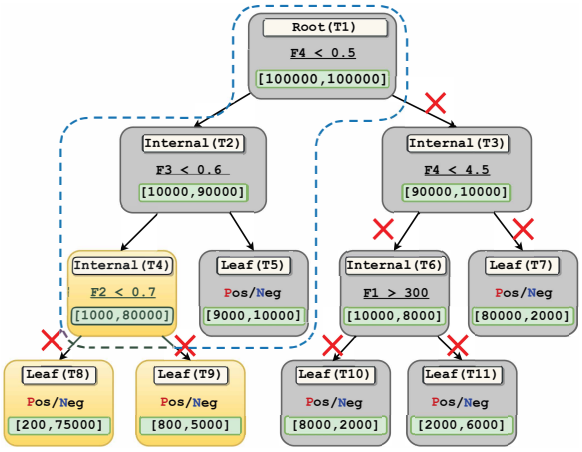

Figure 5: A showcase of a trained decision tree. The crosses denote pruned branches. The yellow nodes predict "stop". The conditional expressions are learnt decision logic based on selected features. "Pos/Neg" denotes the number of positive/negative samples fall in leaves, indicating stop tendency.

for search parameters, adaptively stopping the search earlier for simpler queries can significantly enhance the amortized processing time and reduce excessive explorations. To equip IP-GNNS with "self-awareness", our method integrates a decision function to monitor key features indicative of the optimal stopping point, namely Adaptive Early Termination (AET), incorporates the following:

Feature Selection. Features for the decision model are chosen based on the following principles. **(1) Relevance**: Features must be closely related to the IP metric, the query, and the tendency toward termination. **(2) Distinguishability**: Features must clearly distinguish between states before and after the termination point. **(3) Simplicity**: The computational cost of the features should be minimized to maintain efficiency. Theorem 4 indicates the **vector norms**, the **IP values**, and **exploring neighborhood** of top-1 MIPS solution are key aspects in feature engineering. Notably, the update frequency of the top-k candidate pool serves as a good indicator of the neighborhood exploration. After extensive and rigorous experiments (Figure 4(b)), we finalized the feature set, detailed in Table 2.

Decision Model. Utilizing the identified four highly discriminative features, a simple decision tree (DT) is employed, trained on labeled data points collected via searching on PSP. The data is generated through the following steps: **(1)** Divide the base vectors into new base and new query sets; **(2)** Perform searches on the new queries using the base indexed by PSP. For each query, record the node ID (boundary node) when the recall of the result set ceases to increase; **(3)** Randomly sample nodes before the boundary node (labeled as 1) and after the boundary node (labeled as 0) to form the training data. **(4)** Train the DT on the collected training data.

Function Generation. To facilitate efficient implementation of the AET decision function, it is essential to convert the DT into C++ code. We propose a structured approach to simplify the DT while maintaining its effectiveness and efficiency: **(1)** Limit the height of the DT to the number of features to prevent over-fitting. **(2)** Adjust the prediction mechanism of the leaf nodes (Figure 5). A leaf node predicts "stop" only when the ratio $\frac{\#Negative\ Samples}{\#Positive\ Samples} > \theta$. This approach prioritizes higher confidence in stopping the search, reducing the risk of prematurely terminating. Users can adjust θ to control the aggressiveness of the AET function. **(3)** Remove subtrees whose leaf nodes share the same predictions to simplify the tree. **(4)** Translate the resulting DT into 'if-else' clauses in C++ from the tree structure. Figure 5 is an example of this process. The final interpreted clause is "if (F4<0.5 & F3 < 0.6) stop;else continue;"

4.3 Complexity Analysis

The Indexing algorithm consists of two main stages: **(1)** NSSG indexing and **(2)** the EF and SN. Let n denote the dataset size, r the max degree of the graph, d the dimension, c the number of clusters in SN, and m the number of navigation nodes in SN. The time complexity of NSSG indexing is dominated by the IVFPQ [31] based kNN graph initialization [18], which has an empirical complexity of $O(dn \log n)$. For each node, the EF phase involves IP calculation ($O(ndr^2)$), sorting ($O(nr^2 logr)$) and new neighbor selection ($O(nr^2)$), cumulatively yielding $O(nr^2(d + logr))$. The SN phase, dominated by the k-means cluster assignment, scales as $O(ncd)$. Absorbing smaller constants into c_4, the overall indexing complexity of PSP is $O(c_4 n \log n + n)$, dominated by $O(n \log n)$ term, aligned with the empirical results presented in §5.2.

The Search complexity for generally favoured top-k MIPS is approximated by $O(c_0 \frac{\log n + c_1 d}{\log R + c_2 d} + c_3 f(n) k R n^{1/d} \log n)$ in §3. It is mainly

Table 3: Dataset statistics. Dim indicates vector dimension.

Dataset	Base Size	Dim	Query Size	Modality
Netflix [14]	17,770	300	1,000	Video
MNIST [1]	60,000	784	10,000	Image
DBpedia100K [48]	100,000	3072	1,000	Text
DBpedia1M [48]	1,000,000	1536	1,000	Text
Music100 [26, 46]	1,000,000	100	10,000	Audio
Text2Image1M [2]	1,000,000	200	100,000	Multi
Text2Image10M [2]	10,000,000	200	100,000	Multi
Commerce100M	100,279,529	48	64,111	E-commerce

dominated by the $O(kRn^{1/d}\log n)$ term. Empirical evaluations (detailed in §5) aligns with our estimation. Notably, our method achieves 95% recall with an average of only 3ms per query on a dataset of 100 million vectors. This efficiency implies a small constant factor in the complexity formula and underscores the practical scalability of our approach for large-scale applications.

Worst Cases. The worst case will invalidate any acceleration techniques for approximate methods, e.g., points form a straight in the space. In such cases, the search complexity growing with n on PSP will downgrade to near $O(n)$. Similarly, the indexing complexity of PSP will downgrade to near $O(n^2)$.

5 EXPERIMENTAL EVALUATION

We conduct extensive experiments on real-world datasets to validate the theoretical contributions of our proposed framework, as well as its efficiency, scalability, and practical applicability for the MIPS task. Our analyses are guided by the following questions:

- **Q1**: How does the proposed theory align MIPS with NNS?
- **Q2**: How does PSP perform compared to established baselines across various modalities, dimensionalities, and cardinalities?
- **Q3**: What benefits do the individual optimization techniques bring to the search efficiency of PSP?
- **Q4**: How is the viability of PSP in large-scale applications?

5.1 Experimental Setup

Datasets. All empirical evaluations were conducted on eight real-world datasets, covering diverse modalities, cardinalities and dimensionalities (Table 3): **Netflix** [14], **MNIST** [1] and **Music100** [26, 46] datasets are the established benchmarks for the MIPS problem. **DBpedia** [48] is extracted by OpenAI text-embedding-3-large model. **Text2Image** [2] is a cross-modality dataset where image embeddings (base) are extracted from Se-ResNext-101 and text embeddings (query) from a DSSM Model. **Commerce100M**[3], generated from real traffic logs of a large-scale e-commerce platform. The embeddings are inferred from an advanced pre-trained personalized deep retrieval model. The dimension is set to 48 due to the resource and application demands in industrial scenarios, while 16 to 64 are common choices in this field [69, 77], discussed in [4].

Competitors were selected from recent state-of-the-art research, encompassing **(1) ip-NSW** [46]: A graph based method using inner-product navigable small world graph. **(2) ip-NSW+** [39]: An enhancement of ip-NSW that introduces an additional angular proximity graph. **(3) Möbius-Graph** [84]: A graph based method that reduces the MIPS problem to an NNS problem using Möbius transformation. Since the original code is not available, we implemented a version based on the paper. **(4) NAPG** [62]: the state-of-the-art graph based method. We focus mainly on in-memory methods and exclude those for hardware-oriented optimisation. **(5) Fargo** [81]: The latest state-of-the-art LSH based method with theoretical guarantees. **(6) ScaNN** [25]: The latest state-of-the-art quantization method, an optimised version based on SOAR [61];

[3]Data source withheld for anonymity during review; will be released upon publication

Table 4: The average overlap ratio of NNS-MIPS search paths and recall@100 with varying μ, on MNIST.

Metric	$\mu = 0.1$	$\mu = 1$	$\mu = 10$	$\mu = 100$	$\mu = 1200$
Overlap ratio	7.4%	15.3%	72.4%	99.5%	100%
Recall@100	0.0	0.08	0.83	1.0	1.0

Implementation. All baselines except for *ScaNN* are written in C++. The *ScaNN* library is called by Python bindings yet written in C. The experiments are executed on a CentOS machine with 128G RAM on Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU. We use OpenMP for parallel index construction, utilizing 48 threads for all methods. For query execution, we turn off additional optimizations to ensure fair comparison. Our codes are available at [3].

Parameter Settings. PSP required tuning several hyper-parameters: number of neighbors in kNN graph (K), NSSG candidate size (L), maximum out-degree (R), minimal angle between edges (α), and the number of nodes added in EF (S). By default, we set $K = 400$, $L = 800$, $R = 40$, $\alpha = 60^\circ$, and $S = 5$ for all datasets. For a fair comparison, we also employ grid search to optimize the parameters of baseline methods over all datasets and select the best empirical configuration for each baseline.

Evaluation Metrics. For search performance comparison, we measure effectiveness and efficiency using two popular metrics: **(1) Recall vs. Queries Per Second (QPS)**, denoting the number of queries that an algorithm can process per second at each $recall@100$ level; and **(2) Recall vs. Computations**, indicating the number of distance computations performed by the search algorithm. Let R'_t denote the set of k vectors returned by the algorithm, and R_t represent the ground truth, the recall@k is formally defined as $recall@k = \frac{|R_t \cap R'_t|}{|R_t|} = \frac{|R_t \cap R'_t|}{k}$ Additionally, we consider construction time and index size to evaluate the scalability and application viability of the algorithms.

5.2 Experimental Results

Exp-1: Theory Verification (Q1). To explore the MIPS-NNS equivalence with respect to μ, we conduct GNNS on an ideal PSP index, obtained by setting L to the size of the base vectors and R to infinity in Algorithm 2. This experiment is conducted on MNIST, and we also evaluate Theorem 4 on synthetic datasets sampled from a standard normal distribution, examining the relationship between top-k MIPS solutions and the top-1 solution's Euclidean neighbors.

Duality of NNS and MIPS on Proximity Graph. We assessed the average overlap ratio of search paths for NNS and MIPS, and recall@100 for MIPS, across all queries on MNIST under different μ (Table 4). The result indicates that while precisely defining the lower

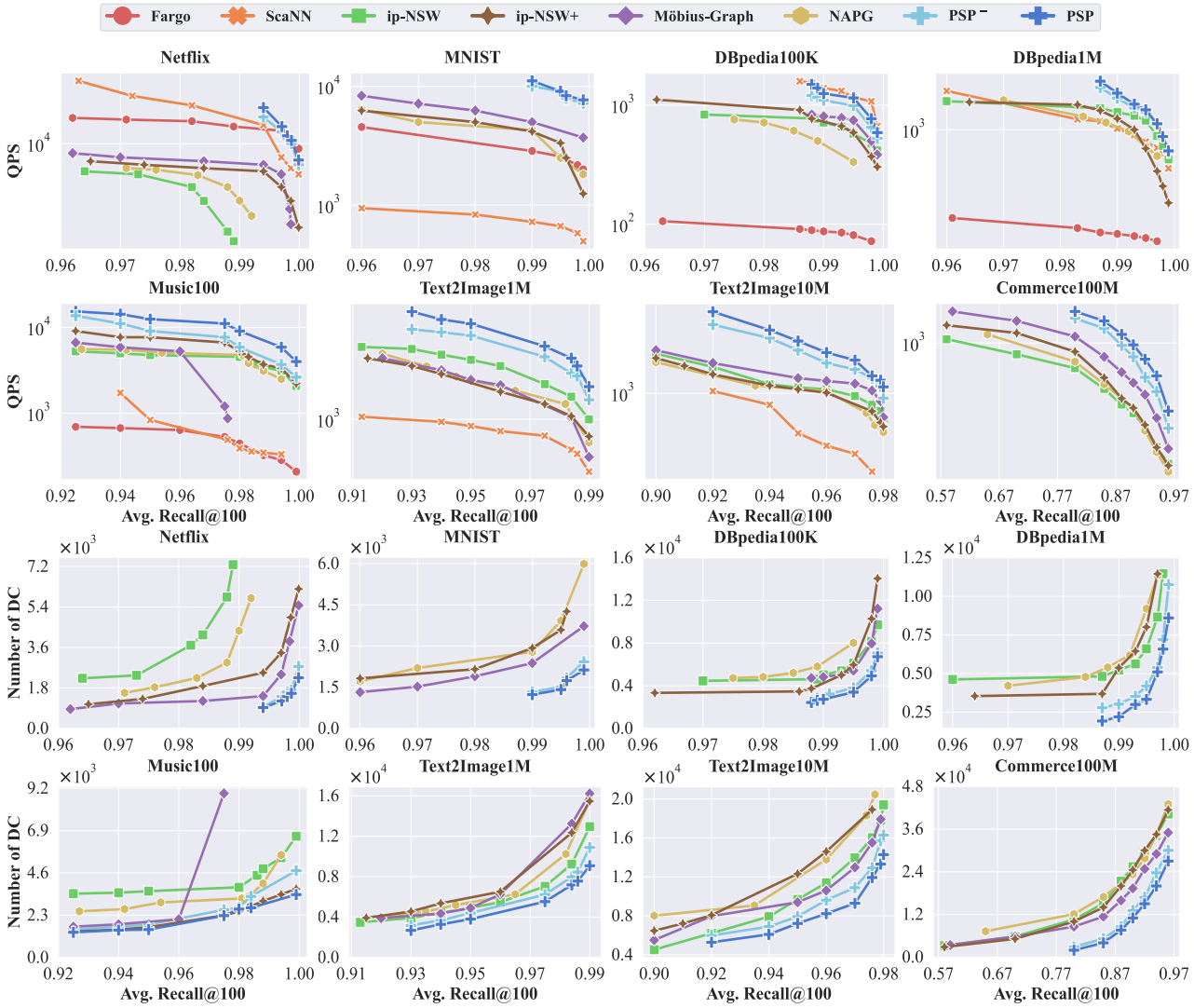

Figure 6: Experimental results of query process performance on eight datasets. DC denotes Distance Computation.

bound of μ for every possible query is impractical, a sufficiently large μ can effectively unify the MIPS and NNS paths for all queries. Even sub-optimal μ yields reasonably good candidates due to highly overlapping paths. Case study in [4] visualizes this. Additionally, we evaluate the overlap ratio between the top-k MIPS solutions and the neighborhood of the top-1 MIPS solution. The results in [4] confirm that top-k MIPS solutions are likely to be distributed in the Euclidean neighborhood of the top-1 MIPS solution.

Exp-2: Performance Assessments (Q2). The proposed method is thoroughly assessed over all datasets on both query processing cost and indexing overhead, as presented in Figure 6 and 7. The cases, where a method is *absent in the figure*, indicates either excessively high processing time or low recall (check [4] for full results).

Query Processing Performance. Figure 6 presents the query processing performance for different methods. The top rows depict queries per second (QPS) at varying recall levels, where performance in the upper right indicates superior speed and accuracy. The bottom rows show the number of distance computations required, with the lower right being more desirable for efficiency. Log scales are applied for clarity across wide value ranges.

PSP consistently outperforms all baselines across all datasets, achieving up to 4× and 40% speedup over Möbius-Graph (2^{nd} best) on MNIST and Commerce100M, respectively. This demonstrates PSP's superior efficiency with high-dimensional and large-scale data, effectively leveraging its strong theoretical foundations and ensuring robust connectivity. In contrast, other graph-based methods suffer from connectivity issues, and non-graph methods' inefficiency mainly owes to inefficient indexing large-norm points.

PSP⁻, representing an unoptimized NSSG, still surpasses other baselines across datasets by an average of 20%-25%, except on Music100. This aligns with our analysis that MIPS can be executed efficiently without transformation. Further optimizations, like EF, SN, and AET, improve performance by about 10%-15%.

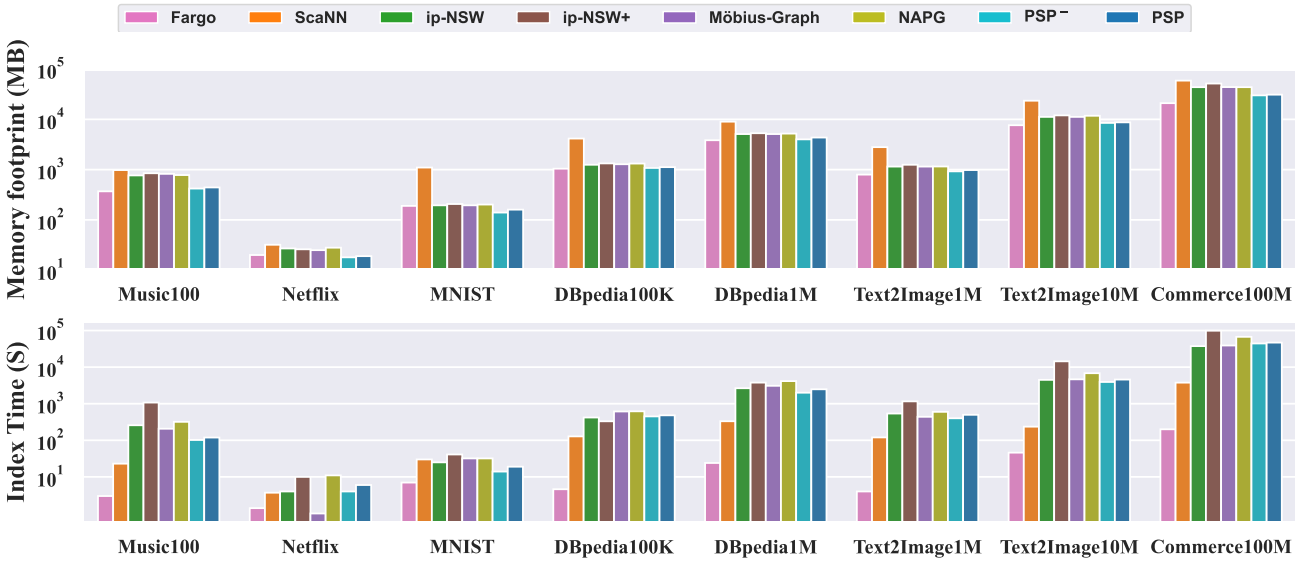

Figure 7: Experimental results on indexing time and memory footprint of different methods across eight datasets.

Table 5: The querying and indexing performance over various data scales on Text2image10M at 99% recall.

Dataset Scale	Query Time	Indexing Time	Index Size
Text2image100K	0.13 (ms)	62 (s)	15 (MB)
Text2image1M	0.46 (ms)	498 (s)	148 (MB)
Text2image10M	0.95 (ms)	5672 (s)	1520 (MB)

Others: (1) Fargo marginally outperforms PSP at high recall on Netflix but underperforms on others, notably achieving only 90% recall on Commerce100M. Its reliance on LSH and susceptibility to transformation-induced distortion limits its general performance. **(2) ScaNN** excels on DBpedia100K but struggles elsewhere, highlighting the sensitivity of quantization methods. For datasets not well-suited to quantization, such as large-scale or IID distributions, performance declines [38], as seen that its recall stuck at only 54% on Commerce100M. **(3) ip-NSW and ip-NSW+** lag behind PSP due to high graph density and precision limitations. Although ip-NSW+ partially addresses precision issues, its performance can be inconsistent, especially on datasets like Text2Image1M, where the additional heuristics-angular graphs-negatively impact performance. **(4) Möbius-Graph** shows strong performance on Commerce100M but encounters precision bottlenecks on Music100 and DBpedia1M due to reliance on transformation-induced assumptions that are often violated in practice (§2). **(5) NAPG** performs similarly to or worse than ip-NSW due to inadequate graph sparsification and use of heuristics-based scaled IP metrics lacking theoretical support.

Indexing Performance. Figure 7 shows the index memory footprint and indexing time for various methods. PSP achieves a balanced memory footprint and indexing time across all datasets. On Commerce100M, PSP requires only 12 GB and 13 hours for indexing, making it feasible for daily updates—essential for adapting to evolving business metrics. In contrast, ip-NSW+ takes over a day for indexing. While Fargo is the fastest and least memory-intensive, it suffers from poor query performance. Indices of ScaNN, ip-NSW, ip-NSW+, Möbius-Graph, and NAPG consume 1.5 to 2× the memory of PSP, reducing their practicality. Graph-based methods have an index size about 3× larger than PSP, excluding data, due to high redundancy in their edges. ScaNN also requires extensive memory for large tree structures to improve search precision.

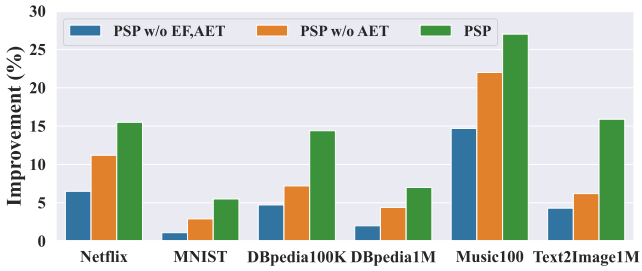

Figure 8: Ablation study results across six different scale datasets. We measure the benefits of three optimization techniques on query execution.

Exp-2 Summary. PSP's superiority is attributed to : strong graph connectivity, no transform in space, efficient edge pruning, and tailored adaptations for MIPS. Superior search performance and efficient indexing makes PSP well-suited for large-scale applications.

Exp-3: Ablation Study (Q3). We validates the impact of key components of PSP across six datasets, excluding Text2Image10M and Commerce100M. We examine the effects of proposed optimisation for MIPS: SN, EF, and AET. Query execution time and distance computations for four PSP variations are recorded at recall@100 level of 99% (Figure 8). Key findings include:

Effect of Spherical Navigation. SN improves average performance by 6.5% across six datasets by starting searches in high-norm areas, reducing redundant computations. On Music100, it provides a 15% speedup, suggesting a highly biased distribution of MIPS solutions.

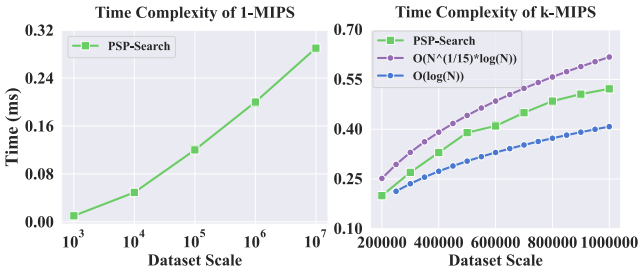

Figure 9: Query time versus data scale on Text2image at 99% recall for top-1 and top-k MIPS. For top-1 queries, both the x and y axes use a logarithmic scale, resulting in an almost straight line from bottom-left to top-right, indicating a growth rate close to $O(\log n)$.

Effect of Edge Refinement. EF adds a 3.8% average improvement, and 8% on Music100, by enhancing connectivity and increasing graph degree slightly, especially towards high-norm regions. The "Spherical Highways" introduced by EF serve as shortcuts, improving navigation efficiency. The effects of EF and SN may overlap, reducing additional benefits when combined.

Effect of Adaptive Early Termination. AET yields an average improvement of 4.9%, mitigating redundant paths once the optimal set is found. It is particularly effective for queries following long-tailed distributions, with improvements of 10.6% on Text2Image1M and 8.3% on DBpedia100K. Further speedups are achievable via smaller θ setting (4.2) with minor acceptable recall loss.

Exp-4: Application Viability (Q4) is determined by several aspects: the ease of hyper-parameter tuning, the scalability with dataset cardinality, the scalability with demanded answer number (k), and the ease of tuning recall@k level. We find: **(1)** Extensive tests on Text2Image confirm the theoretical complexity estimates from §3. Results (Figure 9) show $O(\log n)$ scalability for top-1 MIPS and near $O(\log n)$ scalability for top-k MIPS, with manageable indexing times and approximately linear index size growth (Table 5). PSP achieves 99% recall with a query time of just 0.9 ms on a 10 million scale dataset, demonstrating strong scalability with a small constant factor. **(2)** The performance of PSP is mainly determined by R and quality of the base NSSG. Generally, higher quality of NSSG leads to better performance. Optimal R exists yet remains consistent across all scales, allowing for efficient tuning on smaller subsets. **(3)** The query process time grows near sub-linearly with k, allowing for efficient search with large return quantity.
More results and detailed discussions can be found in [4].

6 RELATED WORK

Inner Product (IP) is ubiquitous in representation learning, classification, clustering, knowledge graphs, recommender system, and information retrieval [6, 20, 22, 27, 67, 73, 76]. The increasing use of high-dimensional vector-based data has emphasized the need for efficient Maximum Inner Product Search [24, 49]. The rapid advancement of large language models [80] and their applications[6, 11], has further fueled interest in vector-based databases [24, 66]. Prevalent MIPS methods can be categorized by their indexing strategies into LSH, tree, quantization, and graph based approaches:

LSH-based methods. Traditional LSH isn't directly suitable for the IP metric, prompting adaptations via increasing vector dimensionality, such as L2 [57], Correlation [58], and the popular XBOX [9], which introduces certain data distortion. Among the works based on XBOX [28, 32, 47, 50, 74, 81], Fargo [81] achieves state-of-the-art.

Tree-based methods. Early MIPS research favours tree-based algorithms [33, 54], which struggled with the challenges of high dimensionality. ProMIPS [60], tackles the dimensionality issue by projecting the original vectors into a lower-dimensional space and utilizing iDistance [30] for indexing. However, it suffers from information loss with the projection. The recently developed LRUS-CoverTree [42] was designed to mitigate these inefficiencies. Despite the improvements, the LRUS-CoverTree often struggles with scenarios involving negative inner product values.

Quantization-based methods. These target to efficiently estimate the distances between the data vectors and query vectors during the query phase to shortlist a set of candidates. ScanNN [25] combine the "VQ-PQ" framework with a novel anisotropic quantization loss, presenting a cutting-edge quantization-based solution. Further, SOAR [61] uses an orthogonality-amplified residual loss to learn more efficient quantization representations and have become state-of-the-art and been integrated into the ScaNN library.

Graph-based methods. Proven effective for NNS [8, 16, 18, 19, 41, 44], graph-based techniques have been adapted for MIPS. ip-NSW [46] replaces Euclidean metric in NSW [43] by IP. ip-NSW+ [39] further improves it by adding an angular proximity graph to reduce traversal on small-norm points. Möbius-Graph [84] establishes isomorphism between the Delaunay graph for IP and that for Euclidean metric through Möbius transformation, albeit with assumptions. Other graph-based algorithm [62, 63] introduce heuristic edge selection strategies to accelerate the search, such as norm range criteria, to build an enhanced graph index for MIPS. From the perspective of this paper, we can also organise these works into vector-space-transformation ones [84] and non-transformation ones [39, 46, 62].

Other Acceleration Techniques. [37] utilizes machine learning to optimize the LSH indices. EI-LSH [40] enhances LSH-based methods with early termination techniques. Similar methodologies have also been explored in graph-based methods [35] for NNS.

7 CONCLUSION

In this paper, we align Maximum Inner Product Search (MIPS) with Nearest Neighbor Search (NNS) through robust theoretical foundations, presenting the first analytical framework for MIPS efficiency on graph indices. Addressing the inherent challenges of biased solutions and excessive explorations in graph-based MIPS, we introduce a practical framework, featuring a novel graph index, PSP, and a novel lightweight adaptive early termination mechanism, AET. Extensive experiments confirm the theoretical validity, efficiency, scalability, and application viability of our approaches. Notably, our method has been deployed in Shopee Search due to its outstanding performance. We also contribute the Commerce100M dataset to the public community to fill the absence of e-commercial data. Our code will be released with this paper.

REFERENCES

[1] 1998. MNIST. http://yann.lecun.com/exdb/mnist/.

[2] 2021. Text-to-Image. https://research.yandex.com/blog/benchmarks-for-billion-scale-similarity-search.

[3] 2024. Code. *https://github.com/TiyCHEN/PSP*.

[4] 2024. Full version. *https://github.com/TiyCHEN/PSP*.

[5] Firas Abuzaid, Geet Sethi, Peter Bailis, and Matei Zaharia. 2019. To index or not to index: Optimizing exact maximum inner product search. In *ICDE*. 1250–1261.

[6] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *ACL*. 41–46.

[7] Franz Aurenhammer. 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *CSUR* 23, 3 (1991), 345–405.

[8] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2023. Elpis: Graph-based similarity search for scalable data science. *PVLDB* 16, 6 (2023), 1548–1559.

[9] 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.

[10] Frank Bowman. 1958. *Introduction to Bessel functions*.

[11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*. 1877–1901.

[12] Sebastian Bruch, Franco Maria Nardini, Amir Ingber, and Edo Liberty. 2023. An approximate algorithm for maximum inner product search over streaming sparse vectors. In *TOIS*. 1–43.

[13] Lijie Chen. 2020. On The Hardness of approximate and exact (bichromatic) maximum inner product. In *Theory OF Computing*. 1–50.

[14] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2015. A learning-rate schedule for stochastic gradient methods to matrix factorization. In *PAKDD*. 442–455.

[15] John H Curtiss. 1942. A note on the theory of moment generating functions. *The Annals of Mathematical Statistics* 13, 4 (1942), 430–433.

[16] Chao Feng, Defu Lian, Xiting Wang, Zheng Liu, Xing Xie, and Enhong Chen. 2023. Reinforcement routing on proximity graph for efficient recommendation. *TOIS* 41, 1 (2023), 1–27.

[17] Cong Fu and Deng Cai. 2016. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. *arXiv preprint arXiv:1609.07228* (2016).

[18] Cong Fu, Changxu Wang, and Deng Cai. 2021. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. *TPAMI* 44, 8 (2021), 4139–4150.

[19] 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.

[20] Yasuhiro Fujiwara, Yasutoshi Ida, Atsutoshi Kumagai, Masahiro Nakano, Akisato Kimura, and Naonori Ueda. 2023. Efficient Network representation learning via cluster similarity. *DSE* 8, 3 (2023), 279–291.

[21] Benyamin Ghojogh, Saeed Sharifian, and Hoda Mohammadzade. 2018. Tree-based optimization: A meta-algorithm for metaheuristic optimization. *arXiv preprint arXiv:1809.09284* (2018).

[22] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *SIGKDD*. 311–320.

[23] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *AISTATS*. 482–490.

[24] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, et al. 2022. Manu: a cloud native vector database management system. *PVLDB* 15, 12 (2022), 3548–3561.

[25] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *ICML*. 3887–3896.

[26] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[27] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *SIGKDD*. 2553–2561.

[28] Qiang Huang, Guihong Ma, Jianlin Feng, Qiong Fang, and Anthony KH Tung. 2018. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1561–1570.

[29] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*. 604–613.

[30] Hosagrahar V Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. 2005. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. *TODS* 30, 2 (2005), 364–397.

[31] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *TPAMI* 33, 1 (2010), 117–128.

[32] Omid Keivani, Kaushik Sinha, and Parikshit Ram. 2018. Improved maximum inner product search with better theoretical guarantee using randomized partition trees. *Machine Learning* 107, 6 (2018), 1069–1094.

[33] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. 2012. Efficient retrieval of recommendations in a matrix factorization framework. In *CIKM*. 535–544.

[34] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*. 9459–9474.

[35] Conglong Li, Minjia Zhang, David G Andersen, and Yuxiong He. 2020. Improving approximate nearest neighbor search through learned adaptive early termination. In *SIGMOD*. 2539–2554.

[36] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD*. 835–850.

[37] Jinfeng Li, Xiao Yan, Jian Zhang, An Xu, James Cheng, Jie Liu, Kelvin KW Ng, and Ti-chung Cheng. 2018. A general and efficient querying method for learning to hash. In *SIGMOD*. 1333–1347.

[38] 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. *TKDE* 32, 8 (2019), 1475–1488.

[39] 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.

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

[41] Kejing Lu, Mineichi Kudo, Chuan Xiao, and Yoshiharu Ishikawa. 2021. HVS: hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search. *PVLDB* 15, 2 (2021), 246–258.

[42] Hengzhao Ma, Jianzhong Li, and Yong Zhang. 2024. Reconsidering Tree based Methods for k-Maximum Inner-Product Search: The LRUS-CoverTree. In *ICDE*.

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

[44] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *TPAMI* 42, 4 (2018), 824–836.

[45] EJ McShane. 1937. Jensen's inequality. *AMS* 43, 8 (1937), 521–527.

[46] Stanislav Morozov and Artem Babenko. 2018. Non-metric similarity graphs for maximum inner product search. In *NeurIPS*. 4726–4735.

[47] Behnam Neyshabur and Nathan Srebro. 2015. On symmetric and asymmetric lshs for inner product search. In *ICML*. 1926–1934.

[48] Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV* 42, 3 (2001), 145–175.

[49] James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of vector database management systems. *arXiv preprint arXiv:2310.14021* (2023).

[50] Ninh Pham. 2021. Simple yet efficient algorithms for maximum inner product search via extreme order statistics. In *SIGKDD*. 1339–1347.

[51] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*. 1–8.

[52] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. 2020. Graph-based nearest neighbor search: From practice to theory. In *ICML*. 7803–7813.

[53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*. 8748–8763.

[54] Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *SIGKDD*. 931–939.

[55] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS*. 25278–25294.

[56] Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*. 4430–4441.

[57] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). *NIPS* (2014), 2321–2329.

[58] Anshumali Shrivastava and Ping Li. 2015. Improved asymmetric locality sensitive hashing (ALSH) for Maximum Inner Product Search (MIPS). In *UAI*. 812–821.

[59] Richard L Smith. 1990. Extreme value theory. *Handbook of applicable mathematics* 7, 437-471 (1990), 18.

[60] Yang Song, Yu Gu, Rui Zhang, and Ge Yu. 2021. ProMIPS: Efficient high-dimensional C-approximate maximum inner product search with a lightweight index. In *ICDE*. 1619–1630.

[61] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2024. SOAR: improved indexing for approximate nearest neighbor search. *NeurIPS* 36 (2024).

[62] Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, and Ping Li. 2021. Norm adjusted proximity graph for fast inner product retrieval. In

SIGKDD. 1552–1560.
[63] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. 2019. On efficient retrieval of top similarity vectors. In *EMNLP-IJCNLP*. 5236–5246.
[64] Christina Teflioudi and Rainer Gemulla. 2016. Exact and approximate maximum inner product search with lemp. *TODS* 42, 1 (2016), 1–49.
[65] Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. 2015. Lemp: Fast retrieval of large entries in a matrix product. In *SIGMOD*. 107–122.
[66] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A purpose-built vector data management system. In *SIGMOD*. 2614–2627.
[67] Mengzhao Wang, Xiangyu Ke, Xiaoliang Xu, Lu Chen, Yunjun Gao, Pinpin Huang, and Runkai Zhu. 2024. Must: An effective and scalable framework for multimodal search of target modality. In *ICDE*.
[68] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *PVLDB* 4, 2 (2021), 1964–1978.
[69] Tian Wang, Yuri M. Brovman, and Sriganesh Madhvanath. 2021. Personalized embedding-based e-Commerce recommendations at eBay. *arXiv preprint arXiv:2102.06156* (2021).
[70] Xueyi Wang. 2011. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *The 2011 international joint conference on neural networks*. 1293–1299.
[71] 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*. 194–205.
[72] Eric W Weisstein. 2002. Hypersphere. *https://mathworld. wolfram. com/* (2002).
[73] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Product knowledge graph embedding for e-commerce. In *WSDM*. 672–680.
[74] Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. 2018. Norm-ranging LSH for maximum inner product search. In *NeurIPS*. 2956–2965.
[75] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. Pase: Postgresql ultra-high-dimensional approximate nearest neighbor search extension. In *SIGMOD*. 2241–2253.
[76] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. 2014. Large-scale multi-label learning with missing labels. In *ICML*. 593–601.
[77] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wenyun Yang. 2020. Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning. In *SIGIR*. 2407–2416.
[78] Jin Zhang, Defu Lian, Haodi Zhang, Baoyun Wang, and Enhong Chen. 2023. Query-aware quantization for maximum inner product search. In *AAAI*. 4875–4883.
[79] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, et al. 2023. VBASE: Unifying online vector similarity search and relational queries via relaxed monotonicity. In *OSDI*. 377–395.
[80] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
[81] 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. *PVLDB* 16, 5 (2023), 1100–1112.
[82] Yuxin Zheng, Qi Guo, Anthony KH Tung, and Sai Wu. 2016. Lazylsh: Approximate nearest neighbor search for multiple distance functions with a single index. In *SIGMOD*. 2023–2037.
[83] Xiaoping Zhou, Xiangyu Han, Haoran Li, Jia Wang, and Xun Liang. 2022. Cross-domain image retrieval: methods and applications. *IJMIR* 11, 3 (2022), 199–218.
[84] Zhixin Zhou, Shulong Tan, Zhaozhuo Xu, and Ping Li. 2019. Möbius transformation for fast inner product search on graph. In *NeurIPS*. 8218–8229.

APPENDIX

A COMPLETE PROOF OF THEOREM 2

*Given a proximity graph $G = (V, E)$ and a query $q \in \mathbb{R}^d$, when using the standard Graph Nearest Neighbor Search (*GNNS*) [52] algorithm to decide the* MIPS *solution for q, there exists a scalar $\bar{\mu}$ such that for $\forall \mu > \max(\bar{\mu}, 0)$, we can identify a node $p^* \in \mathcal{N}_o$ that satisfies: $p^* \in \left\{\arg\max_{p \in \mathcal{N}_o} \langle p, q\rangle\right\} \cap \left\{\arg\min_{p \in \mathcal{N}_o} \delta(p, q')\right\}$. Here, o can be any node on the search path of* GNNS *regarding the query $q' = \mu q$, and $\mathcal{N}_o = \{p | (p, o) \in E\}$ denotes o's neighbor nodes.*

Proof. The search path of the Algorithm 1 on a proximity graph typically expands by selecting a node from the neighbors of the current node that minimizes the Euclidean distance to q for NNS, yet maximizes the inner product for MIPS (more details can be found in [68]). To align the search paths under both metrics, we must unify the node-selection behavior along the path.

This challenge is addressed by localizing the problem: solving for the optimal p^* among the neighbors at each greedy step, akin to the conditions established in Theorem 1. Let l be the number of steps in the path. Define $U = \{\bar{\mu}_i | 1 \le i \le l\}$, the set of $\bar{\mu}$ solving all sub-problems. We have $\forall \mu > \sup(U \cup \{0\})$, Algorithm 1 selects the same node under both IP and Euclidean metrics at each step.

In the special case where multiple neighbors qualify as local MIPS solutions for μq, this ambiguity can be resolved by a simple adaptation to the standard Algorithm 1 protocol: always select the nearest neighbor of μq with smallest ID among the qualified nodes.

In summary, there exists a scalar boundary $\sup(U \cup \{0\})$ which forces the Algorithm 1 under both IP and Euclidean metrics to generate the same search path targeting the scaled query μq. □

B COMPLETE PROOF OF THEOREM 3

Consider a vector database $\mathcal{D} \subset \mathbb{R}^d$ containing n points, where n is sufficiently large to ensure robust statistical properties. Let $\mathcal{G}$ be a SSG [18] defined on $\mathcal{D}$. Assume that the base and query vectors are independently and identically distributed (i.i.d.) and are drawn from the same Gaussian distribution, with each component having zero mean and a variance σ^2. The expected length of the search path L from any randomly selected start node p to a query $q \in \mathbb{R}^d$ can be bounded by $\mathbb{E}[L] < c_0 \frac{\log n + c_1 d}{\log R + c_2 d}$, where R is the max-degree of all possible $\mathcal{G}$, independent with n [18], and c_0, c_1, c_2 are constants.

Proof. Given Theorem 1, 2, along with the monotonic search property of SSG [18], Algorithm 1 identifies a greedy search path where each step minimizes the Euclidean distance to μq while maximizes the IP distance with respect to q, i.e. $\langle r_i, q\rangle > \langle r_{i-1}, q\rangle$. The expected length of the search path can be calculated as:

$$\mathbb{E}[L] = \mathbb{E}_{p \in \mathcal{D}, q \in \mathbb{R}^d}\left[\frac{\langle r_{mip}, q\rangle - \langle p, q\rangle}{\mathbb{E}[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q]}\right]$$

where r_{mip} is the MIPS solution of q. The outer expectation cannot be simplified directly due to potential dependencies among p, r_i, and q. Given that the base and query vectors are finite and drawn from the same distribution, we can enclose them in a hypersphere of diameter $2H$ [72], which is independent of p, r_i, and q. Since $\langle p, q\rangle < H^2$ and $\langle r_{mip}, q\rangle < H^2$, we can derive:

$$\mathbb{E}[L] < \frac{2H^2}{\mathbb{E}_{p \in \mathcal{D}, q \in \mathbb{R}^d}\left[\mathbb{E}\left[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q\right]\right]}$$

Although the exact solution to this inequality is intractable, a practical approach involves approximating both the numerator and the denominator. By applying Extreme Value Theory (EVT) [59], we can derive a tight upper bound for this approximation, ensuring convergence to the bound as n becomes sufficiently large.

As H is the half-diameter, H^2 is the maximum squared norm of the dataset $\mathcal{D}$. We define $M_0 = H^2 = \max_1^n\{X_1^2, ..., X_n^2\}$, where X_i are element-wise i.i.d. samples from $\mathcal{N}(0, \sigma^2)$. Consequently, X^2 follows a chi-square distribution with d degrees of freedom. The Moment Generating Function (MGF) [15] of M_0 is given:

$$MGF(t)_{X^2} = (1 - 2\sigma^2 t)^{-\frac{d}{2}}, 0 < t < \frac{1}{2\sigma^2}$$

From Jensen's Inequality [45] with $\phi(x) = e^{t_0 x}$, $t_0 > 0$, we have:

$$\begin{aligned} e^{t_0 \mathbb{E}[M_0]} &\le \mathbb{E}[e^{t_0 M_0}] = \mathbb{E}\max_{i=1}^{n} e^{t_0 X_i^2} \\ &\le \sum_1^n \mathbb{E}\left[e^{t_0 X_i^2}\right] = n MGF(t_0)_{X^2} \\ \mathbb{E}[M_0] &\le \frac{1}{t_0}\left(\log n + \frac{d}{2}\log(1 - 2\sigma^2 t_0)^{-1}\right) \end{aligned}$$

Here, t_0 is a hyper-parameter that can be optimized within the range $(0, 0.5)$ to enhance the tightness of this bound. When n is sufficiently large, this upper bound serves as a good approximation of $\mathbb{E}[M_0]$.

A similar approach can be applied to the random variable XY for approximating the denominator of Eq. 5, where X and Y element-wise i.i.d. sampled from $\mathcal{N}(0, \sigma^2)$. In this case, XY follows a distribution characterized by a modified Bessel function of the second kind [10]. The MGF for XY is given by:

$$MGF(t)_{XY} = \left(\sqrt{1 - \sigma^2 t_1^2}\right)^{-d}$$

Define $M_1 = \max_0^R\{X_1, .., X_R\}$, where R is the maximum degree of SSG $\mathcal{G}$. Applying Jensen's Inequality again, we obtain:

$$\begin{aligned} e^{t_1 \mathbb{E}[M_1]} &\le E[e^{t_1 M_1}] \\ \mathbb{E}[M_1] &\le \frac{1}{t_1}\left(\log R + d\log(1 - \sigma^2 t_1^2)^{-1}\right) \end{aligned}$$

where $0 < \sigma^2 t_1^2 < 1$. This derived upper bound assumes a distribution centered at the origin. By applying Re-scaling (shifting), we can derive the bound for the case of neighbors of r_{i-1} (points centered at r_{i-1} in a Euclidean proximity graph) as follows:

$$\begin{aligned} \mathbb{E}[\langle r_i, q\rangle] &= \mathbb{E}\left[\max_{m=1}^{R}\{\langle r_m, q\rangle | r_m \in N(r_{i-1})\}\right] \\ &\le \mathbb{E}[\langle r_{i-1}, q\rangle] + \frac{1}{t_1}\left(\log R + d\log(1 - \sigma^2 t_1^2)^{-1}\right) \end{aligned}$$

Substituting them into Eq.(5), we have:

$$\begin{aligned} \mathbb{E}[L] &< \frac{2H^2}{\mathbb{E}_{p \in \mathcal{D}, q \in \mathbb{R}^d}\left[\mathbb{E}\left[\langle r_i, q\rangle - \langle r_{i-1}, q\rangle | r_{i-1}, q\right]\right]} \\ &\approx \frac{\frac{2}{t_0}\left(\log n + \frac{d}{2}\log(1 - 2\sigma^2 t_0)^{-1}\right)}{\frac{1}{t_1}\left(\log R + d\log(1 - \sigma^2 t_1^2)^{-1}\right)} \end{aligned}$$

where R is the maximum degree of $\mathcal{G}$, independent of n and choices of p, q, and r_i[18]. Thus, the outer expectation can be eliminated. When t_0 and t_1 are optimized for the tightest approximation, introducing constants c_0, c_1 simplifies the formula to:

$$\mathbb{E}[L] < c_0 \frac{\log n + c_1 d}{\log R + c_2 d}$$

□

C DETAILS OF COMMERCE100M DATASET.

Commerce100M is derived from real traffic logs of a large-scale e-commerce platform, with embeddings generated by an advanced, pre-trained personalized deep retrieval model. The embeddings have a dimensionality of 48, aligning with industrial norms and resource constraints, where vector dimensions typically range from 16 to 64 due to the vast number of candidates and memory limitations [69, 77]. In this setup, storing 100 million 48-dimensional vectors consumes about 20GB of memory, leaving 30GB for indices under a 50GB per-shard memory allocation for MIPS tasks. Additionally, these vectors are utilized for ranking items among retrieved candidates, with the ranking engine capping the dimension at 64 for efficient similarity ranking.

Query vectors represent joint user and search keyword profiles, while base vectors encompass over 100 million popular grocery items. The query vectors are categorized into three types. Each vector is composed of three segments, each containing 16 dimensions, used in different combinations for various similarity calculations reflecting user behaviors:

- When using part1 (16 dimensions) of the query and part1 (16 dimensions) of the item for similarity calculation, it indicates the user's click tendency.
- When using the concatenation of part1 and part2 of the query with the concatenation of part1 and part2 of the item for similarity calculation (32 dimensional vector), it indicates the user's add-to-cart tendency.
- When using the concatenation of part1, part2, and part3 of the query with the concatenation of part1, part2, and part3 of the item for similarity calculation (48 dimensional vector), it indicates the user's purchase tendency.

Moreover, each query contains a sequence of items with which the user has actually interacted and their corresponding labels. A query may correspond to one or multiple interacted items. Different labels represent different levels of interaction:

- label=3: The user only clicked.
- label=4: The user clicked and added to the cart.
- label=5: The user clicked, added to the cart, and made a purchase.

D ADDITIONAL RESULTS FOR FARGO AND MÖBIUS-GRAPH.

Figure 10 demonstrates the the query performance of Fargo on Commerce100M and Möbius-Graph on DBpedia1M, which is absent in Figure 6 due to either excessively high processing time or low recall.

Specifically, on the Commerce100M dataset, Fargo achieves only 90% recall@100 and exhibits significantly slower performance. Möbius-Graph's recall on DBpedia1M gets stuck at 78% because its vector space transformation approach, which, as discussed in §2, relies on assumptions that are frequently violated in real-world data, thereby impairing performance.

E EMPIRICAL VERIFICATION OF THEORETICAL RESULTS.

Exp-1: Theory Verification (Q1). As detailed in §3, the key to equating MIPS with NNS on a proximity graph is finding an appropriate query-scaling factor μ. We explore this property by conducting GNNS on an ideal PSP index. An ideal PSP can be obtained by executing Algorithm 2 with L set to the cardinality of the base vectors and R set to infinity. We perform this experiment on MNIST. Moreover, we evaluate Theorem 4 on synthetic datasets based on standard normal distribution, about the relationship between top-k MIPS solutions and the Euclidean neighborhood around top-1 MIPS solution.

Duality of NNS and MIPS on Proximity Graph. The case study is conducted by: (1) Randomly select a query and run GNNS with varying μ values: μ in $\{0.1, 1, 10, 100, 500\}$; (2) Trace node IDs visited during the search up to a maximum of 15 steps, as MIPS usually reaches its solution within this limit. The results, as shown in Figure 11, demonstrate that the search paths of MIPS and NNS diverge initially but increasingly overlap as μ increases. At $\mu = 100$, the search paths coincide completely. Furthermore, we assessed the average overlap ratio of search paths for NNS and MIPS, and recall@100 for NNS, across all queries on MNIST under different μ (as shown in Table 4). The result indicates that while precisely defining the lower bound of μ for every possible query is impractical, a sufficiently large μ can effectively unify the MIPS and NNS paths for all queries. Sub-optimal μ does not lead to collapsed results, yet yields reasonably good candidates due to highly overlapped paths.

Overlap Between K-MIPS and Nearest Neighbors. Theorem 4 reveals a high confidence of the overlap between top-k MIPS solutions and the neighborhood around the top-1 MIPS solution. We evaluate how the overlap ratio changes with s on standard Gaussian distribution datasets ($\mathcal{N}(0, 1)$). The results shown in Figure 12 verify our analysis that after the top-k MIPS solutions are distributed around the top-1 MIPS solution's Euclidean neighborhood with a high confidence. With higher dimensions and larger Euclidean distance range, the overlap gradually goes down, which aligns with intuition and the curse of dimensionality. In higher dimensional spaces, the relationship between vectors are more complicated. The duality of Euclidean and IP metric will degrade. Luckily, most real-world datasets, though with high dimension, lie on a very low-dimension manifold. For example, the intrinsic dimension of 960-dimensional GIST1M dataset is about 20+, making the search on it as simple as on a Gaussian random dataset of 20 dimensions.

F DETAILED DISCUSSION ON MAIN EXPERIMENTS.

Exp-2: Performance Assessments (Q2). The proposed method is thoroughly assessed over all datasets on both query processing cost and indexing overhead, as presented in Figure 6 and 7. The cases,

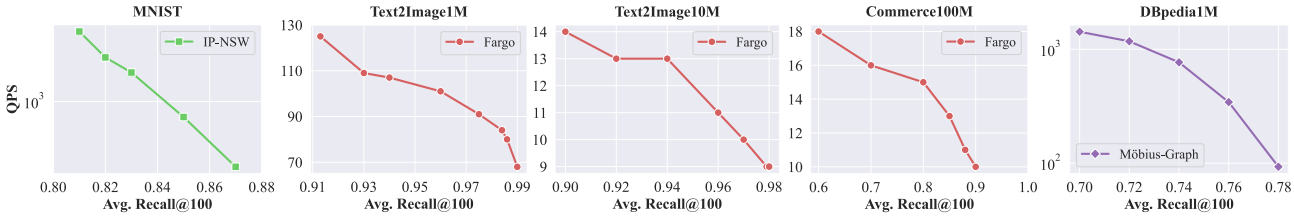

Figure 10: Query performance of ip-NSW, Fargo and Möbius-Graph.

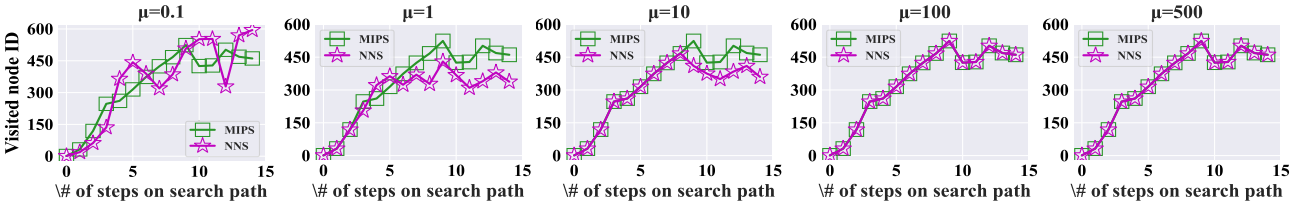

Figure 11: A case study on the overlap of MIPS's and NNS's search paths with different configuration of μ.

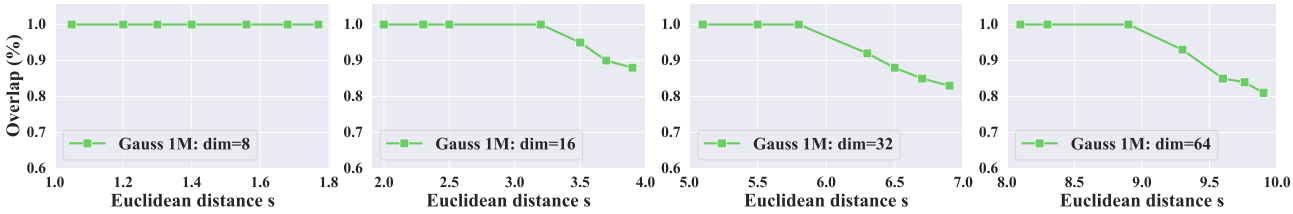

Figure 12: A case study on the overlap of overlap between k-MIPS solutions and the neighborhood around 1-MIPS solution.

where a method is *absent in the figure*, indicates either excessively high processing time or low recall.

Query Processing Performance. The top two rows in Figure 6 depict queries per second (QPS) at different recall levels for all methods, where the curves to the upper right corner indicate better performance (i.e., the ability of a method to quickly process queries while maintaining high accuracy). The two bottom rows in Figure 6 depict the number of distance computations at different recall levels for all graph-based methods, where the desired zone is located in the lower right corner (i.e., fewer distance computations required to achieve a high recall level). Log scales are applied to ensure that differences across a broad range of values are clearly visible.

PSP consistently delivers top-tier performance across all datasets, notably outperforming all baselines in most scenarios. For example, PSP achieves a 4× speedup and a 40% speedup over Möbius-Graph (2^{nd} place) at high recall levels on the MNIST dataset and Commerce100M dataset respectively. This demonstrates PSP's superior efficiency with high-dimensional multi-hot vectors and in large-scale industrial applications. Notably, PSP shows exceptional performance on the MNIST dataset, attributable to its effectiveness with sparse vectors. For sparse data, only non-zero entries in corresponding positions contribute to the IP value, and vectors with larger norms often have more non-zero entries. PSP ensures robust connectivity inherent from its theoretical foundations, facilitating search navigation toward these large-norm vectors. However, other graph-based methods might struggle with connectivity issues, and non-graph based methods often fail to efficiently index large-norm points, which appears to be 'outliers' in their perspective.

The number of distance computations is a crucial metric for determining the efficiency of graph-based methods. It provides a clear indication by eliminating potential system-related biases. Our proposed PSP significantly outperforms all graph-based baselines, achieving an average 37% reduction in distance computations across eight datasets at their respective highest recall@100 levels.

PSP$^-$ represents a basic NSSG without optimizations such as EF, SN, and AET. Conducting IP-based GNNS on PSP$^-$ yields performance that, while not as robust as PSP, still significantly surpasses other baselines across various datasets, achieving an average 20% speedup and 25% speedup excluding Music100. This supports our theoretical analysis in Section 3: MIPS can be efficiently executed on a Euclidean graph without the need for space transformation and distortion. Importantly, carefully designed edge-pruning significantly boosts search efficiency through enhanced edge sparsity and efficient routing. The additional optimizations demonstrate a limited improvement (about 6%) on MNIST as this dataset is least affected by the biased solution problem compared to others.

Fargo slightly outperforms PSP at high recall region on the Netflix dataset, however it generally underperforms on seven others. Specifically, on the Commerce100M dataset, Fargo achieves only 90% recall@100 and exhibits significantly slower performance compared to other methods (hence not shown in the figure). This stems from the inefficiency of LSH in indexing the space [38] and the potential structure distortion of the XBOX transformation [9].

ScaNN excels on the DBpedia100K dataset but struggles on others, reflecting the sensitivity of quantization methods to data distribution and scale. Quantization can effectively reduce distance

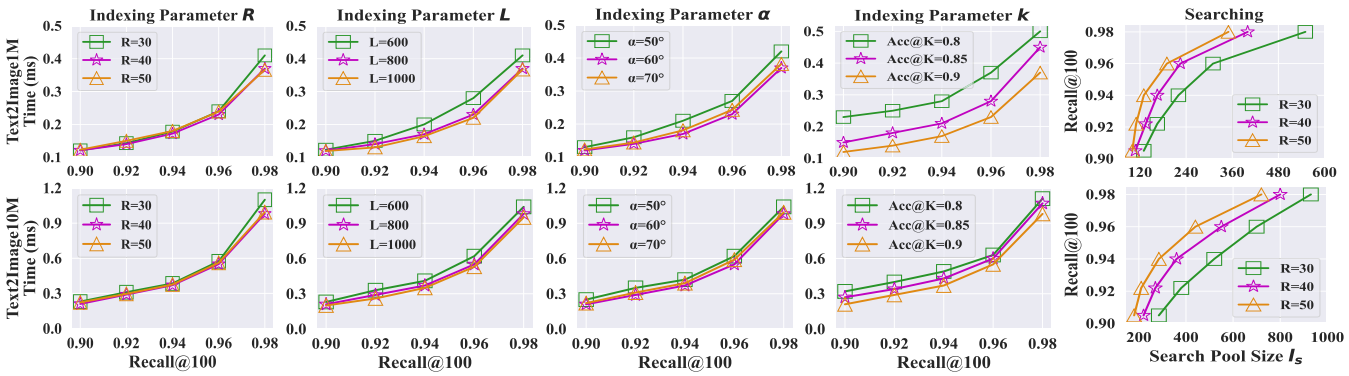

Figure 13: Parameter sensitivity experimental results for key parameters and factors on Text2image1M and Text2image10M

calculation costs when dimensionality is high and dimensions are highly correlated. However, for datasets not amenable to quantization, such as those from IID distributions or in large scales, their efficiency substantially declines [38], e.g., ScaNN is stuck at a recall@100 of only 54% with 120 QPS on Commerce100M dataset, lagging far behind others and therefore is omitted from the figure.

ip-NSW and ip-NSW+: ip-NSW consistently lags behind PSP across all datasets due to high density of the graph. Although theoretically robust, it faces precision limitations even on small datasets like Netflix and MNIST. ip-NSW+, an enhanced version, resolves some precision issues but still cannot consistently improve the performance. In particular, on datasets such as Text2Image1M, Text2Image10M, and DBpedia1M, ip-NSW+ performs notably worse than ip-NSW in high-recall regions. This is due to the extra angular graph introduced is sensitive to data distribution and can negatively impact performance in some cases.

Möbius-Graph stands out among graph-based competitors on the Commerce100M dataset but encounters precision bottleneck on Music100, being stuck at 97%. Worse still, its recall on DBpedia1M gets stuck at 78%, leading to its exclusion from the figure. These issues stem from its vector space transformation approach, which, as discussed in §2, relies on assumptions that are frequently violated in real-world data, thereby impairing performance.

NAPG performs on-par with or even worse than IP-NSW/NSW+ for two main reasons. First, they implement the index upon IP-NSW/NSW+ but their edge-selection strategy does not sparsify the graph, which is proved to be crucial in Theorem 3. Second, their approach involves constructing the graph based on a non-continuous metric, specifically linking neighbors using a scaled IP metric $\alpha\langle p, q\rangle$, where α varies with the norms of the neighbors. This design, based on heuristics rather than on solid theoretical backing, lacks support for such an unconventional metric space.

Indexing Performance. Figure 7 illustrates the index memory footprint and indexing time for different methods. PSP maintains a moderate memory footprint and indexing time across all datasets. Remarkably, on the Commerce100M dataset, PSP requires only 12 GB of index size and 13 hours for indexing, enabling daily updates. This is critical for maintaining AI model performance in response to evolving business metrics. In contrast, ip-NSW+ takes over a day to build on the same dataset. Fargo, while the fastest and least memory-intensive, suffers from poor query processing performance. The indices created by ScaNN, ip-NSW, ip-NSW+, Möbius-Graph, and NAPG consume 1.5 to 2 × the memory footprint of PSP, limiting their practical usability. For graph-based methods, the high-dimensional vectors dominate the memory use. these methods have an index size approximately 3 × larger than PSP, excluding the data. The large index sizes of ip-NSW and ip-NSW+ result from their inability in pruning redundant edges effectively, underscoring the value of our theoretical framework. Meanwhile, ScaNN's extensive memory usage is primarily due to maintaining large tree structures to enhance the search precision.

Exp-2 Summary. We attribute PSP's superiority as follows:

- PSP overcomes the precision bottleneck as it avoids altering the vector space and ensure the graph connectivity. This prevents information loss or structural distortion, the common issues in methods which transform data.
- Leveraging established methods for pruning redundant edges and computations, PSP optimizes search processes effectively based on our rigorous theoretical foundations.
- PSP incorporates adaptations specifically designed to address MIPS challenges, consistently outperforming baselines. It achieves superior search performance while maintaining efficient indices, making it highly suitable for large-scale applications.

G PARAMETER SENSITIVITY EXPERIMENTS.

Parameter Sensitivity. Parameter tuning is often easier when patterns are predictable or when the parameter-performance relationship is smooth and convex. Our evaluations of PSP's parameter sensitivity reveal several key *consistent patterns of optimal configurations*: **(1)** Parameters like R (maximum out-degree) and S (number of nodes added in EF) remain consistent *across different dataset scales*, as depicted in Figure 13. This consistency simplifies parameter tuning on a smaller subsets of data. **(2)** For the minimum angle between edges (α), 60° is optimal across all datasets, aligning with previous findings [18]. **(3)** For other parameters like *Acc@K* (accuracy of the k-NNG to initialize NSSG) and L (candidate size for NSSG), larger values generally enhance accuracy but potentially increase processing time. Users can adjust these parameters based on specific performance requirements and computational constraints. **In summary**, parameter tuning for PSP indexing and searching

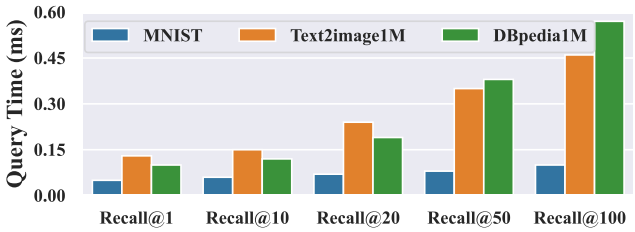

Figure 14: The query process performance of various k of recall on MNIST, Text2image1M and DBpedia1M at 99% recall.

is straightforward because of the predictable and consistent patterns observed across various datasets. The default configuration outlined in §5.1 ensures robust performance and is an alternative to parameter tuning, given PSP's demonstrated superiority across the evaluated datasets.

H ADDITIONAL SCALABILITY EXPERIMENTS.

Scalability with Recall Quantity (k). To assess scalability relative to varying values of k, we adjusted k from $\{1, 10, 20, 50, 100\}$ across the datasets MNIST, Text2Image1M, and DBpedia1M, each differing in cardinality and dimensionality. The results (Figure 14) suggest that both cardinality and dimensionality impact scalability in k. Higher cardinality and dimensionality both contribute to a faster increase in query processing time, aligning with expectations. Despite this, the scaling trend in query processing time remains practical due to the small basis of the time (lower than 1ms per query).

I IMPLICATIONS AND OPEN QUESTIONS

Impact and Implication. This study represents the first attempt to align MIPS and NNS without altering the origin vector space, the first theoretical framework on search efficiency for graph-based MIPS methods, introducing a novel approach that could inspire a range of non-transformative methods. The theoretical insights provided have broad implications, extending beyond just graph-based vector retrieval to potentially influence other database and machine learning tasks.

Limitations. **(1)** The current study's experiments may not fully capture performance variability across extremely skewed data distributions, particularly with outliers or distant clusters of varying norms. **(2)** Our theory does not specifically address redundant computation pruning tailored for the IP metric. Developing solutions for this issue could further enhance the efficiency of our method. **(3)** The research has not yet explored incremental indexing strategies.

Future Works. Building on the current study's implications and its identified limitations, future research should focus on several key areas: **(1)** Further investigations should ascertain the adaptability and robustness of our methods across a broader array of heterogeneous datasets. **(2)** We consider developing acceleration techniques specifically tailored to the MIPS problem, backed by robust theoretical frameworks. **(3)** There is a need to explore incremental indexing techniques that can accommodate real-time updates to data without requiring a complete rebuilding of the index.