Probabilistic Routing for Graph-Based Approximate Nearest Neighbor Search

Kejing Lu [1] **Chuan Xiao** [1 2] **Yoshiharu Ishikawa** [1]

Abstract

Approximate nearest neighbor search (ANNS) in high-dimensional spaces is a pivotal challenge in the field of machine learning. In recent years, graph-based methods have emerged as the superior approach to ANNS, establishing a new state of the art. Although various optimizations for graph-based ANNS have been introduced, they predominantly rely on heuristic methods that lack formal theoretical backing. This paper aims to enhance routing within graph-based ANNS by introducing a method that offers a probabilistic guarantee when exploring a node's neighbors in the graph. We formulate the problem as probabilistic routing and develop two baseline strategies by incorporating locality-sensitive techniques. Subsequently, we introduce PEOs, a novel approach that efficiently identifies which neighbors in the graph should be considered for exact distance calculation, thus significantly improving efficiency in practice. Our experiments demonstrate that equipping PEOs can increase throughput on commonly utilized graph indexes (HNSW and NSSG) by a factor of 1.6 to 2.5, and its efficiency consistently outperforms the leading-edge routing technique by 1.1 to 1.4 times. The code and datasets used for our evaluations are publicly accessible at `https://github.com/ICML2024-code/PEOs`.

1. Introduction

Nearest neighbor search (NNS) is the process of identifying the vector in a dataset that is closest to a given query vector. This technique has found widespread application in machine learning, with numerous solutions to NNS having been proposed. Given the challenge of finding exact answers, practical efforts have shifted towards approximate NNS (ANNS) for efficiency. While some approaches provide theoretical guarantees for approximate answers – typically through locality-sensitive hashing (LSH) (Charikar, 2002; Datar et al., 2004; Andoni et al., 2015) – others prioritize faster search speeds for a desired recall level by utilizing quantization (Jégou et al., 2011; Ge et al., 2014) or graph indexes (Malkov & Yashunin, 2020; Fu et al., 2019; 2022).

Graph-based ANNS, distinguished by its exceptional empirical performance, has become the leading method for ANNS and is widely implemented in vector search tools and databases. During the indexing phase, graph-based ANNS constructs a proximity graph where nodes represent data vectors and edges connect closely located vectors. In the search phase, it maintains a priority queue and a result list while exploring the graph. Nodes are repeatedly popped from the priority queue until it is empty, calculating the distance from their neighbors to the query. Neighbors that are closer than the furthest element in the result list are added to the queue, and the results are accordingly updated.

To further improve the performance of graph-based ANNS, practitioners have introduced various empirical optimization techniques, including routing (Xu et al., 2021; Muñoz et al., 2019; Baranchuk et al., 2019; Chen et al., 2023), edge selection (Fu et al., 2022; Subramanya et al., 2019), and quantization (Lu et al., 2021; Japan, 2023). The primary objective is to minimize distance calculations during neighbor exploration. However, most of these optimizations are heuristic, based on empirical observations (e.g., over 80% of data vectors are less relevant than the furthest element in the results list and thus should be pruned before exact distance calculations (Chen et al., 2023)), making them challenging to quantitatively analyze. Although analyses elucidate the effectiveness of graph-based ANNS (Prokhorenkova & Shekhovtsov, 2020; Indyk & Xu, 2023), they concentrate on theoretical aspects rather than empirical improvements.

In this paper, we investigate routing in graph-based ANNS, aiming to identify which neighbors should be evaluated for distance to the query during the search phase efficiently. Our objective is to bridge the theoretical and practical aspects of ANNS by providing a theoretical guarantee. While achieving an LSH-like guarantee for graph-based ANNS is challenging, we demonstrate that it is feasible to establish a probabilistic guarantee for exploring a node's neighbors. Specifically, we

[1]Graduate School of Informatics, Nagoya University, Japan [2]Graduate School of Information Science and Technology, Osaka University, Japan. Correspondence to: Kejing Lu <lu@db.is.i.nagoya-u.ac.jp>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

introduce probabilistic routing in graph-based ANNS: for a given top node v in the priority queue, an error bound ϵ, and a distance threshold δ, any neighbor u of v with a distance to the query less than δ will be calculated for distance with a probability of at least $1 - \epsilon$. Addressing the probabilistic routing problem yields several benefits: First, it ensures that ANNS explores the most promising neighbors (less than 20% of all neighbors, as observed in (Chen et al., 2023)) with high probability, facilitates quantitative analysis of a search algorithm's effectiveness. Second, by devising probabilistic routing algorithms that accurately and efficiently estimate distances, we can significantly enhance practical efficiency. Third, the theoretical framework ensures consistent performance across different datasets, contrasting with heuristic approaches that may result in high estimation errors and consequently, lower recall rates.

To address the probabilistic routing problem, we initially integrate two foundational algorithms from existing research – SimHash (Charikar, 2002) and (reverse) CEOs (Pham, 2021) – into graph-based ANNS. Subsequently, we introduce a novel routing algorithm, namely **P**artitioned **E**xtreme **O**rder **S**tatistics (PEOs), characterized by the following features:

(1) PEOs utilizes space partitioning and random projection techniques to estimate a random variable which represents the angle between each neighbor and the query vector. By aggregating projection data from multiple subspaces, we substantially reduce the variance of the estimated random variable's distribution, thereby enhancing the accuracy of neighbor-to-query distance estimations.

(2) Through comprehensive analysis, we show that PEOs addresses the probabilistic routing problem within a user-defined error bound ϵ ($\epsilon \leq 0.5$). The algorithm introduces a parameter L, denoting the number of subspaces in partitioning. An examination of L's influence on routing enables us to identify an optimal parameter configuration for PEOs. Comparative analysis with baseline algorithms reveals that an appropriately selected L value yields a variance of PEOs' estimated random variable than that obtained via SimHash-based probabilistic routing and the reverse CEOs-based probabilistic routing is a special case of PEOs with $L = 1$.

(3) The implementation of PEOs is optimized using pre-calculated data and lookup tables, facilitating fast and accurate estimations. The use of SIMD further enhances processing speed, allowing for the simultaneous estimation of 16 neighbors and leveraging the data locality, a significant improvement over conventional methods that require accessing raw vector data stored disparately in memory.

Our experiments encompass tests on several publicly accessible datasets. By integrating PEOs with HNSW (Malkov & Yashunin, 2020) and NSSG (Fu et al., 2022), two predominant graph indexes for ANNS, we achieve a reduction in the necessity for exact distance calculations by 70% to 80%, thereby augmenting queries per second (QPS) by 1.6 to 2.5 times under various recall criteria. Moreover, PEOs demonstrates superior performance to FINGER (Chen et al., 2023), a leading-edge routing technique, consistently enhancing efficiency by 1.1 to 1.4 times while reducing space costs.

2. Related Work

Various types of ANNS approaches have been proposed, encompassing tree-based approaches (Curtin et al., 2014), hashing-based approaches (Andoni & Indyk, 2008; Lu et al., 2018; Lei et al., 2019; Lu & Kudo, 2020; Andoni et al., 2015; Pham & Liu, 2022), quantization based approaches (Jégou et al., 2011; Ge et al., 2014; Babenko & Lempitsky, 2016; Guo et al., 2020; Sun et al., 2023), learn-to-index-based approaches (Gupta et al., 2022; Li et al., 2023), and graph-based approaches (Malkov & Yashunin, 2020; Subramanya et al., 2019; Fu et al., 2019; 2022). Among these, graph-based methods are predominantly considered state-of-the-art (SOTA). To enhance the efficiency of graph-based ANNS, optimizations can be broadly categorized into: (1) routing, (2) edge-selection, and (3) quantization, with these optimizations generally being orthogonal to one another. Given our focus on routing, we briefly review relevant studies in this domain. TOGG-KMC (Xu et al., 2021) and HCNNG (Muñoz et al., 2019) employ KD trees to determine the direction of the query, thereby restricting the search to vectors within that specified direction. Despite fast estimation, it tends to yield suboptimal query accuracy, limiting its effectiveness. FINGER (Chen et al., 2023) estimates the distance of each neighbor to the query. Specifically, for each node, it generates promising projected vectors locally to establish a subspace, and then applies collision counting, as in SimHash, to approximate the distance in each visited subspace. Learn-to-route (Baranchuk et al., 2019) learns a routing function, utilizing additional representations to facilitate optimal routing from the starting node to the nearest neighbor. In addition to these methods that accelerate ANNS itself, learning representations with a coarse granularity (Kusupati et al., 2022) may help reduce the dimensionality and yield a faster search. Such method is orthogonal to the techniques proposed in this paper and can be further applied to speed up ANNS.

3. Problem Definition

Definition 3.1 (Nearest Neighbor Search (NNS)). Given a query vector $\boldsymbol{q} \in \mathbb{R}^d$ and a dataset of vectors $\mathcal{O}$, find the vector $\boldsymbol{o}^* \in \mathcal{O}$ such that $dist(\boldsymbol{q}, \boldsymbol{o}^*)$ is the smallest.

For distance function $dist(\cdot, \cdot)$, two widely utilized metrics are ℓ_2 distance and angular distance. Maximum inner product search (MIPS), closely related to NNS, aims to identify the vector that yields the maximum inner product with the query

Algorithm 1: Graph-based ANNS with routing

```
   Input  :Query $\boldsymbol{q}$, # results $K$, graph index $G$
   Output :$K$-NN of $\boldsymbol{q}$
1  $R \leftarrow \emptyset$ ;          /* an ordered list of results, $|R| \leq efs$ */
2  $P \leftarrow \{$ entry node $v_0 \in G \}$ ;        /* a priority queue */
3  while $P \neq \emptyset$ do
4      $v \leftarrow P.pop()$;
5      foreach unvisited neighbor $u$ of $v$ do
6          if $|R| < efs$ then $\delta \leftarrow \infty$;
7          else $p \leftarrow R[efs]$, $\delta \leftarrow dist(\boldsymbol{p}, \boldsymbol{q})$;
8          if RoutingTest($\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{q}, \delta$) = true then
9              if $dist(\boldsymbol{u}, \boldsymbol{q}) < \delta$ then
10                 $R.push(u)$, $P.push(u)$;
11 return ($\{ R[1], \ldots, R[K] \}$)
```

vector. We elaborate on extension to MIPS in Appendix D.2 and plan to assess its performance in future work.

There is a significant interest not only in identifying a singular nearest neighbor but also in locating the top-K nearest neighbors, a task referred to as K-NN search. It is a prevailing view that calculating exact NNS results poses a considerable challenge, whereas determining approximate results suffices for addressing many practical applications (Qin et al., 2021). Notably, many SOTA ANNS algorithms leverage a graph index, where each vector in $\mathcal{O}$ is linked to its nearby vectors.

The construction of a graph index can be approached in various ways, e.g., through a KNN graph (Dong et al., 2011), HNSW (Malkov & Yashunin, 2020), NSG (Fu et al., 2019), and the improved variant NSSG (Fu et al., 2022). Among these, HNSW stands out as the most extensively adopted model, implemented in many ANNS platforms such as Faiss and Milvus. Given a graph index $G = (V, E)$ built upon $\mathcal{O}$, traversal of this graph enables the discovery of ANNS results, as delineated in Algorithm 1. The search initiates from an entry node $v_0 \in G$, maintaining R, an ordered list that contains no more than efs ($efs \geq K$) – a list size parameter – results identified thus far. Neighbors of the entry node in the graph are examined against $\boldsymbol{q}$ for proximity and added to a priority queue if they are closer to the query than the most distant element in R or if R is not full. Nodes are popped from the priority queue to further explore their adjacent nodes in the graph, until the priority queue becomes empty. It is noteworthy that practical search operations may extend beyond those depicted in Algorithm 1, e.g., by employing pruning strategies to expedite the termination of the search process (Malkov & Yashunin, 2020). Algorithms designed for graph-based ANNS with angular distance can be adapted to accommodate ℓ_2 distance, since calculating the ℓ_2 distance between $\boldsymbol{q}$ and $\boldsymbol{o} \in \mathcal{O}$ merely involves determining the angle between them during graph traversal: $\ell_2(\boldsymbol{q}, \boldsymbol{o})^2 = \|\boldsymbol{q}\|^2 + \|\boldsymbol{o}\|^2 - 2\boldsymbol{q}^\top \boldsymbol{o}$, where $\|\boldsymbol{q}\|$ is fixed and $\|\boldsymbol{o}\|$ can be calculated beforehand.

While naive graph exploration entails calculating the exact distance for all neighbors, a *routing* test can be applied to assess whether a neighbor warrants exact distance calculation. An efficient routing algorithm can substantially enhance the performance of graph-based ANNS. We study routing algorithms with the following probability guarantee.

Definition 3.2 (Probabilistic Routing)**.** Given a query vector $\boldsymbol{q}$, a node v in the graph index, an error bound ϵ, and a distance threshold δ, for an arbitrary neighbor u of v such that $dist(\boldsymbol{u}, \boldsymbol{q}) < \delta$, if a routing algorithm returns true for u with a probability of at least $1 - \epsilon$, then the algorithm is deemed to be $(\delta, 1 - \epsilon)$-routing.

Given our interest in determining whether a neighbor has the potential to refine the temporary results, our focus narrows to the case when δ equals the distance between $\boldsymbol{q}$ and the most distant element in the result list R, with $|R| = efs$ (Lines 6 – 7, Algorithm 1).

4. Baseline Algorithms

To devise a baseline algorithm for probabilistic routing, we adapt SimHash (Charikar, 2002) and CEOs (Pham, 2021) for routing test, which were designed for the approximation with recall guarantee for ANNS and MIPS, respectively.

4.1. SimHash Test

SimHash, a classical random projection-based LSH method for the approximation of angular distance, has the following result (Charikar, 2002).

Lemma 4.1. ***(SimHash)*** *Given $\boldsymbol{u}$, $\boldsymbol{q}$, and m random vectors $\{ \boldsymbol{a}_i \}_{i=1}^m \sim \mathcal{N}(0, I^d)$, the angle θ between $\boldsymbol{u}$ and $\boldsymbol{q}$ can be estimated as*

$$\hat{\theta} = \frac{\pi}{m} \sum_i [\operatorname{sgn}(\boldsymbol{u}^\top \boldsymbol{a}_i) \neq \operatorname{sgn}(\boldsymbol{q}^\top \boldsymbol{a}_i)]. \tag{1}$$

Based on the above lemma, we design a routing test:

$$\textbf{SimHash Test:} \quad \#Col(\boldsymbol{u}, \boldsymbol{q}) \geq T_\epsilon^{\text{SimHash}}(\boldsymbol{u}, \boldsymbol{q}, \delta, m) \tag{2}$$

where $\#Col$ denotes the collision number of the above random projection along $\{ \boldsymbol{a}_i \}_{i=1}^m$, and $T_\epsilon^{\text{SimHash}}(\boldsymbol{u}, \boldsymbol{q}, \delta, m)$ denotes a threshold determined by ϵ and δ. A neighbor u of v passes the routing test iff. it satisfies the above condition. By careful setting of the threshold, we can obtain the desired probability guarantee. Besides probabilistic routing, SimHash has also been used in FINGER (Chen et al., 2023) to speed up graph-based ANNS, serving as one of its building blocks but utilized in a heuristic way.

It can be seen that the result of the above SimHash test is regardless of v. This means that if a node u fails in a test, it will never be inserted into the result set or the priority queue. Since this will compromise the recall of ANNS, we design the following remedy by regarding the neighbors of v as residuals w.r.t. v: for each neighbor u of v, let $\boldsymbol{e} = \boldsymbol{u} - \boldsymbol{v}$, and this residual can be associated with the edge e from v to u in the graph index. $\boldsymbol{e}$ is then used instead of $\boldsymbol{u}$ in the SimHash test. As such, the routing test becomes dependent on $\boldsymbol{v}$, and a threshold $T_\epsilon^{\text{SimHash}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{q}, \delta, m)$ can be derived by the Hoeffding's inequality applied to the Binominal distribution. Hence a neighbor u may fail to pass the test w.r.t. v but succeed in the test w.r.t. node v' in the graph index. Moreover, to model the angle between $\boldsymbol{e}$ and $\boldsymbol{q}$ in the routing test, we normalize $\boldsymbol{q}$ to $\boldsymbol{q}'$ (and optionally, $\boldsymbol{e}$ to $\boldsymbol{e}'$) to simplify calculation. We discuss algorithms in the context of the above remedy hereafter.

4.2. RCEOs Test

In the realm of LSH, Andoni et al. proposed Falconn (Andoni et al., 2015) for angular distance whose basic idea is to find the closest or furthest projected vector to the query and record such vector as a hash value, leading to a better search performance than SimHash. Pham and Liu (Pham & Liu, 2022) employed Concomitants of Extreme Order Statistics (CEOs) (Pham, 2021) to record the minimum or maximum projection value, further improving the performance of Falconn. By swapping the roles of query and data vectors in CEOs, we have Reverse CEOs (RCEOs) and it can be applied to graph-based ANNS with probabilistic routing:

Lemma 4.2. *(RCEOs) Given two normalized vectors $\boldsymbol{e}'$, $\boldsymbol{q}'$, and m random vectors $\{\boldsymbol{a}_i\}_{i=1}^m \sim \mathcal{N}(0, I^d)$, and m is sufficiently large, assuming that $\boldsymbol{a}_1 = \operatorname{argmax}_{\boldsymbol{a}_i} |\boldsymbol{e}'^\top \boldsymbol{a}_i|$, we have the following result:*

$$\boldsymbol{q}'^\top \boldsymbol{a}_1 \sim \mathcal{N}(\operatorname{sgn}(\boldsymbol{e}'^\top \boldsymbol{a}_1) \cdot \boldsymbol{e}'^\top \boldsymbol{q}' \sqrt{2 \ln m}, 1 - (\boldsymbol{e}'^\top \boldsymbol{q}')^2). \tag{3}$$

Despite an asymptotic result, it has been shown that m does not need to be very large (≥ 128) to ensure an enough closeness to the objective normal distribution (Pham, 2021).

Based on Lemma 4.2, we design a routing test:

$$\textbf{RCEOs Test:} \quad \boldsymbol{q}'^\top \boldsymbol{a}_1 \geq T_\epsilon^{\text{RCEOs}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{q}, \delta, m) \tag{4}$$

where $\boldsymbol{a}_1 = \operatorname{argmax}_{\boldsymbol{a}_i} |\boldsymbol{e}'^\top \boldsymbol{a}_i|$ and $T_\epsilon^{\text{RCEOs}}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{q}, \delta, m)$ denotes a threshold related to ϵ and δ.

5. Partitioned Extreme Order Statistics (PEOs)

5.1. Space Partitioning

In PEOs, the data space is partitioned into L subspaces. Let $M \subseteq \mathbb{R}^d$ be the original data space. We decompose M into L orthogonal d'-dimensional subspaces $M_1, M_2, \ldots, M_L$, $d' = d/L$. When $L > 1$, we can significantly decrease the variance of the normal distribution in RCEOs ((3)), hence delivering a better routing test. Specifically, (1) RCEOs test is a special case of PEOs test with $L = 1$, and (2) by choosing an appropriate L ($L > 1$), PEOs test outperforms SimHash test while RCEOs test cannot (Appendix C).

5.2. PEOs Test

Following the space partitioning, for each neighbor u of v, $\boldsymbol{e}$ is partitioned to $[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_L]$, where $\boldsymbol{e}_i$ is the sub-vector of $\boldsymbol{e}$ in M_i. The PEOs test consists of the following steps.

(1) (**Orthogonal Decomposition**) We decompose $\boldsymbol{e}$ as $\boldsymbol{e} = \boldsymbol{e}_{reg} + \boldsymbol{e}_{res}$ such that $\boldsymbol{e}_{reg} \perp \boldsymbol{e}_{res}$ and the direction of $\boldsymbol{e}_{reg}$ is determined as follows:

$$\frac{\boldsymbol{e}_{reg}}{\|\boldsymbol{e}_{reg}\|} = [\frac{\boldsymbol{e}_1}{\sqrt{L}\|\boldsymbol{e}_1\|}, \frac{\boldsymbol{e}_2}{\sqrt{L}\|\boldsymbol{e}_2\|}, \ldots, \frac{\boldsymbol{e}_L}{\sqrt{L}\|\boldsymbol{e}_L\|}]. \tag{5}$$

We call $\boldsymbol{e}_{reg}$ the regular part of $\boldsymbol{e}$ and $\boldsymbol{e}_{res}$ the residual part of $\boldsymbol{e}$. Besides, we introduce two weights w_{reg} and w_{res} such that $w_{reg} = \|\boldsymbol{e}_{reg}\|/\|\boldsymbol{e}\|$ and $w_{res} = \|\boldsymbol{e}_{res}\|/\|\boldsymbol{e}\|$.

(2) (**Generating Projected Vectors**) In each M_i, we independently generate m projected vectors $\{\boldsymbol{a}_{\boldsymbol{j}}^{\boldsymbol{i}}\}_{j=1}^m$, where $\boldsymbol{a}_{\boldsymbol{j}}^{\boldsymbol{i}} \sim \mathcal{N}(0, I^{d' \times d'})$. In the original space M, we independently generate m projected vectors $\{\boldsymbol{b}_{\boldsymbol{j}}\}_{j=1}^m$ such that $\boldsymbol{b}_{\boldsymbol{j}} \sim \mathcal{N}(0, I^{d \times d})$.

(3) (**Collection of Extreme Values**) In each M_i, we collect $L + 1$ extreme values that yield the greatest inner products with the projected vectors: $e[i] = \operatorname{sgn}(\boldsymbol{e}_i^\top \boldsymbol{a}_{\boldsymbol{j}}^{\boldsymbol{i}}) j$, where $j = \operatorname{argmax}_j |\boldsymbol{e}_i^\top \boldsymbol{a}_{\boldsymbol{j}}^{\boldsymbol{i}}|$, and $e[0] = \operatorname{sgn}(\boldsymbol{e}_{res}^\top \boldsymbol{b}_{\boldsymbol{j}}) j$, where $j = \operatorname{argmax}_j |\boldsymbol{e}_{res}^\top \boldsymbol{b}_{\boldsymbol{j}}|$.

(4) (**CDF of Normal Distribution**) Let $\mathcal{N}_{\min}^{\boldsymbol{e},x}$ be the following normal distribution associated with $\boldsymbol{e}$:

$$\mathcal{N}_{\min}^{\boldsymbol{e},x} = \mathcal{N}(x\sqrt{2L \ln m}, w_{reg}^2 + L w_{res}^2 - \frac{Lx^2}{L+1}) \tag{6}$$

where $0 < x < 1$. Let $F_{\boldsymbol{e},x}$ be the CDF of $\mathcal{N}_{\min}^{\boldsymbol{e},x}$. We define $F_{\boldsymbol{e}}^{-1}(x, z)$ such that $F_{\boldsymbol{e},x}(F_{\boldsymbol{e}}^{-1}(x, z)) = z$, where $0 < z \leq 0.5$. Note that $F_{\boldsymbol{e}}^{-1}$ can be well-defined since $F_{\boldsymbol{e},x}(z)$ is a monotone function of x when z is fixed. When setting $z = \epsilon$, we write $F_{\boldsymbol{e},\epsilon}^{-1}(x) = F_{\boldsymbol{e}}^{-1}(x, \epsilon)$.

(5) (**Query Projection**) Given query $\boldsymbol{q}$, we normalize it to $\boldsymbol{q}'$ and calculate the inner products with the projected vectors to obtain two values $H_1(\boldsymbol{e})$ and $H_2(\boldsymbol{e})$ w.r.t. $\boldsymbol{e}$:

$$H_1(\boldsymbol{e}) = \sum_i \operatorname{sgn}(e[i])(\boldsymbol{q}_i'^\top \boldsymbol{a}_{|e[i]|}^i), \tag{7}$$

$$H_2(\boldsymbol{e}) = \operatorname{sgn}(e[0])(\boldsymbol{q}'^\top \boldsymbol{b}_{|e[0]|}). \tag{8}$$

(6) (**Routing Test**) With $H_1(\boldsymbol{e})$ and $H_2(\boldsymbol{e})$, we calculate $A_r(\boldsymbol{e})$ as follows for ℓ_2 distance:

$$A_r(\boldsymbol{e}) = \frac{\|\boldsymbol{u}\|^2/2 - r - \boldsymbol{v}^\top \boldsymbol{q}}{\|\boldsymbol{q}\|\|\boldsymbol{e}\|} \tag{9}$$

where $r = \|\boldsymbol{p}\|^2/2 - \boldsymbol{p}^\top \boldsymbol{q}$ and p is the furthest element to $\boldsymbol{q}$ in the temporary result list R. It is easy to see that $\delta^2 - 2r = \|\boldsymbol{q}\|^2$ when δ captures the ℓ_2 distance from $\boldsymbol{q}$ to the furthest element in R. For angular distance, we remove the norms $\|\boldsymbol{u}\|^2/2$ and $\|\boldsymbol{p}\|^2/2$ in r to obtain $A_r(\boldsymbol{e})$.

With $A_r(\boldsymbol{e})$, we design a routing test for u: If $A_r(\boldsymbol{e}) \geq 1$, it returns false. If $A_r(\boldsymbol{e}) \leq 0$, it returns true. In case $0 < A_r(\boldsymbol{e}) < 1$, we calculate $H(\boldsymbol{e})$ and $T_r(\boldsymbol{e})$ as follows:

$$H(\boldsymbol{e}) = w_{reg} H_1(\boldsymbol{e}) + \sqrt{L} w_{res} H_2(\boldsymbol{e}), \tag{10}$$

$$T_r(\boldsymbol{e}) = F^{-1}_{\boldsymbol{e},\epsilon}(A_r(\boldsymbol{e})). \tag{11}$$

Then, the test returns true iff. the following condition is met.

$$H(\boldsymbol{e}) - T_r(\boldsymbol{e}) \geq 0. \tag{12}$$

5.3. Implementation of PEOs

In the PEOs test, Steps (1), (2), and (3) can be pre-calculated during index construction. Since space partitioning may result in unbalanced norms in subspaces, we can optionally permute the dimensions so that the norms of all $\boldsymbol{e}_i$'s ($1 \leq i \leq L, e \in E$) in the graph are as close to each other as possible (Appendix D.1). Such permutation does not affect the topology of the graph or the theoretical guarantee of PEOs. Steps (4), (5), and (6) are calculated during the search. $H(\boldsymbol{e})$ and $T_r(\boldsymbol{e})$ can be calculated efficiently because (1) $\boldsymbol{q}_i'^\top \boldsymbol{a}^i_{|e[i]|}$ and $\boldsymbol{q}'^\top \boldsymbol{b}_{|e[0]|}$ can be calculated for $\boldsymbol{q}$ only once and stored in a projection table, and (2) although the online calculation of $F^{-1}_{\boldsymbol{e},\epsilon}(x)$ is costly, we can build a lookup table containing the quantiles corresponding to the different values of the variance in $\mathcal{N}^{\boldsymbol{e},x}_{\min}$ since such variance is bounded. From the lookup table, we can choose a quantile slightly smaller than the true quantile by employing the monotonicity, which does not affect the correctness of the probability guarantee. An illustration of the implementation of the PEOs test is depicted in Figure 1, with pseudo-code given in Algorithm 2.

Another optimization is the use of SIMD, where 16 edges can be processed at a time. This will significantly accelerate ANNS, because the raw vectors of neighbors are stored separately in the memory and loading them into CPU is costly.

6. Analysis of PEOs

6.1. Probability Guarantee of PEOs

We assume $\|\boldsymbol{e}\| = \|\boldsymbol{q}\| = 1$ and let θ denote the angle between them. By the independence of projected vectors in different subspaces and the result in Lemma 4.2, we have $H_1(\boldsymbol{e}) \sim \mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$, where $\mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$ is defined as follows:

$$\mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}} = \mathcal{N}(\eta \sum_i \|\boldsymbol{q}_i\| \cos\theta_i, 1 - \sum_i \|\boldsymbol{q}_i\|^2 \cos^2\theta_i) \tag{13}$$

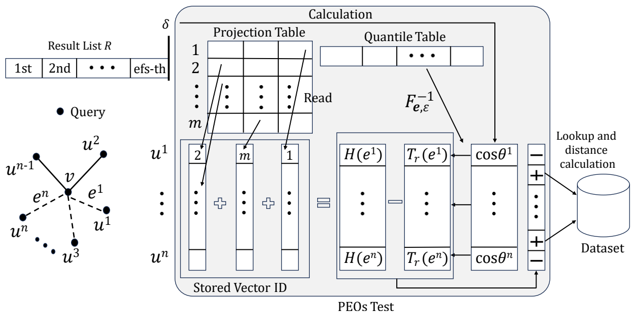


Figure 1. Illustration of the PEOs test. There are n neighbors of v. $\theta^1, \ldots, \theta^n$ denote the angles between $\boldsymbol{e}^1, \ldots, \boldsymbol{e}^n$ and $\boldsymbol{q}$, respectively. u^2 and u^{n-1} pass the test (indicated by "+"). We access their raw vectors from the dataset and calculate their distances to $\boldsymbol{q}$.

Algorithm 2: PEOs Test

```
Input  :query $\boldsymbol{q}$, edge $e = (v, u)$, threshold $r$, projected
        vectors $\{ \boldsymbol{a}^i_j \}$ and $\{ \boldsymbol{b}_j \}$ ($1 \leq i \leq L$,
        $1 \leq j \leq m$), quantile table $Q$
Output :whether $u$ passes the routing test
1 Calculate $A_r(\boldsymbol{e})$;
2 if $A_r(\boldsymbol{e}) \geq 1$ then return (false);
3 if $A_r(\boldsymbol{e}) \leq 0$ then return (true);
4 Normalize $\boldsymbol{q}$ to $\boldsymbol{q}'$;
5 Build a projection table $T_I$ for all $\boldsymbol{q}_i'^\top \boldsymbol{a}^i_j$ and $\boldsymbol{q}'^\top \boldsymbol{b}_j$ ;
    /* only once and used throughout the search */
6 Calculate $H(\boldsymbol{e})$ with $T_I$;
7 Calculate $T_r(\boldsymbol{e})$ with $Q$;
8 if $H(\boldsymbol{e}) \geq T_r(\boldsymbol{e})$ then return (true) ;
9 else return (false) ;
```

where $\eta = \sqrt{2 \ln m}$ and θ_i denotes the angle between $\boldsymbol{e}_i$ and $\boldsymbol{q}_i$. Next, we analyze the relationship between θ and $\mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$. To this end, we introduce the following definition.

Definition 6.1. We define two partial orders $\prec$ and $\preceq$ such that, for two normal distributions $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, $\mathcal{N}(\mu_1, \sigma_1^2) \prec \mathcal{N}(\mu_2, \sigma_2^2)$ iff. $\mu_1 \leq \mu_2$ and $\sigma_1^2 \geq \sigma_2^2$, and $\mathcal{N}(\mu_1, \sigma_1^2) \preceq \mathcal{N}(\mu_2, \sigma_2^2)$ iff. $\mu_1 = \mu_2$ and $\sigma_1^2 \geq \sigma_2^2$.

With the notations defined above, we want to find an appropriate normal distribution $\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}}$ such that $\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}} \prec \mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$ ($\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}} \preceq \mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$ is more favorable) for all adequate pairs $(\boldsymbol{e}, \boldsymbol{q})$'s. We define $\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}}$ as follows, where $e_{\min} = \min_{1 \leq i \leq L} \|\boldsymbol{e}_i\|$ and $e_{\max} = \max_{1 \leq i \leq L} \|\boldsymbol{e}_i\|$:

$$\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}} = \mathcal{N}\left(\frac{(\cos\theta + (e_{\min} - e_{\max}) \sum_i \|\boldsymbol{q}_i\|)\eta}{e_{\max}}, 1 - \cos^2\theta\right). \tag{14}$$

Then, we have the following lemma.

Lemma 6.2. $\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}} \prec \mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$. *When* $\|\boldsymbol{e}_1\| = \cdots = \|\boldsymbol{e}_L\|$, $\tilde{\mathcal{N}}^\theta_{\boldsymbol{e},\boldsymbol{q}} = \mathcal{N}(\cos\theta\sqrt{2L \ln m}, 1 - \cos^2\theta) \preceq \mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}$.

From Lemma 6.2, we can see that, for the case when $e_{\max} - e_{\min}$ is large, we can only get a loose lower bound of $\mathbb{E}[\mathcal{N}^\theta_{\boldsymbol{e},\boldsymbol{q}}]$ due to the impact of unknown θ_i's (although it is possible to get a strict lower bound by solving a linear programming, the calculation is too costly for a fast test), while the estimation of $\mathbb{E}[\mathcal{N}^{\theta}_{\boldsymbol{e},\boldsymbol{q}}]$ is always accurate when $\|\boldsymbol{e}_1\| = \cdots = \|\boldsymbol{e}_L\|$ holds. This explains why we decompose vector $\boldsymbol{e}$ into $\boldsymbol{e}_{reg}$ and $\boldsymbol{e}_{res}$ and deal with them separately. Then, we have the following theorem for the probability guarantee of PEOs.

Theorem 6.3. *(1) (**Probabilistic Guarantee**) Suppose that m is sufficiently large. The PEOs test is $(\delta, 1-\epsilon)$-routing.*

*(2) (**False Positives**) Consider a neighbor u whose distance to $\boldsymbol{q}$ is at least δ. If $\cos\theta \le \tilde{F}^{-1}_{\tilde{\theta}}(\epsilon)$ ($\epsilon \le 0.5$), where $\cos\tilde{\theta} = A_r(\boldsymbol{e})$ and $\tilde{F}_{\theta}$ is the CDF of distribution $\mathcal{N}^{\mathbf{e},\cos\theta}_{\min}/\sqrt{2L\ln m}$. Then the probability that u passes PEOs test is at most $1 - \tilde{F}_{\theta}(\tilde{F}^{-1}_{\tilde{\theta}}(\epsilon))$.*

*(3) (**Variance of Estimation and Comparison to RCEOs**) Suppose that $H(\boldsymbol{e})/\sqrt{2L\ln m} \sim \mathcal{N}_H$, where $\mathcal{N}_H$ is an unknown normal distribution, and let $\mathcal{N}^{\theta}_{opt} = \mathcal{N}(\cos\theta, \sin^2\theta/(2L\ln m))$. When $w_{res} \le 1/(L+1)$,*

$$-\frac{L+2}{L(L+1)^2\ln m} \le \mathrm{Var}[\mathcal{N}_H] - \mathrm{Var}[\mathcal{N}^{\theta}_{opt}] \le \frac{1}{(L+1)^2\ln m} \tag{15}$$

where θ is the (unknown) angle between $\mathbf{e}$ and $\boldsymbol{q}$.

Remarks. (1) The first statement of Theorem 6.3 guarantees that promising neighbors are explored with a high confidence.

(2) The second statement shows that the routing efficiency is determined by the variance of $\mathcal{N}^{\mathbf{e},\cos\theta}_{\min}/\sqrt{2L\ln m}$. Such variance is expected to be as small as possible since a smaller variance leads to a smaller probability of a false positive.

(3) It is easy to see that, for RCEOs with m^L projected vectors, the distribution associated with $\cos\theta$ is $\mathcal{N}^{\theta}_{opt}$. For a comparison, we use $\mathcal{N}_H$ to denote the distribution of $H(\boldsymbol{e})/\sqrt{2L\ln m}$. Clearly, $\mathbb{E}[\mathcal{N}_H] = \mathbb{E}[\mathcal{N}^{\theta}_{opt}]$. On the other hand, the third statement shows that, if w_{res} is a small value, the variances of such two distributions are very close. In this situation, the effect of PEOs is close to that of RCEOs with m^L projected vectors, which explains why PEOs can perform much better than RCEOs empirically.

6.2. Impact of L

Based on the second and the third statements in Theorem 6.3, w_{reg} and w_{res} are critical values which control the routing efficiency. First, we want to show that w_{reg} is generally close to 1. To this end, we calculate $\mathbb{E}[w_{reg}]$ under the assumption that vector $\boldsymbol{e}$ obeys an isotropic distribution. Because prevalent graph indexes (e.g., HNSW) diversify the selected edges in the indexing phase, such assumption is not very strong for the real datasets. Besides, we can permute the dimensions (Sec. 5.3) to make $\boldsymbol{e}$ follow an isotropic distribution. Let $\bar{w}_{reg}(L,d)$ denote $\mathbb{E}_{\boldsymbol{e}\sim U(\mathbb{S}^{d-1})}[w_{reg}]$ ($\bar{w}_{reg}(L,d)$ means that w_{reg} is affected by L and d). Then, we have the following lemma ($d' > 3$).

Lemma 6.4. *Given L, d', and $d' = d/L$,*

$$\bar{w}_{reg}(L,d) \ge \frac{(d'-1)\sqrt{2Ld-3L}}{(d-1)\sqrt{2d'+2\sqrt{3}-6}}. \tag{16}$$

As an example, $\bar{w}_{reg}(L,d) \ge 0.978$, when $d = 128$ and $L = 8$. For other reasonable choice of L w.r.t. d, we can also obtain a $\bar{w}_{reg}(L,d)$ close to 1. Next, we analyze the relationship between L and w_{res} more accurately. To this end, we consider the distribution $\mathcal{N}^{\mathbf{e},\cos\theta}_{\min}/\sqrt{2L\ln m}$. Its expected value is $\cos\theta$ and its variance, which is expected to be as small as possible, as shown in Theorem 6.3, is of great interest to us. For its variance, we particularly focus on the remaining part J_{rel} by removing the part regrading $\cos\theta$, which is a value close to 0 for most of $\boldsymbol{e}$'s. Specifically, J_{rel} is defined as follows:

$$J_{rel}(L) = \frac{1 + (L-1)\mathbb{E}[w^2_{res}(L,d)]}{L}. \tag{17}$$

On the other hand, for $\mathcal{N}^{\theta}_{opt}$, we have $J_{opt}(L) = 1/L$, which corresponds to RCEOs with m^L projected vectors. Due to the effect of w_{res}, there is a difference between $J_{opt}(L)$ and $J_{rel}(L)$. Thus, an appropriate value of L should satisfies the following two requirements:

(1) (**Major**) $J_{rel}(L)$ should be as small as possible.

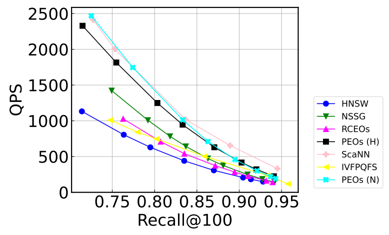
(2) (**Minor**) w_{res} is close to $1/(L+1)$.

Here, the first requirement is to improve the routing efficiency of PEOs, which is more important, and the second one is to measure the deviation in the condition in the third statement (Theorem 6.3). In Figure 3a – 3c, under the assumption of isotropic distribution, we plot the curves of $J_{rel}(L)$, $J_{opt}(L)$ and $\Delta = |w_{res} - [1/(L+1)]|$, for $d = 128$, 384 and 960, respectively. It can be seen that (1) w_{res} increases as L grows, which means that L should not be overlarge because $J_{rel}(L') > J_{rel}(L)$ may occur when $L' > L$, and (2) due to the closeness of J_{opt} and J_{rel}, the effect of PEOs is very close to that of RCEOs with m^L projected vectors when L is small (e.g. $L \le 8$). Based on the analysis above, we set L to 8, 15, 16 for these three dimensions. By varying the value of L, the performance of PEOs on the real datasets with these dimensions is consistent with our analysis (Figures 3d – 3f), which will be elaborated in Sec. 7.3.

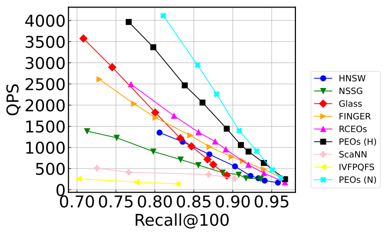
6.3. Computational Cost and Space Cost

We assume that PEO is applied on HNSW. The analysis of other graph indexes (e.g., NSSG) is similar.

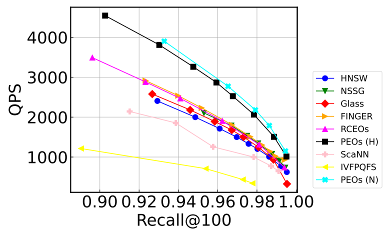
For the time complexity, it is difficult to provide a strict analysis since currently no graph-based method can offer such a guarantee. Nonetheless, if we assume that the length of the search path is ω, where ω can be roughly estimated as $O(\log n)$ in practice, where n is the number of data points,



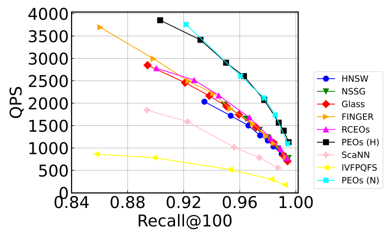
(a) GloVe200-angular, $K = 100$



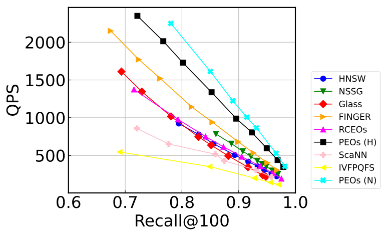
(b) GloVe300-ℓ_2, $K = 100$



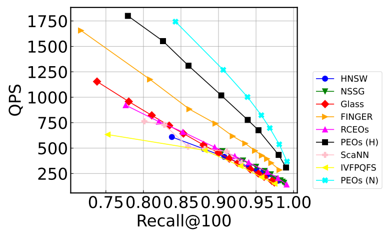
(c) DEEP10M-angular, $K = 100$



(d) SIFT10M-ℓ_2, $K = 100$



(e) Tiny5M-ℓ_2, $K = 100$



(f) GIST-ℓ_2, $K = 100$

Figure 2. Recall-QPS evaluation. PEOs (H) denotes HNSW+PEOs and PEOs (N) denotes NSSG+PEOs. The recalls of Glass and FINGER are lower than 30% on GloVe200 and thus not shown.

then the complexity of HNSW is $O(vd\omega)$, where v is the average out-degree and d is dimension. Thus, the complexity of HNSW+PEOs is $O(v\omega(L+4)+\rho vd\omega)$ (without the consideration of SIMD), where 4 is the number of scalar operations and ρ denotes the filtering ratio, which depends on the data distribution and is around 0.25 in practice (Appendix F.2).

As for the space cost of PEOs, take $M = 32$ as an example (the maximum out-degree is 64). We use a_{16}, a_{32}, a_{48}, a_{64} to denote the number of nodes whose out-degrees lie in $[1, 16]$, $[17, 32]$, $[33, 48]$, and $[49, 64]$, respectively (for the utilization of SIMD). Let $X = 16 \times a_{16} + 32 \times a_{32} + 48 \times a_{48} + 64 \times a_{64}$. Let n denote the number of data points and d denote the dimension. Then, the space complexity is $O(X(L + 4) + 4nd)$, assuming that we use 8 bits to compress every scalar. If we use 16 bits, the complexity is $O(X(L + 8) + 4nd)$. X is dependent on the edge-selection strategy and data distribution. For most cases, X is around $32n$. Such space cost is affordable for 10M-scale and smaller datasets on a single PC (Sec. 7.5). For 100M-scale and larger datasets, we can slightly sacrifice efficiency to significantly reduce space consumption (Sec. 7.6). In addition to the index, space consumption also includes the projection table and quantile lookup table. Both are are very small and can be stored in CPU cache – one is 100 or 1000 floats and the other is $256L$ floats, where L is typically in $[8, 32]$.

7. Experiments

7.1. Datasets and Methods

We use six million-scale datasets and a 100M dataset for scalability test. The statistics can be found in Table 1.

Table 1. Dataset statistics.

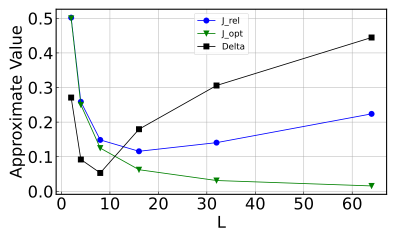
Dataset	Size ($\lvert\mathcal{O}\rvert$)	Dim. (d)	Type	Metric
GloVe200	1,183,514	200	Text	angular
GloVe300	2,196,017	300	Text	ℓ_2
DEEP10M	9,990,000	96	Image	angular
SIFT10M	10,000,000	128	Image	ℓ_2
Tiny5M	5,000,000	384	Image	ℓ_2
GIST	1,000,000	960	Image	ℓ_2
DEEP100M	100,000,000	96	Image	angular

We implement PEOs on two widely used SOTA graph index: HNSW and NSSG. In the experiments, PEOs refers to HNSW+PEOs unless stated otherwise. We select seven competitors: (vanilla) HNSW, (vanilla) NSSG (Fu et al., 2022), Glass (Zilliz, 2023), ScaNN (Guo et al., 2020), IVFPQFS (in Faiss) (Douze et al., 2024), FINGER (Chen et al., 2023), and RCEOs. The default value of K is 100.

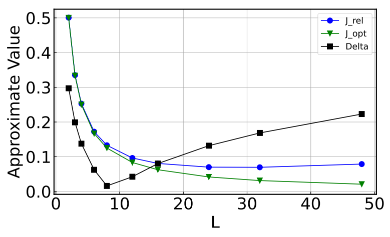
7.2. Queries Per Second (QPS) Evaluation

Figure 2 reports the recall-QPS curves of all the competitors on the million-scale datasets. We have four observations.

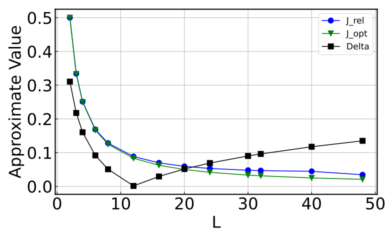
(1) On all the datasets, the winner is PEOs, trailed by FINGER in most cases. This demonstrate that our routing is effective. In particular, PEOs accelerates HNSW by 1.6 to 2.5 times and it is faster than FINGER by 1.1 to 1.4 times. (2) On GloVe300, Tiny5M and GIST, NSSG+PEOs obviously outperforms HNSW+PEOs, despite a smaller performance gap of the two vanilla versions. This is because, compared with HNSW, NSSG achieves a higher search accuracy at the cost of longer time for routing, making the improvement of PEOs over NSSG more significant. (3) The improvement of



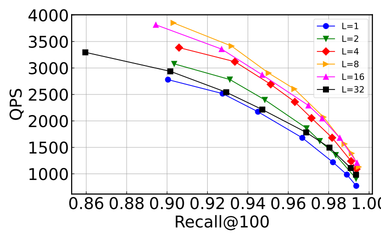
(a) Dimension: 128 (SIFT10M)



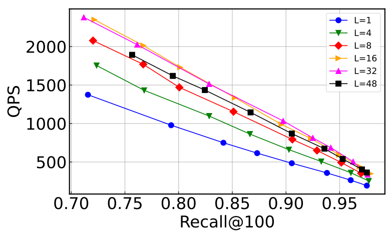
(b) Dimension: 384 (Tiny5M)



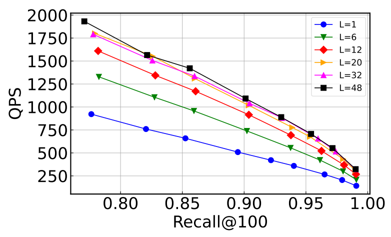
(c) Dimension: 960 (GIST)



(d) SIFT10M-ℓ_2, $K = 100$

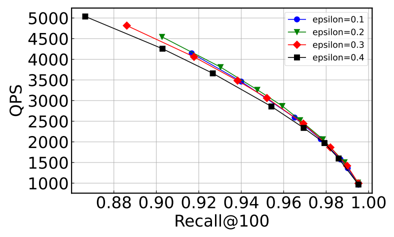


(e) Tiny5M-ℓ_2, $K = 100$

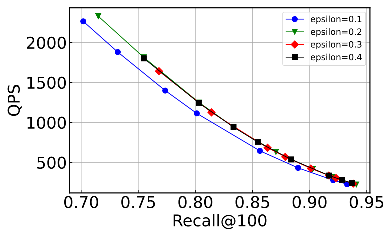


(f) GIST-ℓ_2, $K = 100$

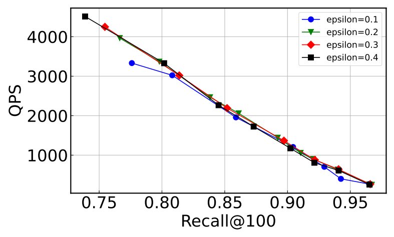
Figure 3. Effect of L. We plot the approximate values of J_{opt}, J_{rel}, and Δ under the isotropic distribution and the empirical performance.



(a) DEEP10M-angular, $K = 100$



(b) GloVe200-angular, $K = 100$



(c) GloVe300-ℓ_2, $K = 100$

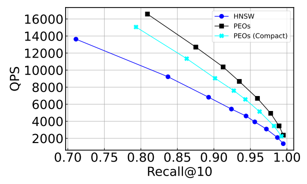
Figure 4. Effect of ϵ on DEEP10M, GloVe200 and GloVe300.

Table 2. Index size and indexing time.

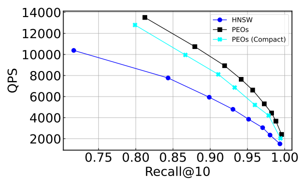
Dataset	Index Size (GB)			Indexing Time (s)		
	HNSW	HNSW+FINGER	HNSW+PEOs	HNSW	HNSW+FINGER	HNSW+PEOs
GloVe200	1.19	3.89 (+2.27x)	2.27 (+0.91x)	737	463+38	794+33
GloVe300	3.02	8.04 (+1.66x)	3.70 (+0.23x)	1310	1408+178	1346+24
DEEP10M	6.14	31.15 (+4.07x)	12.13 (+0.98x)	1245	1103+849	1296+208
SIFT10M	7.34	31.44 (+3.28x)	14.39 (+0.96x)	1490	1308+1025	1536+204
Tiny5M	8.44	20.49 (+1.43x)	12.89 (+0.53x)	2738	2959+1220	2880+158
GIST	3.83	6.12 (+0.60x)	4.64 (+0.21x)	738	790+706	793+40

Table 3. Effect of compact implementation on index size (GB).

Dataset	HNSW	PEOs	PEOs (Compact)
DEEP10M	6.14	12.13 (+0.98x)	7.89 (+0.29x)
SIFT10M	7.34	14.39 (+0.96x)	9.62 (+0.26x)



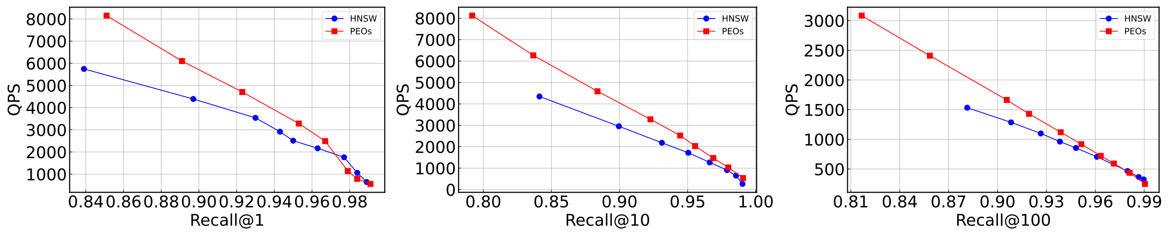
(a) DEEP10M-angular, $K = 10$



(b) SIFT10M-ℓ_2, $K = 10$

Figure 5. Effect of compact implementation on search speed.

PEOs is more significant on the datasets with more dimensions, because calculating the exact distance to the query is more costly on these datasets. (4) On GloVe200, FINGER and Glass report very low recall ($< 30\%$) recall while the improvement of PEOs is still obvious under high recall settings. Specifically, FINGER is also based on routing, but incurs unbounded estimation errors and the errors might be very large on GloVe200, resulting in many false negatives of and rendering the graph under-explored. This result evidences the importance of the probability guarantee of routing.



(a) DEEP100M-angular, $K = 1$ (b) DEEP100M-angular, $K = 10$ (c) DEEP100M-angular, $K = 100$

Figure 6. Recall-QPS evaluation on DEEP100M. $L = 2$ for PEOs.

7.3. Effect of Space Partition Size L

We evaluate the effect of L in HNSW+PEOs and report the results in Figure 3 on the SIFT10M, Tiny5M, and GIST datasets (other datasets are evaluated in Appendix F.3). Figures 3a – 3c show theoretical results, and Figures 3d – 3f are their empirical counterparts. The empirical results are consistent with our analysis in Sec. 6.2. That is, the smaller J_{rel} is, the better the performance is. We also have the following observations. (1) The performance under $L > 1$ is obviously better than that under $L = 1$, showcasing the effectiveness of space partitioning. (2) An L of 32 leads to a worse performance than an L of 8 on the SIFT10M dataset, because the variance is larger when $L = 32$. (3) When $L > 16$ on the GIST dataset, the performance tends to be stable since the variance barely changes when L exceeds 16.

7.4. Effect of Error Bound ϵ

We vary the value of ϵ and report the results in Figure 4. The performance of PEOs under $\epsilon = 0.1$ is slightly worse than those under other ϵ settings. On the other hand, $\epsilon = 0.2$ is consistently the best choice, leading to the best recall-QPS curve. Based on this observation, we suggest users choose $\epsilon = 0.2$ to seek best performance with a guaranteed routing.

7.5. Indexing Cost

In Table 2, the index size of HNSW+PEOs is 1.2x – 2.0x larger than that of HNSW. On the two datasets with lower dimensions, the additional space overheads are more obvious. Meanwhile, FINGER requires more space cost than PEOs due to storing the information of generated subspaces. The indexing time of PEOs is much shorter than the time of graph construction. On the other hand, FINGER spends more indexing time due to its additional subspace for each node.

From the index size results, the amount of additional storage for PEOs is evident for relatively low-dimensional ($d < 200$) datasets (over 90%). Therefore, we propose another implementation, dubbed PEOs (Compact), to alleviate this issue. In this implementation, we use $L = 2$ (a smaller d leads to a smaller L) and 8 bits to compress scalars (the standard one uses 16 bits). In Figure 5 and Table 3, despite slightly sacrificing search speed, a significant reduction in index size can be achieved. Note that PEOs (Compact) is still faster than other graph-based methods compared.

Table 4. Index size and indexing time on DEEP100M.

Method	Index Size (GB)	Indexing Time (s)
HNSW	66.0	8877
HNSW+PEOs	67.9 (+0.029x)	5832+622

7.6. Scalability

We discuss how to tackle the scalability issue on datasets larger than the million scale. There are three ways to reduce the space cost of PEOs: (1) decreasing M, (2) decreasing L, and (3) using only one byte for scalar quantization. When $L \leq 4$, w_{res} is generally very small, meaning that we can set w_{reg} to 1 and w_{res} to 0 for every $\boldsymbol{e}$. In this case, for each neighbor u, we only need L bytes ($2 \leq L \leq 4$) for sub-vector IDs, one byte for the norm of $\boldsymbol{u}$ and one byte for the norm of $\boldsymbol{e}$. Although such setting is not optimal for speed, it can significantly reduce space cost. Following this analysis, we use $L = 2$ and $M = 16$ for PEOs on DEEP100M. Figure 6 and Table 4 show that in most cases, PEOs has 30% performance improvement on HNSW with a 3% additional space cost. For datasets with higher dimensions, the percentage of additional space cost can be smaller.

8. Conclusion

We studied the problem of probabilistic routing in graph-based ANNS, which yields a probabilistic guarantee of estimating whether the distance between a node and the query should be calculated when exploring the graph index for ANNS, thereby preventing unnecessary distance calculation. We considered two baseline algorithms by adapting locality-sensitive approaches to routing in graph-based ANNS, and devised PEOs, a novel approach to this problem. We proved the probabilistic guarantee of PEOs and conducted an empirical evaluation using two popular graph indexes on seven datasets. The results showed that PEOs is effective in enhancing the performance of graph-based ANNS and consistently outperforms the SOTA routing technique by 1.1 to 1.4 times.

Acknowledgement

This work was supported by JST CREST Grant Number JPMJCR22M2, Japan and JSPS Kakenhi Grant Numbers 23H03406, 23K17456, 23K21726, 23K24850, 23K25157.

Impact Statement

In this paper, we studied ANNS, a fundamental problem in the area of machine learning. The proposed routing test algorithm can be plugged into current graph indexes for ANNS. To the best of our knowledge, there is no negative societal impact related to this research.

References

Andoni, A. and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I. P., and Schmidt, L. Practical and optimal LSH for angular distance. In *NeurIPS*, pp. 1225–1233, 2015.

Babenko, A. and Lempitsky, V. S. Efficient indexing of billion-scale datasets of deep descriptors. In *CVPR*, pp. 2055–2063, 2016.

Baranchuk, D., Persiyanov, D., Sinitsin, A., and Babenko, A. Learning to route in similarity graphs. In *ICML*, pp. 475–484, 2019.

Bernhardsson, E. Ann benchmarks. https://github.com/erikbern/ann-benchmarks/, 2024.

Charikar, M. Similarity estimation techniques from rounding algorithms. In Reif, J. H. (ed.), *STOC*, pp. 380–388. ACM, 2002.

Chen, P. H., Chang, W., Jiang, J., Yu, H., Dhillon, I. S., and Hsieh, C. FINGER: fast inference for graph-based approximate nearest neighbor search. In *WWW*, pp. 3225–3235. ACM, 2023.

Curtin, R. R., Ram, P., and Gray, A. G. Fast exact max-kernel search. *Statistical Analysis and Data Mining*, 7(1):1–9, February–December 2014.

Dai, X., Yan, X., Ng, K. K. W., Liu, J., and Cheng, J. Norm-explicit quantization: Improving vector quantization for maximum inner product search. In *AAAI*, pp. 51–58, 2020.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, pp. 253–262, 2004.

Dong, W., Moses, C., and Li, K. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*, pp. 577–586, 2011.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. The faiss library. 2024.

Fu, C., Xiang, C., Wang, C., and Cai, D. Fast approximate nearest neighbor search with the navigating spreading-out graph. *PVLDB*, 12(5):461–474, 2019.

Fu, C., Wang, C., and Cai, D. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(8):4139–4150, 2022.

Ge, T., He, K., Ke, Q., and Sun, J. Optimized product quantization. *IEEE Trans. Pattern Anal. Mach. Intell*, 36(4): 744–755, 2014.

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. Accelerating large-scale inference with anisotropic vector quantization. In *ICML*, pp. 3887–3896, 2020.

Gupta, G., Medini, T., Shrivastava, A., and Smola, A. J. BLISS: A billion scale index using iterative re-partitioning. In *KDD*, pp. 486–495. ACM, 2022.

Indyk, P. and Xu, H. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. *CoRR*, abs/2310.19126, 2023.

Japan, Y. Neighborhood graph and tree for indexing high-dimensional data. https://github.com/yahoojapan/NGT, 2023.

Jégou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell*, 33(1):117–128, 2011.

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., et al. Matryoshka representation learning. *NeurIPS*, 35:30233–30249, 2022.

Lei, Y., Huang, Q., Kankanhalli, M., and Tung, A. Sublinear time nearest neighbor search over generalized weighted space. In *ICML*, pp. 3773–3781, 2019.

Li, W., Feng, C., Lian, D., Xie, Y., Liu, H., Ge, Y., and Chen, E. Learning balanced tree indexes for large-scale vector retrieval. In *KDD*, pp. 1353–1362. ACM, 2023.

Lu, K. and Kudo, M. R2LSH: A nearest neighbor search scheme based on two-dimensional projected spaces. In *ICDE*, pp. 1045–1056, 2020.

Lu, K., Wang, H., Xiao, Y., and Song, H. Why locality sensitive hashing works: A practical perspective. *Inf. Process. Lett.*, 136:49–58, 2018.

Lu, K., Kudo, M., Xiao, C., and Ishikawa, Y. HVS: Hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search. *PVLDB*, 15 (2):246–258, 2021.

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell*, 42(4):824–836, 2020.

Muñoz, J. A. V., Gonçalves, M. A., Dias, Z., and da Silva Torres, R. Hierarchical clustering-based graphs for large scale approximate nearest neighbor search. *Pattern Recognit.*, 96, 2019.

Pham, N. Simple yet efficient algorithms for maximum inner product search via extreme order statistics. In *KDD*, pp. 1339–1347, 2021.

Pham, N. and Liu, T. Falconn++: A locality-sensitive filtering approach for approximate nearest neighbor search. In *NeurIPS*, pp. 31186–31198, 2022.

Prokhorenkova, L. and Shekhovtsov, A. Graph-based nearest neighbor search: From practice to theory. In *ICML*, pp. 7803–7813, 2020.

Qin, J., Wang, W., Xiao, C., Zhang, Y., and Wang, Y. High-dimensional similarity query processing for data science. In *KDD*, pp. 4062–4063, 2021.

Subramanya, S. J., Devvrit, F., Simhadri, H. V., Krishnaswamy, R., and Kadekodi, R. Rand-nsg: Fast accurate billion-point nearest neighbor search on a single node. In *NeurIPS*, pp. 13748–13758, 2019.

Sun, P., Guo, R., and Kumar, S. Automating nearest neighbor search configuration with constrained optimization. *arXiv preprint arXiv:2301.01702*, 2023.

Xu, X., Wang, M., Wang, Y., and Ma, D. Two-stage routing with optimized guided search and greedy algorithm on proximity graph. *Knowl. Based Syst.*, 229:107305, 2021.

Zilliz. Graph library for approximate similarity search. https://github.com/zilliztech/pyglass, 2023.

A. Frequently Used Notations

Table 5 shows the notations used frequently in this paper.

Table 5. Frequently used notations.

Notation	Explanation
d	Data dimension
$\boldsymbol{q}$	Query
$\boldsymbol{e}$	Edge
ϵ	Error rate
w_{reg}, w_{res}	Two weights associated with edges
m	The number of projected vectors
$\{a_j^i\}, \{b_j\}$	Projected vectors
δ	Threshold of priority queue
L	The number of space partitions
J_{rel}	Indicator of estimation error
$\mathcal{N}_{opt}^{\theta}$	Optimal normal distribution in PEOs test
$\mathcal{N}_H$	Real normal distribution in PEOs test

B. Proofs

B.1. Proof of Lemma 6.2

Proof. We first consider the variance. By $\|\boldsymbol{e}\| = 1$, $\|\boldsymbol{q}\| = 1$ and the Cauchy-Schwarz inequality, we have

$$|\boldsymbol{e}^\top \boldsymbol{q}| \leq \sqrt{\sum_i \|\boldsymbol{q}_i\|^2 \cos^2 \theta_i} \tag{18}$$

which means that $\mathrm{Var}[\mathcal{N}_{\boldsymbol{e},\boldsymbol{q}}^{\theta}] \leq 1 - \cos^2\theta$. Next, we consider $\mathbb{E}[\mathcal{N}_{\boldsymbol{e},\boldsymbol{q}}^{\theta}]$. To get a lower bound of $\sum_i \|\boldsymbol{q}_i\| \cos\theta_i$, we only need to solve the following linear programming problem, that is, to calculate S.

$$\begin{aligned} S &= \min \sum_i s_i \\ s.t. \quad & -\|\boldsymbol{q}_i\| \leq s_i \leq \|\boldsymbol{q}_i\| \qquad (1 \leq i \leq L) \\ & \sum_i s_i \|\boldsymbol{e}_i\| = \cos\theta. \end{aligned} \tag{19}$$

By replacing s_i by $t_i = s_i + \|\boldsymbol{q}_i\|$, we only need to solve the following problem. Noting that all $\|\boldsymbol{q}_i\|$'s can be viewed as constants since $\boldsymbol{q}$ is fixed in the query phase.

$$\begin{aligned} S' &= \min \sum_i t_i \\ s.t. \quad & 0 \leq t_i \leq 2\|\boldsymbol{q}_i\| \quad (1 \leq i \leq L) \\ & \sum_i t_i \|\boldsymbol{e}_i\| = \cos\theta + \sum_i \|\boldsymbol{q}_i\| \|\boldsymbol{e}_i\|. \end{aligned} \tag{20}$$

Let $\|\boldsymbol{q}_{\alpha(1)}\| \geq \|\boldsymbol{q}_{\alpha(2)}\| \cdots \geq \|\boldsymbol{q}_{\alpha(L)}\|$ and $\|\boldsymbol{e}_{\beta(1)}\| \geq \|\boldsymbol{e}_{\beta(2)}\| \cdots \geq \|\boldsymbol{e}_{\beta(L)}\|$, where $\alpha(\cdot)$ and $\beta(\cdot)$ are two permutations of $1, \cdots, L$. Let $U = \cos\theta + \sum_i \|\boldsymbol{q}_i\| \|\boldsymbol{e}_i\|$. Suppose that $1 \leq L' \leq L - 1$ is an integer such that the following relationship holds (noting that $U > 0$ and the leftest term is set to 0 when $L' = 0$).

$$\sum_{1 \leq i \leq L'} \|\boldsymbol{q}_{\alpha(i)}\| \|\boldsymbol{e}_{\beta(i)}\| \leq \frac{U}{2} < \sum_{1 \leq i \leq L'+1} \|\boldsymbol{q}_{\alpha(i)}\| \|\boldsymbol{e}_{\beta(i)}\|. \tag{21}$$

Note that we do not need to consider the case $U/2 \geq \sum_{1 \leq i \leq L} \|\boldsymbol{q}_{\alpha(i)}\| \|\boldsymbol{e}_{\beta(i)}\|$ since it does not occur for any adequate θ. If it occurs, we can get the following inequality by the rearrangement inequality:

$$\cos\theta \geq \sum_i \|\boldsymbol{q}_i\| \|\boldsymbol{e}_i\|. \tag{22}$$

In this case, we definitely know the neighbor u cannot be added into the priority queue.

By (21), we can easily calculate S' in (20) as follows:

$$S' = 2 \sum_{1 \leq i \leq L'} \|\boldsymbol{q}_{\alpha(i)}\| + \frac{U - U'}{\|\boldsymbol{e}_{\beta(L'+1)}\|} \tag{23}$$

where $U' = 2 \sum_{1 \leq i \leq L'} \|\boldsymbol{q}_{\alpha(i)}\| \|\boldsymbol{e}_{\beta(i)}\|$. Then we have the following inequalities:

$$\begin{aligned} S' &\geq 2 \sum_{1 \leq i \leq L'} \|\boldsymbol{q}_{\alpha(i)}\| + \frac{\cos\theta + \sum_i \|\boldsymbol{q}_i\| e_{\min} - U'}{e_{\max}} \\ &\geq \frac{\cos\theta + \sum_i \|\boldsymbol{q}_i\| e_{\min}}{e_{\max}}. \end{aligned} \tag{24}$$

Thus, we have the following result on the expected value:

$$\frac{\cos\theta + (e_{\min} - e_{\max}) \sum_i \|\boldsymbol{q}_i\|}{e_{\max}} \leq S \leq \mathbb{E}[\mathcal{N}_{\boldsymbol{e},\boldsymbol{q}}^{\theta}]/\eta. \tag{25}$$

For the case $\|\boldsymbol{e}_1\| = \cdots = \|\boldsymbol{e}_L\|$, because $\sum_i \|\boldsymbol{q}_i\| \cos\theta_i = \sqrt{L} \cos\theta$ always holds, we can prove the lemma. □

B.2. Proof of Theorem 6.3

Proof. (1) In Sec. 5.2, from the formulation of $A_r(\boldsymbol{e})$, we can see that $A_r(\boldsymbol{e})$ yields a threshold of the angle between $\boldsymbol{e}$ and $\boldsymbol{q}$. Therefore, we only consider the angle between them by assuming $\|\boldsymbol{e}\| = \|\boldsymbol{q}\| = 1$ to simplify the proof.

For the case when $A_r(\boldsymbol{e}) \geq 1$, $dist(\boldsymbol{u}, \boldsymbol{q}) \geq \delta$. Hence $\boldsymbol{u}$ can be safely pruned and the PEOs test returns false. For the case when $A_r(\boldsymbol{e}) \leq 0$, $\boldsymbol{u}$ is likely to be have a distance less than δ from $\boldsymbol{q}$, and the PEOs test always returns true. Therefore, the statement can be proved by showing that the case $0 < A_r(\boldsymbol{e}) < 1$ is $(\delta, 1 - \epsilon)$-routing.

By the result in Lemma 6.2, we have $w_{reg} H_1(\boldsymbol{e}) \sim \mathcal{N}_{reg}$ and $\sqrt{L} w_{res} H_2(\boldsymbol{e}) \sim \mathcal{N}_{res}$, where $\mathcal{N}_{reg}$ and $\mathcal{N}_{res}$ have the following properties:

$$\mathcal{N}(w_{reg}\cos\theta_1\sqrt{2L\ln m}, w_{reg}^2(1-\cos^2\theta_1)) \preceq \mathcal{N}_{reg}, \tag{26}$$

$$\mathcal{N}_{res} = \mathcal{N}(w_{res}\cos\theta_2\sqrt{2L\ln m}, w_{res}^2 L(1-\cos^2\theta_2)) \tag{27}$$

where θ_1 is the angle between $\boldsymbol{e}_{reg}$ and $\boldsymbol{q}$, and θ_2 is the angle between $\boldsymbol{e}_{res}$ and $\boldsymbol{q}$. Note that (26) holds since $e[i] = e_{reg}[i]$ $(1 \le i \le L)$. Suppose that $dist(\boldsymbol{u},\boldsymbol{q}) < \delta$. Then, by the definition of r, the condition that u can pass the PEOs test is equivalent to the following inequality:

$$\cos\theta \ge F_{\boldsymbol{e},\epsilon}(T_r(\boldsymbol{e})) = A_r(\boldsymbol{e}) = \cos\tilde{\theta} > 0. \tag{28}$$

That is, $\tilde{\theta}$ is the threshold angle. By $\|\boldsymbol{e}\| = \|\boldsymbol{q}\| = 1$, we have

$$\cos\theta = \boldsymbol{e}^\top \boldsymbol{q} = w_{reg}\cos\theta_1 + w_{res}\cos\theta_2. \tag{29}$$

Thus, we have the following relationship:

$$\mathcal{N}_{rel} = \mathcal{N}(\cos\theta\sqrt{2L\ln m}, V) \preceq \mathcal{N}_{reg} + \mathcal{N}_{res} \tag{30}$$

where $V = w_{reg}^2(1-\cos^2\theta_1) + w_{res}^2 L(1-\cos^2\theta_2)$. Then, we have the following claim.

Claim: if the following inequality holds:

$$V \le w_{reg}^2 + Lw_{res}^2 - \frac{L\cos^2\theta}{L+1}, \tag{31}$$

then, $\Pr[H(\boldsymbol{e}) \ge T_r(\boldsymbol{e})] \ge 1-\epsilon$.

Proof of Claim: Clearly, $H(\boldsymbol{e})$ can be viewed as a random variable and $H(\boldsymbol{e}) \sim \mathcal{N}_{reg} + \mathcal{N}_{res}$. Then, we have

$$\Pr[H(\boldsymbol{e}) \ge Q_{\mathcal{N}_{reg}+\mathcal{N}_{res}}(\epsilon)] = 1-\epsilon \tag{32}$$

where $Q_{\mathcal{N}}$ denotes the quantile function with respect to the normal distribution $\mathcal{N}$. By (31), we have $\mathcal{N}_{\min}^{\boldsymbol{e},\cos\theta} \prec \mathcal{N}_{rel}$. Then, it can be seen that $Q_{\mathcal{N}_{rel}}(\epsilon) \ge Q_{\mathcal{N}_{\min}^{\boldsymbol{e},\cos\theta}}(\epsilon)$, where $\epsilon \le 0.5$. For two normal distributions $\mathcal{N}(\mu_1,\sigma_1)$ and $\mathcal{N}(\mu_2,\sigma_2)$, when $\epsilon \le 0.5$, $\mu_1 \le \mu_2$ and $\sigma_1 \ge \sigma_2$, i.e., $\mathcal{N}(\mu_1,\sigma_1) \prec \mathcal{N}(\mu_2,\sigma_2)$, we have the following relationship:

$$\mu_1 + \sigma_1\sqrt{2}erf^{-1}(2\epsilon-1) \le \mu_2 + \sigma_2\sqrt{2}erf^{-1}(2\epsilon-1). \tag{33}$$

That is, $Q_{\mathcal{N}(\mu_1,\sigma_1)} \le Q_{\mathcal{N}(\mu_2,\sigma_2)}$. On the other hand, because $\mathcal{N}_{\min}^{\boldsymbol{e},\cos\tilde{\theta}} \preceq \mathcal{N}_{\min}^{\boldsymbol{e},\cos\theta}$, we have

$$\begin{aligned}\Pr[H(\boldsymbol{e}) \ge T_r(e)] &= \Pr[H(\boldsymbol{e}) \ge Q_{\mathcal{N}_{\min}^{\boldsymbol{e},\cos\tilde{\theta}}}(\epsilon)] \\ &\ge \Pr[H(\boldsymbol{e}) \ge Q_{\mathcal{N}_{\min}^{\boldsymbol{e},\cos\theta}}(\epsilon)] \\ &\ge \Pr[H(\boldsymbol{e}) \ge Q_{\mathcal{N}_{rel}}(\epsilon)] \\ &\ge \Pr[H(\boldsymbol{e}) \ge Q_{\mathcal{N}_{reg}+\mathcal{N}_{res}}(\epsilon)] \\ &= 1-\epsilon.\end{aligned} \tag{34}$$

Thus, the claim is proved. Note that the result in inequality (33) is used three times.

Now, the remaining work is to prove the inequality in (31). First, we introduce W such that

$$W = w_{reg}^2\cos^2\theta_1 + Lw_{res}^2\cos^2\theta_2. \tag{35}$$

To get a lower bound of W, we only need to solve the following linear programming problem:

$$\begin{aligned} W_{\min} &= \min\{s_1^2 + Ls_2^2\} \\ s.t.\quad & s_1 + s_2 = \cos\theta \\ & -w_{reg} \le s_1 \le w_{reg} \\ & -w_{res} \le s_2 \le w_{res}. \end{aligned} \tag{36}$$

Because

$$\begin{aligned} s_1^2 + Ls_2^2 &= (L+1)(s_2 - \frac{\cos\theta}{L+1})^2 + \frac{L\cos^2\theta}{L+1} \\ &\ge \frac{L\cos^2\theta}{L+1}, \end{aligned} \tag{37}$$

we prove the first statement.

(2) For the second statement, we first note that

$$\Phi_{\lambda\mathcal{N}_1}(Q_{\lambda\mathcal{N}_2}(\epsilon)) = \Phi_{\mathcal{N}_1}(Q_{\mathcal{N}_2}(\epsilon)) \tag{38}$$

where $\lambda > 0$, $\mathcal{N}_1$ and $\mathcal{N}_2$ are two normal distributions, and Φ is the CDF of normal distribution. Suppose that $dist(\boldsymbol{u},\boldsymbol{q}) \ge \delta$ and $H(\boldsymbol{e})/\sqrt{2L\ln m} \sim \mathcal{N}_H$. By the analysis for the first statement, we have

$$\mathbb{E}[\mathcal{N}_H] = \cos\theta. \tag{39}$$

$$\mathrm{Var}[\mathcal{N}_H] \le \frac{w_{reg}^2 + Lw_{res}^2}{2L\ln m} - \frac{\cos^2\theta}{2(L+1)\ln m}. \tag{40}$$

Under the condition that $\cos\theta \le \tilde{F}_{\tilde{\theta}}^{-1}(\epsilon)$, we can immediately prove the second statement by using a similar result in (33) again.

(3) For the third statement, let $\Delta' = \mathrm{Var}[\mathcal{N}_H] - \mathrm{Var}[\mathcal{N}_{opt}^{\theta}]$ and $\tau = 2L\ln m$, On one hand, we have

$$\tau\Delta' \le (L-1)w_{res}^2 + \frac{\cos^2\theta}{L+1} \le \frac{2L}{(L+1)^2}. \tag{41}$$

On the other hand, we have

$$\begin{aligned} \tau\Delta' &\ge (L-1)w_{res}^2 + \frac{\cos^2\theta}{L+1} - (L+1)(w_{res} + \frac{\cos\theta}{L+1})^2 \\ &\ge -2w_{res}^2 - 2w_{res} \\ &\ge -\frac{2L+4}{(L+1)^2}. \end{aligned} \tag{42}$$

Therefore, the third statement is proved. □

B.3. Proof of Lemma 6.4

Proof. Since $\boldsymbol{e} \sim U(\mathbb{R}^d)$, we can generate $\tilde{\boldsymbol{e}} \sim \mathcal{N}(0, I)$. It can be seen that $\tilde{\boldsymbol{e}}/\|\tilde{\boldsymbol{e}}\| \sim U(\mathbb{S}^{d-1})$. Then we have $\|\boldsymbol{e}_1\|^2 \sim Y/(Y+Z) = Beta(d'/2, (d-d')/2)$, where $Y \sim \mathcal{X}^2(d')$ and $Z \sim \mathcal{X}^2(d-d')$. By the summation of expectations, we have the following result.

$$\mathbb{E}[w_{reg}(L,d)] = \mathbb{E}_X[\sqrt{LX}] \tag{43}$$

where $X \sim Beta(d'/2, (d-d')/2)]$. Let $\alpha = d'/2$ and $\beta = (d-d')/2$. Then we have

$$\begin{aligned}\mathbb{E}[w_{reg}(L,d)] &= \sqrt{L}\int_0^1 \frac{x^{\alpha-\frac{1}{2}}(1-x)^{\beta-1}}{B(\alpha,\beta)}\,dx \\ &= \sqrt{L}\mathbb{E}_{X\sim Beta(\alpha-\frac{1}{2},\beta)}[X]\cdot\frac{B(\alpha-\frac{1}{2},\beta)}{B(\alpha,\beta)} \\ &= \sqrt{L}\cdot\frac{\alpha-\frac{1}{2}}{\alpha-\frac{1}{2}+\beta}\cdot\frac{B(\alpha-\frac{1}{2},\beta)}{B(\alpha,\beta)} \\ &= \frac{\sqrt{L}\Gamma(\frac{d'-1}{2})\Gamma(\frac{d}{2})(d'-1)}{\Gamma(\frac{d'}{2})\Gamma(\frac{d-1}{2})(d-1)} \\ &\geq \frac{(d'-1)\sqrt{2Ld-3L}}{(d-1)\sqrt{2d'+2\sqrt{3}-6}}.\end{aligned} \tag{44}$$

We use the following Gautschi's inequality to get the last inequality in (44).

$$\sqrt{x+\frac{1}{4}} \leq \frac{\Gamma(x+1)}{\Gamma(x+\frac{1}{2})} \leq \sqrt{x+\frac{\sqrt{3}}{2}-\frac{1}{2}}. \tag{45}$$

□

C. Comparison of Routing Tests

To compare the three routing tests in this paper (SimHash, RCEOs, and PEOs), we first introduce the following definition.

Definition C.1. Let θ be the angle of $\boldsymbol{e}$ and $\boldsymbol{q}$ to be estimated, and let $X(\theta)$ and $Y(\theta)$ be two random variables regarding θ, such that (1) $\mathbb{E}[X(\theta)] \in C^1$ and $\mathbb{E}[Y(\theta)] \in C^1$ are two strictly monotone functions of θ, and (2) $\mathrm{Var}[X(\theta)]$ and $\mathrm{Var}[Y(\theta)]$ are continuous. Let $\gamma(\theta) = |(\mathbb{E}[X(\theta)])'|/|(\mathbb{E}[Y(\theta)])'|$, where $|(\mathbb{E}[Y(\theta)])'| > 0$. We say that the estimator associated with Y is better than that associated with X for θ, if $\mathrm{Var}[X(\theta)] > \mathrm{Var}[\gamma(\theta)Y(\theta)]$.

Roughly speaking, for a given θ, Y is considered to be better than X if the distributions associated with X can distinguish the angles around θ more easily than those associated with Y. Based on Definition C.1, we have the following result.

Lemma C.2. *Suppose that SimHash uses n hash functions and RCEOs uses m projected vectors. If $m > \exp(n/[2\theta(\pi-\theta)])$, RCEOs is better than SimHash for θ ($0 < \theta < \pi$).*

Before the proof of Lemma C.2, to explain why we use Definition C.1 to compare the estimators associated with X and Y, we prove the following claim.

Claim: Suppose that $0 < \tilde{\theta}_1 < \tilde{\theta}_2 < \pi$ and Y is better than X for all θ's, where $\tilde{\theta}_1 < \theta < \tilde{\theta}_2$. For an arbitrary θ_1, where $\tilde{\theta}_1 < \theta_1 < \tilde{\theta}_2$, we can find a $\epsilon > 0$ such that for any θ_2, where $\theta_1 - \epsilon \leq \theta_2 \leq \theta_1 + \epsilon$, $\mathrm{Var}[X(\theta_1)] > \mathrm{Var}[\gamma Y(\theta_1)]$ and $\mathrm{Var}[X(\theta_2)] > \mathrm{Var}[\gamma Y(\theta_2)]$, where $\gamma = |\mathbb{E}[X(\theta_1)] - \mathbb{E}[X(\theta_2)]|/|\mathbb{E}[Y(\theta_1)] - \mathbb{E}[Y(\theta_2)]|$.

Proof of the claim: this result can be immediately obtained by the continuity of $\mathrm{Var}[X(\theta)]$, $\mathrm{Var}[Y(\theta)]$, $\gamma(\theta) = |(\mathbb{E}[X(\theta)])'|/|(\mathbb{E}[Y(\theta)])'|$ and the mean value theorem.

This claim shows that, under the conditions in the claim, for two close angles θ_1 and θ_2, the variance associated with Y is smaller than that associated with X when the difference of expected values are the same (after scaling).

Proof. By symmetry, we only need to consider the case $0 < \theta \leq \pi/2$. To make the ranges of θ and $\cos\theta$ the same, we consider $2(n - \#Col)/n$ in SimHash and $f(m)\boldsymbol{q}^\top\boldsymbol{a}_1$ in RCEOs, where $\#Col$ denotes the collision number of the pair $(\boldsymbol{e}, \boldsymbol{q})$ in SimHash and $f(m) = 1/\sqrt{2\ln m}$. By the lemma of SimHash (Lemma 4.1), we have

$$X(\theta) = 2(n - \#Col)/n \sim \frac{2}{n} \times B(n, \frac{\theta}{\pi}). \tag{46}$$

By the lemma of RCEOs (Lemma 4.2), we have

$$Y(\theta) = f(m)\boldsymbol{q}^\top\boldsymbol{a}_1 \sim N(\cos\theta, f^2(m)\sin^2\theta) \tag{47}$$

where $X(\theta)$ and $Y(\theta)$ are two random variables depending on θ. It is easy to verify that all the regularity conditions in Definition C.1 are satisfied. Since both of the ranges of $2\theta/\pi$ and $\cos\theta$ are $[0,1]$, and they are two monotone functions of θ, it is feasible to compare them by the criterion in Definition C.1. First, it is easy to see that $\gamma(\theta) = 2/(\pi sin\theta)$. On the other hand, we have

$$\mathrm{Var}[X(\theta)] = 4\theta(\pi-\theta)/(n\pi^2). \tag{48}$$

$$\mathrm{Var}[\gamma(\theta)Y(\theta)] = 2/(\pi^2 \ln m). \tag{49}$$

Then, we can see that

$$m > \exp(n/[2\theta(\pi-\theta)]) \Leftrightarrow \mathrm{Var[X(\theta)]/Var[\gamma(\theta)Y(\theta)]} > 1. \tag{50}$$

□

Next, we consider PEOs (opt) and PEOs, where w_{res} is assumed to be 0 in PEOs (opt). For PEOs (opt), the distribution associated with $\cos\theta$ can be approximated by $\mathcal{N}^\theta_{opt}$, and for

Table 6. Comparison of routing tests when $d = 384$ and $\theta = \pi/2$. The baseline is SimHash ($n = 64$). We show under which conditions, RCEOs, PEOs (opt), and PEOs can approximately outperform SimHash.

Routing Test	#Projected Vectors	L	Code Length (Byte)
SimHash	$n = 64$	NA	8
RCEOs	$m > 428957$	$L = 1$	4
PEOs (opt)	$m = 128$	$L > 2.67$	4 (3+1)
PEOs	$m = 128$	$L > 2.69$	4 (3+1)

PEOs, the distribution associated with $\cos\theta$ can be approximated by $\mathcal{N}_{act}^{\theta}$, which is defined as follows:

$$\mathcal{N}_{opt}^{\theta} \sim \mathcal{N}(\cos\theta, \frac{\sin^2\theta}{2L\ln m}), \tag{51}$$

$$\mathcal{N}_{act}^{\theta} \sim \mathcal{N}(\cos\theta, \frac{\sin^2\theta(w_{reg}^2 + Lw_{res}^2)}{2L\ln m}). \tag{52}$$

That is, W in (35) is approximated by $(w_{reg}^2 + Lw_{res}^2)\cos^2\theta$. According to the previous analysis, we can see that, to make them outperform SimHash, m and L should satisfy the following requirements, where m is sufficiently large.

$$\textbf{PEOs(opt):} \quad L\ln m > n/[2\theta(\pi - \theta)]. \tag{53}$$

$$\textbf{PEOs:} \quad L\ln m > (w_{reg}^2 + Lw_{res}^2))n/[2\theta(\pi - \theta)]. \tag{54}$$

Based on the results above, we can approximately compare different tests by a concrete example. Let $d = 384$ and $\theta = \pi/2$. Suppose that SimHash ($n = 64$) is regarded as the baseline. For RCEOs, PEOs (opt) and PEOs, we check how many projected vectors are needed to make them outperform SimHash, as shown in Table 6. Here, w_{res} is taken as the expected value under the isotropic distribution.

D. Technical Extensions

D.1. Dimension Permutation

We describe the details of the optional dimension permutation in PEOs. Only in this section, we use e_j to denote the j-th coordinate of $\boldsymbol{e}$, instead of the projection on the j-th subspace. We introduce the following notation.

$$\text{Avg}(e_j) = \sum_{e \in E} (e_j)^2/|E| \tag{55}$$

where E denotes the edge set of the graph index. For the dimension permutation in PEOs, we use the following greedy algorithm to permute the coordinates of data vectors, divided into four steps:

(1) We generate L empty sets, denoted by $S_1, S_2, \cdots, S_L$, each representing a subspace.

(2) We calculate $\text{Avg}(e_j)$ for every dimension j and sort the dimensions in the ascending order of $\text{Avg}(e_j)$.

(3) We execute the allocation procedure by $d' = d/L$ rounds. In the l-th round, for each $1 \le j \le L$, we allocate dimension $(l-1)L + j$, to set S_i, where S_i has the greatest $\sum_{k \in S} \text{Avg}(e_k)$ among all the sets that have not been added any dimension in this round.

(4) After the allocation in Step 3, we permute the coordinates of all $\boldsymbol{e}$'s such that coordinate e_j appears in the i-th sub-vector of $\boldsymbol{e}$ if j is in S_i.

When dimension permutation is finished, we store the permutation, in order to permute the coordinates of the query for search.

D.2. Extension to MIPS

To support MIPS, we only need to use the following $A_r(\boldsymbol{e})$ to replace the one in (9):

$$A_r(\boldsymbol{e}) = \frac{\boldsymbol{p}^\top \boldsymbol{q} - \boldsymbol{v}^\top \boldsymbol{q}}{\|\boldsymbol{q}\|\|\boldsymbol{e}\|} \tag{56}$$

where $\boldsymbol{p}$ denotes the element in the temporary result list having the smallest inner product with $\boldsymbol{q}$. The remaining procedure is completely same as that of ℓ_2 metric.

E. Experimental Setup

All the experiments were performed on a PC with Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz. All the compared methods were implemented in C++, with 64 threads for indexing and a single CPU for searching, following the standard setup in ANN-Benchmarks (Bernhardsson, 2024).

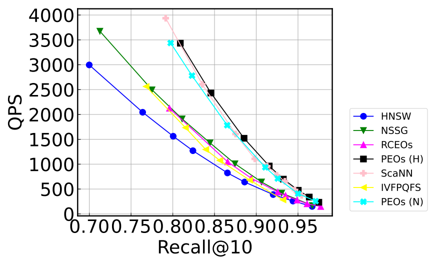
We use the following parameter setup for the competitors:

(1) **HNSW.** $M = 32$, $efc = 2000$ for the two GloVe datasets and $efc = 1000$ for the other datasets. We also use this setting for the HNSW index of FINGER, RCEOs and PEOs.

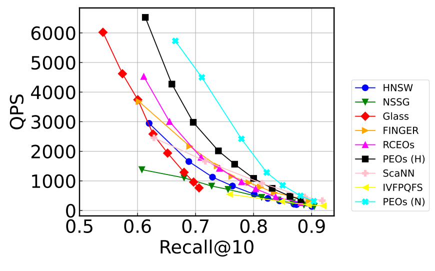
(2) **NSSG.** (L, R, C) is set to (100, 50, 60) on SIFT10M, (500, 70, 60) on GIST, and (500, 60, 60) on the other datasets, The parameter K in the prepared KNN-graph is 400.

(3) **Glass.** For DEEP10M, we use Glass (+HNSW) since Glass (+NSG) failed to finish the index construction. For the other datasets, we use Glass (+NSG) since it works better than Glass (+HNSW) especially for high recall rates. R is 32, L is set to an experimentally optimal value in $[200, 2000]$.

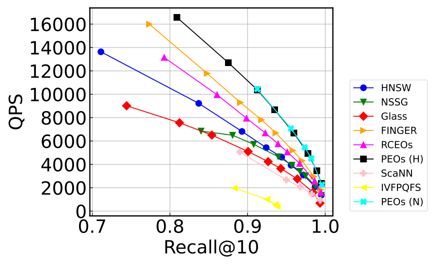
(4) **FINGER (+HNSW).** All the parameters are set to the recommended values in its source code. In particular, the dimension of subspace is set to 64.



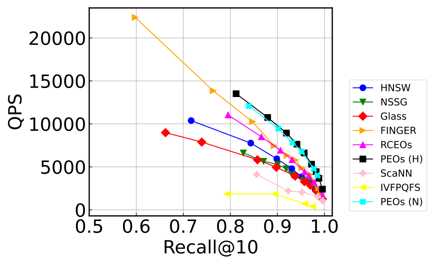
(a) GloVe200-angular, $K = 10$



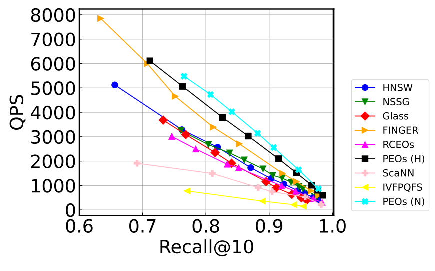
(b) GloVe300-ℓ_2, $K = 10$



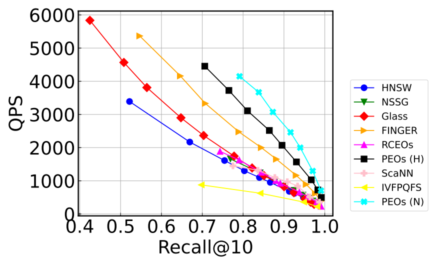
(c) DEEP10M-angular, $K = 10$



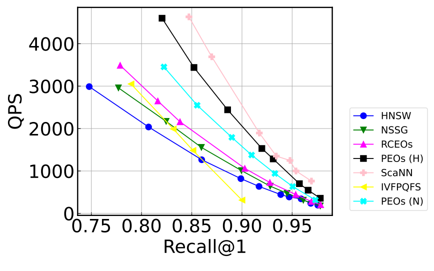
(d) SIFT10M-ℓ_2, $K = 10$



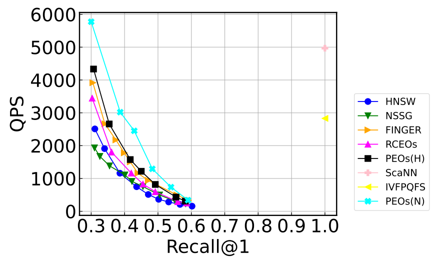
(e) Tiny5M-ℓ_2, $K = 10$



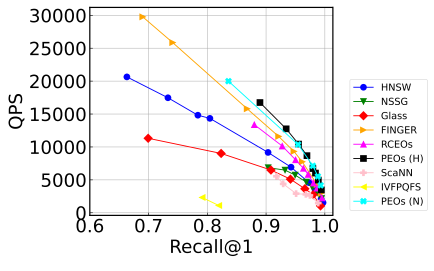
(f) GIST-ℓ_2, $K = 10$



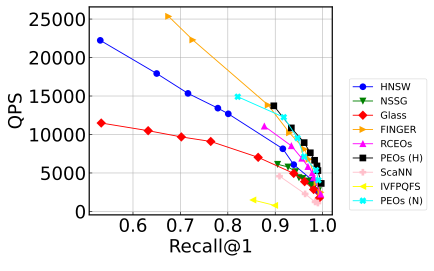
(g) GloVe200-angular, $K = 1$



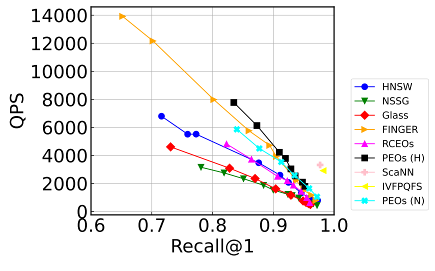
(h) GloVe300-ℓ_2, $K = 1$



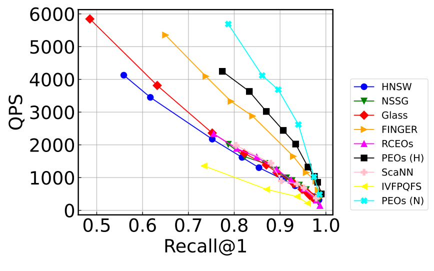
(i) DEEP10M-angular, $K = 1$



(j) SIFT10M-ℓ_2, $K = 1$



(k) Tiny5M-ℓ_2, $K = 1$



(l) GIST-ℓ_2, $K = 1$

Figure 7. Recall-QPS evaluation, $K = 10$ and $K = 1$. PEOs (H) denotes HNSW+PEOs and PEOs (N) denotes NSSG+PEOs.

(5) **RCEOs (+HNSW).** The only difference with PEOs is $L = 1$ in RCEOs.

(6) **PEOs (+HNSW).** Based on the analysis in Sec. 6.2, L is set to 8, 8, 10, 15, 16, 20 on the six datasets sorted by ascending order of dimension. ϵ is set to 0.2 and $m = 128$ such that every vector ID can be encoded by one byte.

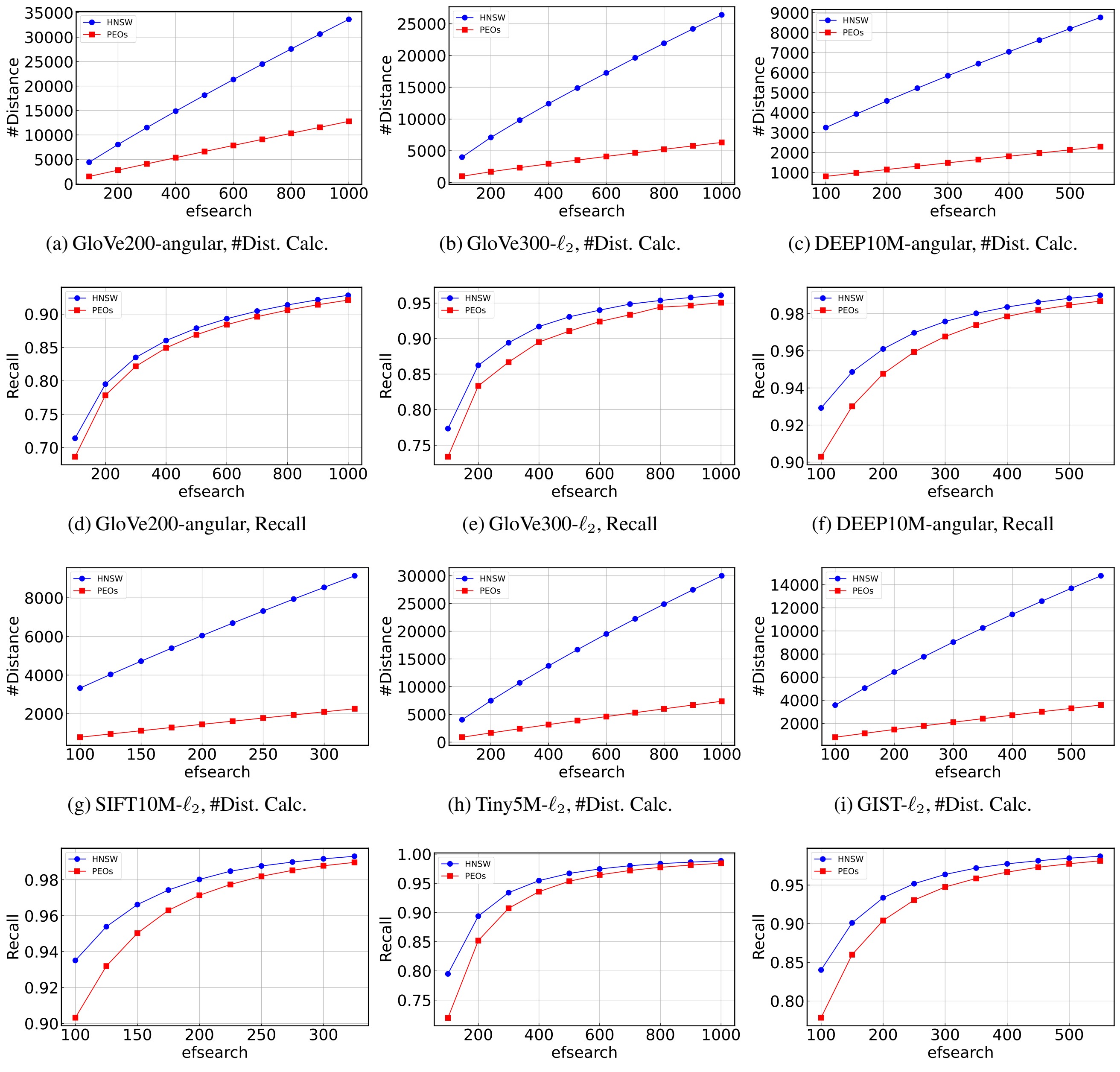
(7) **PEOs (+NSSG).** The settings of L are the same as those in PEOs (+HNSW). The NSSG parameters are set to the same values as in the vanilla NSSG.

(8) **ScaNN.** Dimensions_per_block is set to 2 on SIFT10M, DEEP10M, and GloVe200, and 4 on the other ones. The other user-specified parameters are adjusted to achieve the trade-off curves.

F. Additional Experiments

F.1. Effect of Result Number K

In Figure 7, we show the recall-QPS comparison under $K = 10$ and $K = 1$. We have the following observations.



(a) GloVe200-angular, #Dist. Calc.

(b) GloVe300-ℓ_2, #Dist. Calc.

(c) DEEP10M-angular, #Dist. Calc.

(d) GloVe200-angular, Recall

(e) GloVe300-ℓ_2, Recall

(f) DEEP10M-angular, Recall

(g) SIFT10M-ℓ_2, #Dist. Calc.

(h) Tiny5M-ℓ_2, #Dist. Calc.

(i) GIST-ℓ_2, #Dist. Calc.

(j) SIFT10M-ℓ_2, Recall

(k) Tiny5M-ℓ_2, Recall

(l) GIST-ℓ_2, Recall

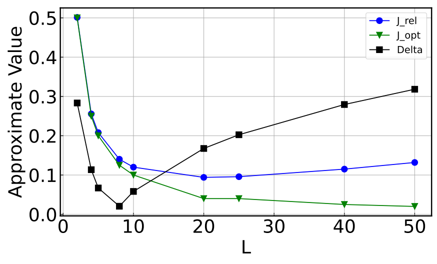
Figure 8. Evaluation of efs-number of distance calculations and efs-recall, $K = 100$.

(1) PEOs still performs the best on all the datasets, showcasing the robustness of PEOs for different values of K. In particular, the performance improvement of PEOs over HNSW under a small K is almost consistent with that under $K = 100$. (2) The improvement of PEOs over FINGER is marginal when $K = 1$. This is because the search under $K = 1$ is much easier than the search under a large K value. When K grows, we have to accordingly increase the size of the result list, under which situation, the routing becomes harder and a more accurate estimation is important for the performance improvement.

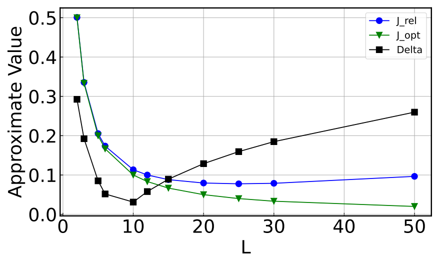
F.2. Effect of List Size efs

To evaluate the effectiveness of PEOs in reducing exact distance calculations, we also plot the efs-number of distance calculations and efs-recall curves, where efs is the size of the temporary result list, adjusted to achieve a different trade-off between efficiency and accuracy. From the results in Figure 8, we have the following observations.

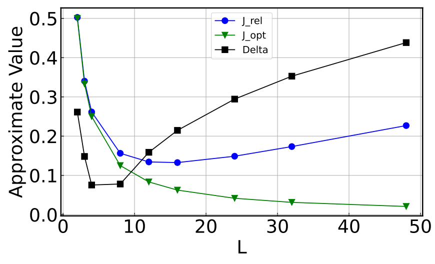
(1) PEOs saves around 75% exact distance calculations on each dataset, and this is the main reason for the improvement on QPS. On the other hand, such improvement is not sensitive



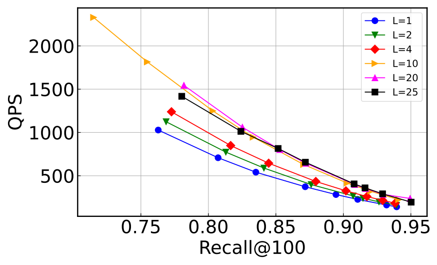
(a) Dimension: 200 (GloVe200)



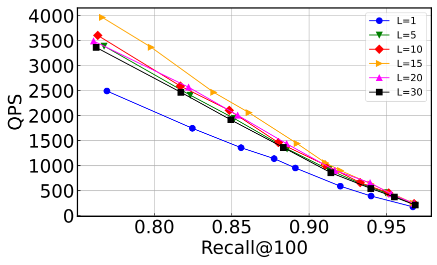
(b) Dimension: 300 (GloVe300)



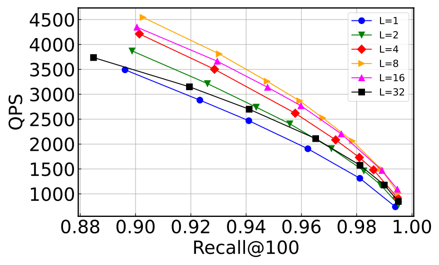
(c) Dimension: 96 (DEEP10M)



(d) GloVe200-angular, $d = 200$

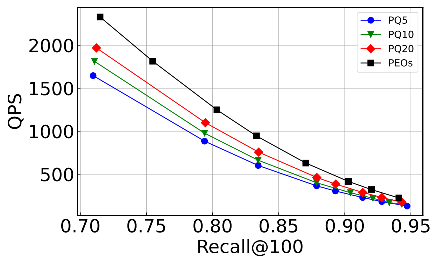


(e) GloVe300-ℓ_2, $d = 300$

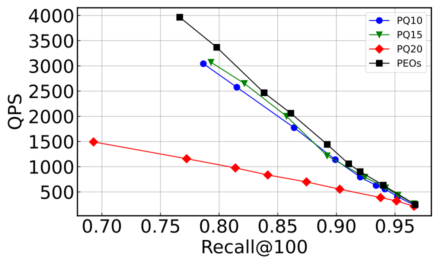


(f) DEEP10M-angular, $d = 96$

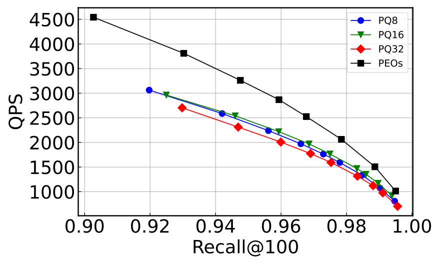
Figure 9. Effect of L on GloVe200, GloVe300, and DEEP10M.



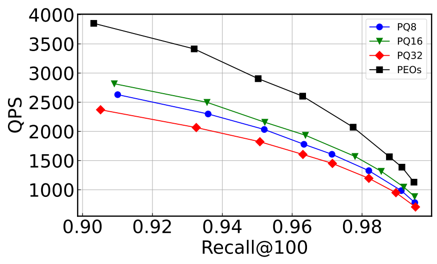
(a) GloVe200-angular, $K = 100$



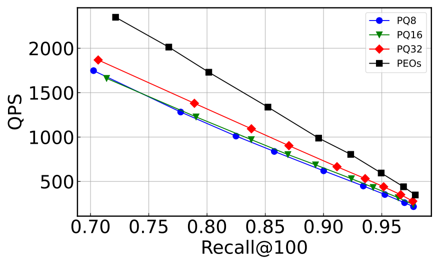
(b) GloVe300-ℓ_2, $K = 100$



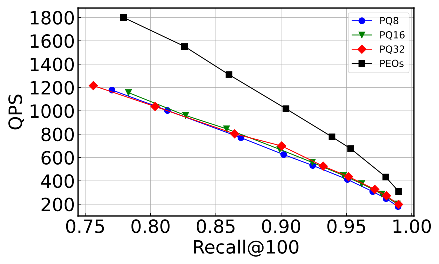
(c) DEEP10M-angular, $K = 100$



(d) SIFT10M-ℓ_2, $K = 100$



(e) Tiny5M-ℓ_2, $K = 100$



(f) GIST-ℓ_2, $K = 100$

Figure 10. Comparison with PQ-based routing. X in PQX denotes the number of sub-codebooks.

to the value of efs. (2) Due to the existence of an estimation error, for the same efs, the recall of PEOs is smaller than that of HNSW. Nonetheless, the difference is quite small, especially when $efs > 100$ (note that $efs \geq 100$ when $K = 100$), thanks to the theoretical guarantee of PEOs. (3) As efs grows, the difference between the recalls of HNSW and PEOs are very small while the saved distance calculations by PEOs are still large. This explains why PEOs works better for larger efs values, which generally corresponds to larger K.

F.3. Effect of Space Partition Size L on Other Datasets

In addition to the evaluation in Sec. 7.3, we show the effect of L on the other three real datasets in Figure 9. We can see that the performances on these three real datasets are still consistent with our theoretical analysis and the advantage of $L > 1$ over $L = 1$ remains.

F.4. Comparison with Product Quantization (PQ)

We can see that PEOs has similarities with PQ (Jégou et al., 2011), both partitioning the original space into orthogonal subspaces and combining the information in different subspaces. Thus, an interesting question is if the quantization techniques can be used for the acceleration of routing. First, we note that, since $|E|$ is much larger than the data size $|\mathcal{O}|$, and the local intrinsic dimensionality (LID) of $\boldsymbol{e}$'s is also very large due to the edge-selection strategy, the quantization of $\boldsymbol{e}$'s is not as effective as the quantization of raw vectors. On the other hand, apart from the probability guarantee, PEOs has the following two advantages. (1) The impact of w_{res} is fully considered in PEOs while the impact of individual quantization error is hard to be measured. (2) PEOs applies a non-linear transformation, i.e., $F^{-1}_{\boldsymbol{e},\epsilon}$, to the threshold $\cos\theta$. After such transformation, as threshold θ decreases from $\pi/2$ to 0, $T_r(\boldsymbol{e})$ will grow rapidly such that passing the PEOs test becomes much harder, because the possibility that a small angle θ exists between two high-dimensional vectors is very small. That is, PEOs takes the potential impact of threshold into consideration thanks to the probability guarantee. On the other hand, quantization-based techniques focus on the minimization of quantization errors and do not consider the impact of the threshold.

For an empirical evaluation, first, we need to adapt the existing quantization-based technique for routing. Specifically, we use norm-explicit product quantization (Dai et al., 2020) – which has shown competitive performance for the estimation of inner product – to quantize $\boldsymbol{e}$'s and calculate the approximate inner product between $\boldsymbol{q}$ and each $\boldsymbol{e}$. Since the standard quantization generally does not consider the effect of quantization error in the search phase, to obtain better search performance, we also maintain the norm of the residual part of $\boldsymbol{e}$ after quantization, denoted by $\|\boldsymbol{e}\|_{res}$. Then we need to introduce a coefficient c and write the test of quantization as follows:

$$c\|\boldsymbol{e}\|_{res}\|\boldsymbol{q}\| \geq I \tag{57}$$

where I denotes the threshold of inner product for being added into the temporary result list, which is calculated in a similar way to $A_r(\boldsymbol{e})$. In practice, although we do not know the optimal value of c for each dataset, we experimentally adjust it in $(0, 1)$ to get a near-optimal value. By the above setting, we can compare the performances of HNSW+PEOs and HNSW+PQ. From the results in Figure 10, we have the following three observations. (1) Expect for GloVe300, PEOs obviously performs better than PQ, as analyzed in the previous discussion. (2) Due to the hardness of residual quantization, the quantization error may be quite large especially for the high-dimensional datasets, such as GIST, which makes the estimation inaccurate. (3) The impact of the number of sub-codebooks is hard to be predicted, as shown on GloVe300.