RESEARCH-ARTICLE

Accelerating High-Dimensional ANN Search via Skipping Redundant Distance Computations

ZIWEN SONG, Northeastern University, Shenyang, Liaoning, China

BIN WANG, Northeastern University, Shenyang, Liaoning, China

XIAOCHUN YANG, Northeastern University, Shenyang, Liaoning, China

Proceedings of the ACM on Management of Data, Volume 3, Issue 6 (December 2025)
https://doi.org/10.1145/3769753
EISSN: 2836-6573

Accelerating High-Dimensional ANN Search via Skipping Redundant Distance Computations

ZIWEN SONG, Northeastern University, China
BIN WANG*, Northeastern University, China
XIAOCHUN YANG, Northeastern University, China

Graph-based methods for high-dimensional Approximate Nearest Neighbor (ANN) search have achieved remarkable success. Recent studies have revealed that DCO (distance comparison operation) is a bottleneck in graph-based methods due to distance computations. Optimizations such as ADSampling, DDC and DADE are employed to alleviate this issue by terminating the distance computation early. Although these optimizations achieve significant speedup, they rely on an estimation-and-testing process for early termination. Their effectiveness diminishes when SIMD (Single Instruction Multiple Data) acceleration is enabled in distance computation, the cost introduced in estimation-and-testing process may outweigh the benefit of early termination, leading to performance degradation. Furthermore, the best-first-search strategy employed in graph-based methods requires DCO for all neighboring points, incurring redundant distance computation. To address these issues, in this paper, we first perform a cost analysis to reveal the inefficiency of existing DCO optimizations under SIMD-enabled setting. We then analyze the current search strategy to demonstrate that not all neighboring nodes require DCO. Based on these analyses, we present SkipComputing to accelerate the ANN search. Specifically, we first propose a subspace-based candidate search strategy to identify promising points for DCO, thereby eliminating the reliance on estimation-and-testing based DCO optimization for low potential points. We then develop a lower-bound pruning method based on distance decomposition to enable early termination of distance computations in DCO. Finally, we optimize the data layout to reduce the overhead of random memory access during candidate search. When SIMD is enabled, experimental results show that SkipComputing substantially outperforms HNSW, achieving up to 6x performance improvement. Furthermore, it achieves up to 2.7x speedup over state-of-the-art optimization methods while maintaining competitive space efficiency.

CCS Concepts: • **Information systems** → **Database query processing**; **Nearest-neighbor search**; **Top-k retrieval in databases**.

Additional Key Words and Phrases: Approximate Nearest Neighbor Search; Distance Comparison Operation; Graph-based Method; Vector Search; High-Dimensional Data

ACM Reference Format:
Ziwen Song, Bin Wang, and Xiaochun Yang. 2025. Accelerating High-Dimensional ANN Search via Skipping Redundant Distance Computations. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 288 (December 2025), 29 pages. https://doi.org/10.1145/3769753

*Corresponding author

Authors' Contact Information: Ziwen Song, qingyunyimo@outlook.com, Northeastern University, Shenyang, China; Bin Wang, binwang@mail.neu.edu.cn, Northeastern University, Shenyang, China; Xiaochun Yang, yangxc@mail.neu.edu.cn, Northeastern University, Shenyang, China.

ACM 2836-6573/2025/12-ART288
https://doi.org/10.1145/3769753

1 Introduction

Nearest neighbor search (NNS) in high-dimensional space [37] is a fundamental and challenging problem with a wide range of applications in fields such as data mining [58], recommendation systems [20], and information retrieval [10, 73, 78, 79]. With the continuous advancement of artificial intelligence, especially with the emergence of large language models [6, 14], vector databases and vector search have gained a lot of attention [53, 54, 61, 80], highlighting the increasing importance of NNS [67, 74]. Due to the curse of dimensionality [37, 72], it is challenging to find the exact nearest neighbors through an index without scanning the entire dataset. For many real-world applications, it is not critical to obtain exact search results. Researchers resort to the approximate nearest neighbor (ANN) search to tolerate small errors in exchange for high search performance. Many methods have been proposed, including quantization-based, hash-based, and graph-based methods [47] to improve search performance. Among them, graph-based methods [51, 57] such as HNSW [51] have attracted wide attention due to their efficiency and effectiveness in handling the ANN search.

In general, the search process can be divided into **candidate generation** and **verification** stages [75]. The candidate generation stage is responsible for generating a set of potential points which are likely to be k-nearest neighbors of the query point, while the verification stage further identifies k-nearest neighbors from candidate points. The graph-based methods rely on graph traversal to perform search operations. The graph search process involves iteratively examining neighboring points of verified points (candidate generation) and updating the result queue (verification) until the termination condition is satisfied. In this procedure, the distances between candidate points and the query are calculated to determine whether they satisfy the given threshold condition and return the exact distance if so, this fundamental step is formally called the distance comparison operation (DCO) [29]. DCOs constitute a substantial portion of the computational overhead due to its distance computation. Previous studies [16, 24, 29, 49, 75] have proposed various estimation methods to reduce distance computations in DCO.

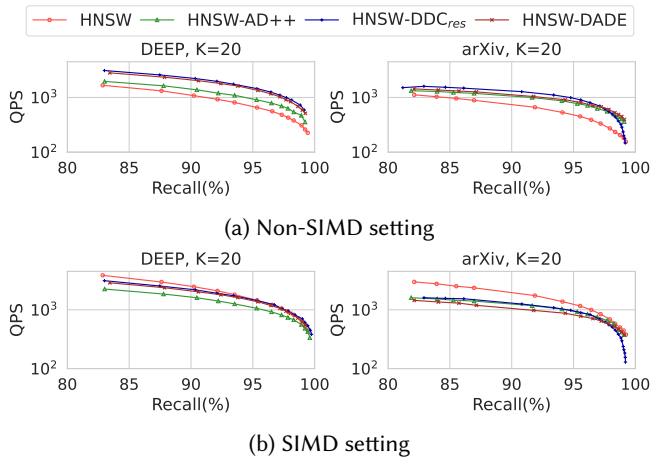


Fig. 1. Search performance degradation observed. When using SIMD for distance calculations, early termination strategy like ADSampling, DDC, and DADE struggles to improve performance and can cause performance degradation compared to full distance computation (SCAN) in HNSW.

ADSampling [29], DDC [75] and DADE [24] employ estimation-and-testing process to terminate distance computations early, thus reducing computational overhead in HNSW, as illustrated in

Figure 1a. However, these methods disable SIMD during distance computation in their experiments. SIMD, a common feature in modern CPUs, enables parallel processing of multiple data elements, and enabling it leads to substantial improvements in search performance for HNSW. However, when SIMD is enabled, ADSampling, DDC and DADE show limited effectiveness and experience serious performance degradation. As illustrated in Figure 1b, performance degradation is observed after applying ADSampling, DDC or DADE in HNSW. While in non-SIMD setting, they can significantly improve search performance. The primary reason for this result is that, in the SIMD setting, the additional costs introduced by these methods can negate the benefits of early termination, leading to potential performance degradation. We will provide a detailed analysis in Section 3.1.

Distance computation optimization under SIMD context poses stringent challenges. Specifically, we face two extreme challenges that must be addressed simultaneously to achieve performance optimization. First, we should minimize the additional costs as much as possible to avoid performance degradation. Current DCO optimization methods introduce substantial additional costs that limit their adaptability within the SIMD context. Second, we must skip as many redundant computations as possible to achieve an effective performance improvement. Current DCO optimization methods inadequately reduce computational costs to compensate for their additional overhead, yielding marginal performance gains or even degradation. To address these challenges, we develop our strategy from the following three aspects.

Firstly, DCO is not required for all candidate points, only necessary for those with high potentials. DCO can be considered as a step within the verification stage, which focuses on verifying candidate points. In graph-based methods, each node maintains a set of approximate nearest neighbors. As the search progresses, the algorithm converges toward the true neighborhood of the query [57], where neighboring nodes are increasingly likely to contain true nearest neighbors (*"a neighbor of a neighbor is also likely to be a neighbor"*) [25]. Simultaneously, low-confidence candidate points are gradually replaced by high-confidence candidate points as the search advances. By avoiding DCO for these low-confidence candidates, we can skip redundant distance computations and avoid estimation-and-testing based DCO optimization for these points to improve the search performance.

Secondly, estimation-and-testing is not the only strategy to reduce the cost of a single DCO. Lower bound pruning offers an alternative: given a lower bound $LB(\mathbf{q}, \mathbf{p})$ of the distance between the query $\mathbf{q}$ and the point $\mathbf{p}$, and a threshold τ, if $LB(\mathbf{q}, \mathbf{p}) > \tau$, $\mathbf{p}$ can be immediately discarded without an exact distance computation. This method simplified the filtering to a simple comparison while avoiding false negatives, thus significantly reducing the costs. However, two critical challenges arise: (1) weak lower bounds may prune too few points and (2) the computation of lower bounds itself may become expensive, potentially degrading performance. To address these challenges, we use an iterative refinement method adapted from the ADSampling estimation scheme. Meanwhile, we decompose the distance into multiple parts based on the property of distance decomposition to iteratively calculate the distance. We also transform the data to enable the computation of tighter lower bounds with fewer iterations.

Thirdly, not only the computation cost needs to be optimized. Graph-based methods suffer from cache-inefficient access patterns, making memory access a critical bottleneck [19, 68]. PEOs [49] employ probabilistic routing to filter neighbors, introducing computation cost, but achieving a significant performance improvement. This is accomplished by strategically leveraging data locality to mitigate memory access bottlenecks, although at the cost of increased space usage. This insight motivates a critical direction for our work: we further improve the search performance by reducing random memory accesses during neighbor traversal. By reorganizing data to exploit spatial locality, we improve cache prefetching during neighbor traversal. Furthermore, we design

integration strategies to ensure compatibility with existing quantization methods, jointly optimizing memory access and computation costs.

In this paper, we present a search method SkipComputing to improve the performance of graph-based method through three key innovations. First, we propose an effective candidate selection mechanism that reduces the number of points participating in DCO. Using projection-based methods, we map data points into a lower-dimensional space and conduct candidate search in this space to identify promising candidates to conduct DCO, thus avoiding DCO for low potential points and skipping redundant distance computations. Second, we design a lower bound pruning method to reduce the computational cost of DCO. Leveraging the property that the square Euclidean distance is the accumulation of distance across individual dimensions, we decompose the distance into different components by transforming the data points. Most of the distance is captured by a few of these components, enabling adaptive lower-bound pruning to refine the lower bound and eliminating redundant computations effectively. Third, we optimize the data layout to use cache prefetching to reduce memory access latency during neighboring points access. In addition, we integrate quantization techniques to reduce the memory footprint incurred by optimizing the data layout. The proposed method can be used in various graph-based methods such as NSG [27], Vamana [39] and HNSW [51] to enhance the performance of the multiple graph-based methods. In this paper, we use HNSW.

Our main contributions are summarized as follows:

- We perform a cost analysis to analyze the performance overhead of current DCO optimization methods. We further analyze the redundant evaluation of neighbor nodes in current search algorithm.
- We propose SkipComputing (refer to alg. 5) to skip redundant distance computations by leveraging dimension-reduced data to conduct candidate search to reduce the number of points participating in DCO, and employing adaptive lower bound pruning to terminate distance computations earlier in DCO.
- We optimize memory access efficiency through data layout optimization, enhancing cache prefetching capabilities. We further employ state-of-the-art quantization methods to jointly optimize both storage requirements and computational costs.
- We provide flexible options to meet different requirements in terms of search performance and space consumption trade-off. Experiments demonstrate the effectiveness of our method, which achieves considerably performance improvement over existing methods.

2 Preliminaries

This section outlines the core concepts and definitions used in this paper. We first introduce basic mathematical notations. Vectors, matrices, and points are denoted in boldface. Let P be the dataset with N data points in $\mathbb{R}^D$. Here, $\mathbf{q} \in \mathbb{R}^D$ is a query point. For a point $\mathbf{p} \in P$, we use $\delta(\mathbf{p}, \mathbf{q}) = \sum_i^D (p_i - q_i)^2 = \|\mathbf{p} - \mathbf{q}\|_2^2$ to denote the square of Euclidean distance between two points. and $\delta_d(\mathbf{p}, \mathbf{q})$ denotes the square of distance between two points in the first d dimensions and $\mathbf{q}_{[0:d]}$ the first d dimensions of $\mathbf{q}$. We denote p_i as the ith dimension of a point $\mathbf{p}$. Additional symbols will be defined as they first appear.

2.1 Problem Setting

Definition 2.1 (k-Nearest Neighbor Search). Given a set of points P with each points $\mathbf{p} \in \mathbb{R}^D$, a query point $\mathbf{q} \in \mathbb{R}^D$, and distance function $\delta(\mathbf{p}, \mathbf{q})$, find a set $K \subset P$ with k points, where for any point $\mathbf{p} \in D \setminus K$, $\delta(\mathbf{p}, \mathbf{q}) \geq \max_{\mathbf{p} \in K} \delta(\mathbf{p}, \mathbf{q})$.

Approximate nearest neighbor search allows for retrieval of approximate, rather than exact nearest neighbors. In practice, we do not specify constant error as discussed in [27]. We use recall to measure the accuracy of the result [69], which definition is as:

$$recall(R_o) = \frac{|R_o \cap R_k|}{|R_k|} \tag{1}$$

where$|\cdot|$ denotes the number of elements in a set, R_o denotes the search result, and R_k is the correct k nearest neighbors.

2.2 Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma (JL lemma) [42] is a foundational result in dimensionality reduction that guarantees the approximate preservation of pairwise distances between points when projected into a lower-dimensional space. The proof of the JL lemma rely on the distributional Johnson-Lindenstrauss lemma [45], which shows in following.

Lemma 2.2 (distributional JL lemma [42, 45]). *Let $\epsilon, \beta \in (0,1)$, there exists a probability distribution F over linear functions $f: \mathbb{R}^D \to \mathbb{R}^d$, where $d = \Theta\left(\epsilon^{-2} \log \frac{1}{\delta}\right)$, such that for any fixed $\mathbf{x} \in \mathbb{R}^D$,*

$$\Pr_{f \sim F}\left[\left|\|f(\mathbf{x})\|_2^2 - \|\mathbf{x}\|_2^2\right| \leq \epsilon \|\mathbf{x}\|_2^2\right] \geq 1 - \beta. \tag{2}$$

Let S be a random d-dimensional subspace, let $\mathbf{x}'_\mathbf{i}$ be the projection of $\mathbf{x_i} \in P$ into S, we set $\rho = \|\mathbf{x}'_\mathbf{i} - \mathbf{x}'_\mathbf{j}\|_2^2$ to be the square l_2 norm, and $\mu = (d/D)\|\mathbf{x_i} - \mathbf{x_j}\|_2^2$, then we have the following inequality [21]:

$$\Pr\left[\rho \geq (1+\epsilon)\mu\right] \leq \exp\left(-\frac{d(\epsilon^2/2 - \epsilon^3/3)}{2}\right). \tag{3}$$

The distributional JL lemma provides probabilistic bounds on the distortion of distances between points when data are projected into a lower-dimensional space. This theoretical foundation guides our approach to dimensionality reduction in candidate search of Section 4, where it helps us design the basic search method.

2.3 MS-distance

The MS-distance [36] is a measure of distance between two vectors in the Euclidean space, transformed using their mean and standard deviation. M stands for mean and S stands for standard deviation. The MS-distance aims to use mean and standard deviation to approximately represent the overall information of a vector. It provides lower and upper bounds for the Euclidean distance, with the lower bound form being the focus of this work.

Definition 2.3 (MS-transformation [36]). Given a set of points P in Euclidean space, the MS-transformation maps every point $\mathbf{x} = [x_1, x_2, \ldots, x_D] \in P$ to $\mathbf{x}' = [\mu_x, \sigma_x, x'_1, x'_2, \ldots, x'_D]$ in transformed space where $\mu_\mathbf{x} = \frac{1}{D}\sum_{i=1}^{D} x_i$, and $\sigma_\mathbf{x}^2 = \frac{1}{D}\sum_{i=1}^{D}(x_i - \mu_x)^2$. For each dimension value x_i of $\mathbf{x}$, we map it to $x'_i = \frac{x_i - \mu_\mathbf{x}}{\sigma_\mathbf{x}}$.

The MS-transformation produces a $(D+2)$-dimensional vector $\mathbf{x}'$. Next we define the MS-distance in lower bound form.

Definition 2.4 (MS-distance [36]). Let $\mathbf{x}'$ and $\mathbf{y}'$ be two points in $\mathbb{R}^{D+2}$ obtained by MS-transformation. We set $y'_0, x'_0 = 0$ and $0 \leq m \leq D$. The MS-distance in lower bound form between $\mathbf{x}$ and $\mathbf{y}$ is defined as:

$$MSL(\mathbf{x}', \mathbf{y}', m) = D((\mu_\mathbf{x} - \mu_\mathbf{y})^2 + (\sigma_\mathbf{x} - \sigma_\mathbf{y})^2) + \sigma_\mathbf{x}\sigma_\mathbf{y}\sum_{i=0}^{m}(x'_i - y'_i)^2. \tag{4}$$

The MS-distance in lower bound form provides a lower bound for the Euclidean distance between two points. It has the following properties.

Theorem 2.5. *For any two points* $\mathbf{x}$ *and* $\mathbf{y}$*, we have the following properties [36]:*

- $MSL(\mathbf{x}', \mathbf{y}', m)$ *monotonically increases with* m.
- $\delta(\mathbf{x}, \mathbf{y}) \geq MSL(\mathbf{x}', \mathbf{y}', m)$ *for all* $0 \leq m \leq D$.
- $\delta(\mathbf{x}, \mathbf{y}) == MSL(\mathbf{x}', \mathbf{y}', m)$ *when* $m = D$.

Those properties enable us to use the MS-distance as a lower bound for the Euclidean distance. We can gradually increase dimensions to obtain a better lower bound until exact distance is calculated.

2.4 DCO and Lower Bound Pruning

Distance comparison operation (DCO) [29] is a key step in the search process. Alg. 1 outlines a common step in search procedure, within which the DCO plays a pivotal role. Specifically, DCO computes the distance between a candidate point $\mathbf{x}$ and the query $\mathbf{q}$, then compares it against a dynamic threshold τ. If $dist(\mathbf{x}, \mathbf{q}) < \tau$, then the actual distance will be returned. The DCO result will be used to update the result set and the threshold τ. Previous studies [29] have demonstrated that distance computation constitutes a significant portion of search overhead, and most points will exceed the threshold, providing an opportunity to improve performance.

Algorithm 1: Common Step in Search

Input: query $\mathbf{q}$, data structure T, result set R, threshold τ
Output: result set R, new threshold τ

```
1 Let x be a point from T;
2 dist_x = δ(x, q);                          // calculate distance
3 if dist_x < τ then
4     Update result set R;
5     Update τ;
6 return R, τ;
```

The lower bound is widely used in index design to enable efficient pruning. If a lower bound $LB(\mathbf{q}, \mathbf{p}) \geq \tau$, $\mathbf{p}$ can be pruned. Lower bound can be used to help pruning points in DCO [29]. A common approach to lower bound calculation is to use the triangle inequality [84]. Due to the curse of dimensionality, it is hard to provide a tight lower bound [59]. The partial distance $\delta_d(\mathbf{p}, \mathbf{q})$ of two points $\mathbf{p}$,$\mathbf{q}$ PDScanning [23, 29] can serve as lower bound. We can avoid the expensive computation cost and improve the search performance using lower bound.

3 Analysis

In this chapter, we first analyze DCO optimization methods represented by ADSampling to reveal the reasons for their performance degradation. Then we analyze the search method used in graph-based method in the case of HNSW to demonstrate the redundant evaluation of neighbor nodes. These analyses collectively illustrate the design principle of our search method SkipComputing.

3.1 Cost Analysis

The ADSamping[29], DDC[75] and DADE[24] methods work mainly by obtaining partial distances through paused distance computation, followed by estimation and testing for point preservation. If the point cannot be pruned, the distance computation continues by incrementally adding more

dimensions to conduct the testing until the distance is fully computed or terminated early. This process incurs additional costs, which can outweigh the benefits of early termination, especially in SIMD-enabled environments.

We take ADSampling as an example to illustrate this. ADSampling is based on Equation 5 to determine whether to terminate the distance computation early. The main meaning of this equation is to measure how certain the actual distance is larger than the threshold by using the estimated distance in subset of dimensions. The larger the gap to the threshold, the more certain we are that it is safe to skip the full computation. The ϵ_0 is a hyperparameter that controls the confidence level, and τ serves as a distance threshold, where only the points within this distance are kept. The function $f(d)$ is precomputed in a lookup table to reduce the computation, where d is the number of dimensions used for estimation and D is the total number of dimensions.

$$\delta_d(\mathbf{p}, \mathbf{q}) \geq \tau \cdot f(d),$$
$$f(d) = \frac{d}{D}\left(1 + \frac{\epsilon_0}{\sqrt{d}}\right)^2, \tag{5}$$

DDC method makes termination decisions based on the standard deviation computed across different numbers of dimensions, while the DADE method relies on eigenvalues and sampling errors to estimate how certain the distance exceeds the threshold, thereby terminating the computation.

While these methods can terminate distance computations early, they inevitably incur additional costs. These costs contain two parts: (1) the cost associated with estimation and testing calculations, such as those involved in Equation 5. (2) the cost of continuously pausing distance computation for testing, which prevents full utilization of computational resources and results in longer distance computation times compared to direct scanning.

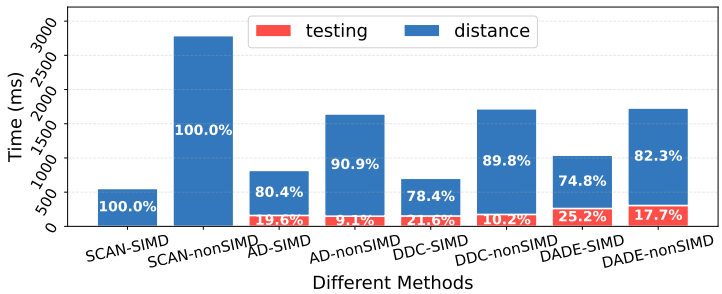
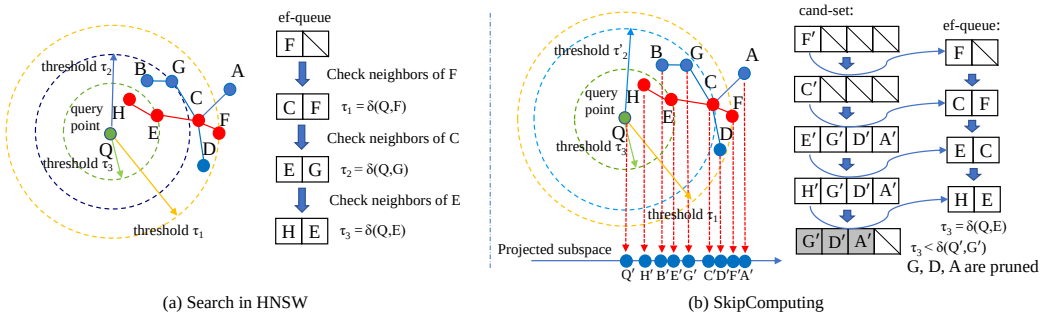


Fig. 2. Profiling of distance computations. The figure shows result on 100000 distance computations between two 960-dimensional vectors: SCAN computes the exact distance, while other methods terminate at 480 dimensions.

Figure 2 illustrates the results of the performance analysis of distance computations with and without ADSampling, DDC, and DADE methods in both SIMD and non-SIMD settings. We use perf tools [1] to analyze proportions. We have the following observations: **SIMD setting:** (1) All comparable methods show a longer execution time than SCAN-SIMD when the computation terminates. (2) Testing costs constitute a significant portion of the total execution time compared to direct distance computation (SCAN-SIMD), with ADSampling taking about one-quarter of the time of SCAN-SIMD, and DADE shows comparable time. (3) The distance computation phase with smaller dimensions exhibits longer time compared to SCAN-SIMD's 960 dimensions. **Non-SIMD setting:** (1) A noticeable performance improvement emerges when completing the computation. (2) Testing costs show relatively lower impact compared to SCAN-nonSIMD. (3) The distance



(a) Search in HNSW (b) SkipComputing

Fig. 3. Search procedure of graph-based method. (a) During HNSW's search procedure, the algorithm conduct DCO on all neighbor points (including A, D, G, E of C) to evaluate their qualification for the ef-queue until the search completion (we omit the cand-queue here). If we can reduce the number of points participating in DCO, we can decrease computational cost. (b) In our proposed method, neighboring points are evaluated in the projected subspace and added to the candidate set cand-set. The ef-queue is updated by selecting point from cand-set. As the search progresses, the threshold gradually decreases, enabling the pruning of points G, D, and A, skip redundant distance computation by avoiding DCO and improve search performance.

computation phase does not show a significant increase in time. These observations provide critical insights into the performance degradation of DCO optimization methods, highlighting the need to balance the gains and costs of optimization.

To achieve performance gains by early termination in SIMD setting, the computation must be terminated earlier in the process. For example, if the time taken to obtain the distance between two points after introducing ADSampling is three times that of direct distance computation, the break-even point necessitates terminating calculations after processing 1/3 of the dimensions.

For a point whose distance $dis > \tau$ with distance gap $\alpha = \frac{dis-\tau}{\tau}$, Gao and Long [29] shows the expected sampled dimensions is $\mathbb{E}(\hat{D}) = O(\min(D, \alpha^{-2}\epsilon_0^2))$, leading to expected iterations $\mathbb{E}(t) = \lceil \mathbb{E}(\hat{D})/\Delta d \rceil$. As $\alpha \to 0$ (objects close to the threshold), or ϵ_0 is larger (more confidence required), more iterations t would be required. And t increases quadratically with α and ϵ_0, making t approach $\lceil D/\Delta d \rceil$ quickly. This means that the expected number of iterations for a point to be terminated early is larger, leading to more computations.

3.2 Search in HNSW

The search algorithm used in HNSW is based on the greedy routing strategy best-first search [70], maintaining two priority queues (or a single sorted array in NSG [27]) to iteratively evaluate candidate points on the graph. Figure 3a shows the search procedure. The search process consists of two implicit phases: (1) candidate generation and (2) candidate verification, which we analyze below based on the roles of two queues.

The first priority queue is ef-queue, which retains $efSearch$ closest points observed during graph traversal. The ef-queue serves two primary functions: (1) dynamically maintaining provisional top-k search results, and (2) controlling the backtracking scope through the queue capacity ($efSearch$), with larger values allowing broader graph exploration to generate more candidates.

The second priority queue is cand-queue, which tracks all points that have been previously added into ef-queue while neighbors have not been explored yet. The neighbors of points in the cand-queue serve as candidate points and will be verified whether to insert into ef-queue.

The above analysis demonstrates that DCO is performed when updating the ef-queue. The current implementation requires DCO on all neighboring nodes, computing the distance for each neighbor

node, and updating the ef-queue. Accessing all neighboring nodes is unnecessary. As shown in [29], most points will not be added to the ef-queue, meaning they neither become part of the final result nor contribute to providing more candidate points in subsequent searches. Exhaustively comparing all neighboring points creates a computational bottleneck. If we can avoid those redundancy distance calculations, we can improve search performance.

The optimization of DCO relies on the comparison with the threshold. We can utilize the lower bound pruning in Section 2.4 or the hypothesis testing in Section 3.1 to pruning points. The smaller the threshold, the better the filtering capability. The threshold monotonically decreases during the search process. In figure 3, the threshold decreases from τ_1 to τ_3. When we access neighbor nodes A, D, and G of node C, the threshold is relatively large. We may not quickly prune them using this threshold, leading to increased computational costs. If we can delay the DCO of these points until the threshold is smaller, we would have a better chance of filtering them with a small computational cost. The existing search approach does not meet the requirements, which requires a redesign of the search strategy.

4 Basic Search Method

In this section, we describe our search method, which consists of two key components: (1) candidate search and (2) adaptive lower bound pruning. We first introduce each component in detail and then present the complete algorithm that integrates all these components.

4.1 Candidate Search

Candidate search employs a two-stage approach to exclude low potential points during search to reduce the number of points conducting DCO: (1) candidate generation and (2) candidate evaluation. The generation stage populates the candidate points, whereas the evaluation stage selects points from candidate points for DCO. Two critical challenges must be addressed: (1) how to populate the candidate set effectively without directly computing the full distance, and (2) how to make decision policy for identifying the promising candidate point from the candidate set for DCO. In this subsection, we present the distance estimation based candidate generation and evaluation method within the two-stage approach, along with theoretical analysis.

4.1.1 Candidate Generation. Candidate generation aims to delay DCO for points accessed during graph traversal. We modify *cand-queue* to a candidate set cand-set to store the candidate points. In index construction phase, we reduce data dimensionality via random orthogonal projection, selecting the first d dimensions as the target subspace while retaining the remaining $(D - d)$ dimensional data to calculate exact distance. In search stage, we use the distances in this lower-dimensional space to estimate the exact distance, which is then used to sort the cand-set. The point with the smallest estimated distance is selected as the next candidate for DCO. We leverage the distributional JL lemma, ensuring that random projections approximately preserve distances. Algorithm 2 details the process of inserting a neighbor into the cand-set. We avoid directly conducting DCO to update ef-queue by first inserting points into cand-set. By computing distances directly in the projected subspace, we avoid directly applying estimation-and-testing based DCO optimization strategies to these points, eliminating computational interruptions and ensuring fully leveraging SIMD for distance computation.

4.1.2 Candidate Evaluation. Our candidate evaluation mechanism is designed to identify suitable points in the candidate generation stage for DCO while filtering out those that did not meet the requirements. By selectively applying DCO operations to high-potential points while filtering out low-potential ones, we avoid unnecessary DCO overhead for the latter. These points only require the computation of the distance in subspace. We choose points with minimal distance

Algorithm 2: CandidateGen($\mathbf{q}', \mathbf{p}', cand\text{-}set, C$)

Input: projected query $\mathbf{q}'$ and point $\mathbf{p}'$, candidate set *cand-set*, set size C
Output: candidate generation result

```
dist = (D/d)||(p' − q')||_2^2 ;
if dist < cand-set.maxdist or |cand-set| < C then
    cand-set.insert(dist, p');                    // maintain set size as C
```

that simultaneously satisfy the threshold comparison criteria as candidates for DCO. The specific process is illustrated in Algorithm 3. Comparison with the threshold τ is made with the estimated distance in the projected space. This mechanism serves as the key to reduce the number of points involved in DCO. As the search progresses, the threshold τ decreases, leading to an evolving selection criterion. Points previously added to the cand-set eventually fail to meet the updated requirements, eliminating the need for further DCO for those points. The candidate point is selected from the cand-set, and the distance is compared with the threshold. If the distance is smaller than the threshold, we proceed with DCO. Otherwise, we terminate the search immediately. This early termination is justified since we always select the point with minimal distance from the cand-set, guaranteeing that subsequent points will also fail to meet the requirements.

Algorithm 3: CandidateEva(cand-set, ϵ, τ)

Input: candidate set cand-set, error ϵ, threshold τ
Output: candidate evaluation result

```
dist, p' = get point with minimal estimated distance in cand-set ;
remove point p' from cand-set;      // reserve in cand-set with delete marking
if dist > (1 + ε) · τ then
    return false;                                          // terminate search
return true and (d/D)dist;                                      // conduct DCO
```

We have the following Theorem 4.1 to show that if a point is close to the query (smaller than the point generates the threshold) in high-dimensional space, it will be selected with high probability. And if a point is far from query (larger than the point generates the threshold), it will be more likely to be excluded without performing DCO.

Theorem 4.1. *Given a point $\mathbf{p} \in P$ from the dataset, a query point $\mathbf{q}$, and a parameter $\epsilon \in (0, 1)$. Let $\mathbf{A} \in \mathbb{R}^{d \times D}$ be the first d rows of a random orthogonal matrix $\mathbf{A}^{D \times D}$, and let τ represent the farthest square distance in the ef-queue. The dimensions d and D (with $d \leq D$) correspond to the projected and original spaces, respectively. Furthermore, let $p_1, p_2 \in (0, 1)$. We then have:*

- *If $\delta(\mathbf{p}, \mathbf{q}) < \tau$, then $\Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot \tau) > p_1$.*
- *If $\delta(\mathbf{p}, \mathbf{q}) > \tau$, then $\Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot \tau) < p_2$.*

Proof. According to distributional JL lemma, the projected result follows a probability distribution. Let $\Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot r) = F = p$, where $F(\cdot)$ is the cumulative distribution function. If $r = \delta(\mathbf{p}, \mathbf{q}) < \tau$, we have $\Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot \tau) > \Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot r) = p = p_1$ Similarly, if $r = \delta(\mathbf{p}, \mathbf{q}) > \tau$, we have $\Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot \tau) < \Pr(\|\mathbf{A}(\mathbf{p} - \mathbf{q})\|_2^2 < (1 + \epsilon)d/D \cdot r) = p = p_2$. □

In order to decrease the probability of filtering out points that are still valid, we set the size of *ef-queue* to be larger than k which is the final number of points we want, to have a larger τ

during the search. Let $\mathbf{p}^*$ be the nearest point of $\mathbf{q}$. Define τ_k and τ_l as thresholds when $ef\text{-}queue$ has size k and l (with $l > k$), respectively. It follows that $\tau_k \leq \tau_l$. Let $dist' = \|\mathbf{A}(\mathbf{p}^* - \mathbf{q})\|_2^2$ and $dist = \|\mathbf{p}^* - \mathbf{q}\|_2^2$, then we have

$$\begin{aligned} \Pr(dist' < (1+\epsilon)d/D \cdot dist) &< \Pr(dist' < (1+\epsilon)d/D \cdot \tau_k) \\ &\leq \Pr(dist' < (1+\epsilon)d/D \cdot \tau_l), \end{aligned} \tag{6}$$

which means we have a high probability to preserve the point $\mathbf{p}^*$ in the cand-set. Also, it helps us to traverse more part of the graph as we will select more points in the cand-set.

4.2 Adaptive Lower Bound Pruning

After the DCO points are selected, the next step is to conduct the DCO. As discussed in Section 3.1, current DCO optimization methods use estimation-and-testing process. We propose an adaptive lower bound pruning (ALBP) method that replaces estimation-and-testing. The complete process is described in Algorithm 4 which follows [29, 71]. Since the subspace distance is calculated during the candidate search, lower bound pruning can accumulate the distance using this precomputed distance. Δ_s controls how many dimensions are included in each iteration.

Algorithm 4: PointPrune($\mathbf{p}', \mathbf{q}', \Delta_s, pdis = 0, \tau, m$)

Input: transformed data $\mathbf{p}'$, query $\mathbf{q}'$, step Δ_s, partial distance in candidate search $pdis$, threshold τ, start dimension m
Output: pruned or exact distance

```
if pdis > τ then
    return false;
lowerbound = pdis;
r = m;                                          // assume divisable by Δ_s
while r < d do
    // Reuse the previous calculation
    lowerbound+ = Σ_{i=r}^{r+Δ_s} (p'_i − q'_i)^2;          // use SIMD
    r+ = Δ_s;
    if lowerbound > τ then
        return false;                                       // pruned
return true and exact distance;
```

The core challenge involves constructing tighter lower bounds for early termination. We can use Principal Component Analysis (PCA) to transform the vector as shown in [71]. We will discuss more optimization details in the later section.

4.3 Complete Search Method

The algorithm 5 describes the complete search method. We make three modifications to HNSW's search algorithm. (1) The ef-queue is updated by selecting points from cand-set and using lower bound pruning method. (2) Instead of directly performing DCO on each neighbor points, we insert them into cand-set for later processing. (3) The search terminates when the cand-set is empty or no points meet the candidate selection criteria. In the specific implementation, when we delete the selected points in the cand-set, we perform tag deletion while the data point still participate in the sorting operation for efficiency of updating cand-set.

The method involves five hyperparameters: subspace dimensions d, error ratio ϵ, cand-set size C, $ef\text{-}queue$ size L, and step size Δ_s. Intuitively, a larger L implies a larger threshold, allowing us to terminate the search later and expand the search scope. Increasing d can enhance the accuracy of distance estimation at the cost of more computations. We can use Equation 3 to calculate the probability of the estimated distance deviates from the true value by a factor of $(1+\epsilon)$ based on d and ϵ. Increasing d, ϵ, and L will decrease the probability of excluding true neighbors at the cost of increased computational overhead. A smaller C can terminate the search earlier, which has a similar function as $efSearch$ in HNSW to get different recall. A larger Δ_s will result in redundant calculations, while a smaller Δ_s will incur the overhead of iterations. In practice, those hyper-parameters require careful tuning based on dataset to balance the recall and query latency.

Algorithm 5: SkipComputing($\mathbf{q}, \mathbf{A}, d, \Delta_s, k, L, C, \epsilon$)

Input: query point $\mathbf{q}$, random projection matrix $\mathbf{A}$, subdim d, top-k k, $ef\text{-}queue$ size L, $cand\text{-}set$ size C, error ratio ϵ

Output: k approximate nearest neighbors to $\mathbf{q}$

```
1  q' = Aq ;
2  Initialize ef-queue, cand-set, visited set V;
3  cand-set ← ep', δ_d(q', ep');                                  // entry points ep
4  Update V with ep';
5  while cand-set is not empty do
6      dist', p' = get minimal point from cand-set; // minimal in cand-set, dist' is the
         actual square distance in subspace
7      if CandidateEva(cand-set, ε, τ) return false && |ef-queue| equals L then
8          break;
9      if |ef-queue| < L then
10         Insert p', exact distance into ef-queue;
11         τ = ef-queue.top();                          // farthest point, square distance
12     else
13         if PointPrune(p', q', Δ_s, dist', τ, d) return true then
14             Update ef-queue with {p', exact distance}; // maintain ef-queue size as L
15             τ = ef-queue.top();
16     for e' ∈ neighborhood(p') do
17         if e' ∈ v then
18             continue;
19         Update V with e';
20         CandidateGen(q'_[0:d], e'_[0:d], cand-set, C);
21 return top-k points in ef-queue;
```

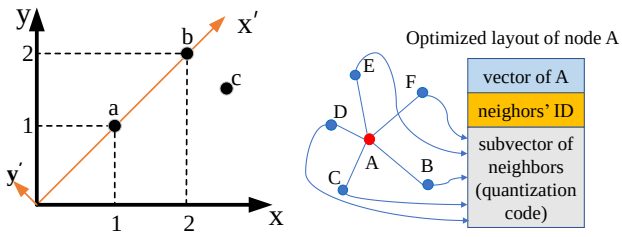
5 Optimized Search Method

This section describes optimizations for our basic search method. We identify three limitations: First, the basic subspace projection method remains inadequate due to the distance estimation error. Second, while PCA-derived lower bounds represent an improvement, it remains to be improved in certain cases. Third, the algorithm neglects memory access efficiency during candidate

generation, creating significant bottlenecks in search. To address those limitations, we propose two optimizations: (1) distance decomposition and (2) data layout optimization. The former improves the distance accuracy and tightens the lower bound, while the latter improves memory access efficiency by using cache effectively in candidate search.

5.1 Distance Decomposition

5.1.1 Limitation of lower bound through PCA. As described in the algorithm 4, the lower bound between two points increases with more iterations. Although more iterations yield a tighter lower bound, it also incurs higher computational cost. Our objective is to derive a more compact lower bound with fewer computational costs. We rotated the data through PCA to get a good lower bound with fewer iterations. As illustrated in Figure 4a, let x', y' be the new axis after PCA is performed on the datasets $\mathbf{P}$, and let $\mathbf{a}, \mathbf{b}$ be two points in $\mathbf{P}$. The PCA transformation projects them to x' and y', enabling measurement of the distance along the axis x'.



(a) PCA (b) Memory Layout

Fig. 4. Illustration of PCA and Memory Layout.

However, this approach is constrained by a key limitation. For points like $\mathbf{b}$ and $\mathbf{c}$ that are not aligned with the x' axis, the first principal component cannot characterize their distance. This limitation arises because one uses the distance in one subspace as a lower bound, while the primary distance between two points lies in another subspace. To improve the lower bound, additional iterations are required to incorporate distances from more subspaces. We further optimize the lower bound to mitigate this limitation.

5.1.2 Refining lower bound. Our core idea involves a distance decomposition approach to break down the square Euclidean distance into different components to obtain a more compact lower bound for the ANN search. Specifically, we decompose the distance into two complementary components: a global distance summary that holistically captures the overall coarse-grained distance characteristics and a subspace specified component that refines the distance within a specific subspace. The first component reduces the subspace selection bias, while the second component enables refinement through subspace iteration.

Previous research has established the theoretical foundation for employing the mean and variance as lower bounds for Euclidean distance between points. Building on this foundation, we propose our method MSPCA, which builds upon the MS-distance [36] by using mean and variance as the distance summary.

From the MS-transformation in Definition 2.3, we can further decompose the second component into two terms: $\sum_{i=0}^{m}(y'_i - x'_i)^2$ and $\sigma_x\sigma_y$. The first term represents the square Euclidean distance between two points transformed by mean and variance. We optimize this part by applying PCA

to rotate the data, enabling faster convergence to a better lower bound during iterations. This approach established the following theorem.

Theorem 5.1. *Let* $\mathbf{x}', \mathbf{y}'$ *be the* $(D+2)$*-dimensional vector transformed according to MS-transformation in Definition 2.3. We apply PCA to subvector* $\mathbf{x}'_{\mathbf{sub}} = [x'_1, x'_2, \ldots, x'_D]$ *to get*

$$\mathbf{x}'' = [\mu_x, \sigma_x, x''_1, x''_2, \ldots, x''_D]. \tag{7}$$

We do the same for $\mathbf{y}'$ *to get* $\mathbf{y}''$*. The MS-distance* $MSL(\mathbf{x}'', \mathbf{y}'', m)$ *can still serve as a lower bound and every properties of Theorem 2.5 still hold.*

Proof. The transform matrix $\mathbf{A} \in \mathbb{R}^{D\times D}$ from the PCA is orthogonal matrix. Let $\mathbf{v}' = \mathbf{x}''_{\mathbf{sub}} - \mathbf{y}''_{\mathbf{sub}}, \mathbf{v}'' = \mathbf{x}''_{\mathbf{sub}} - \mathbf{y}''_{\mathbf{sub}}$ we have

$$\|\mathbf{v}''\|_2^2 = (\mathbf{A}\mathbf{v}')^{\mathsf{T}}(\mathbf{A}\mathbf{v}') = \mathbf{v}'^{\mathsf{T}}\mathbf{v}' = \|\mathbf{v}'\|_2^2, \tag{8}$$

which means distance is the same. When $m' < m$, the function $\delta_{m'}(\mathbf{x}''_{\mathbf{sub}}, \mathbf{y}''_{\mathbf{sub}})$ increases monotonically with m', its maximum value being $\delta_m(\mathbf{x}''_{\mathbf{sub}}, \mathbf{y}''_{\mathbf{sub}}) = \delta_m(\mathbf{x}'_{\mathbf{sub}}, \mathbf{y}'_{\mathbf{sub}})$. □

Theorem 5.1 shows that after applying PCA to the MS transformed data, MSPCA can also be used for pruning in the search process by iteratively refining to obtain a tighter lower bound until the exact distance is obtained.

5.1.3 Improving the iteration efficiency. Our adaptive lower bound pruning method iteratively refines the lower bound by computing the distance in subspace. The inclusion of $\sigma_x\sigma_y$ in second component of MS-distance introduces per-iteration multiplication overhead, potentially affecting performance. We optimize this part to improve efficiency while preserving correctness. The iteration process involves threshold comparison, formally expressed as:

$$D \cdot ((\mu_{\mathbf{x}} - \mu_{\mathbf{y}})^2 + (\sigma_{\mathbf{x}} - \sigma_{\mathbf{y}})^2) + \sigma_{\mathbf{x}}\sigma_{\mathbf{y}} \sum_{i=0}^{k} (x'_i - y'_i)^2 > \tau. \tag{9}$$

We apply the following one-time transformation to the threshold,

$$dis' = \sum_{i=0}^{d} (x'_i - y'_i)^2 \leq \frac{\tau - D((\mu_{\mathbf{x}} - \mu_{\mathbf{y}})^2 + (\sigma_{\mathbf{x}} - \sigma_{\mathbf{y}})^2)}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}} = \tau'. \tag{10}$$

Consequently, we compare the distance between subspaces directly with the modified threshold τ' instead of the original threshold τ. This approach requires only one multiplication and one division operation: one during initialization and another when returning the actual distance. Crucially, the iterative process itself becomes multiplication-free, reducing computational overhead.

5.1.4 Lower bound based candidate search. In this subsection, we present our lower bound based candidate search method. Accurate distance estimation is crucial for query performance, as inaccuracies may falsely exclude valid points, requiring higher projection dimensions to mitigate errors, thus increasing computational overhead. Building on our earlier distance decomposition method, we replace the random projection with the following two options to optimize the candidate search: (1) replacing random projection with PCA-learned projection matrix, retaining the top d dimensions as subvector used in candidate search. (2) using the MSPCA method to combine the global distance summary with the top d dimensions of the second component. Through these two options, we can obtain more accurate distances during candidate search, which effectively guides our search process.

In candidate generation stage, the distance computed through the two options provides sufficiently tight lower bound, enabling us to use lower bound to conduct candidate evaluation. By

Algorithm 6: LBCandidateGen($\mathbf{q}', \mathbf{p}', \textit{cand-queue}, C$)

Input: projected query $\mathbf{q}'$ and point $\mathbf{p}'$, candidate set *cand-set*, set size C
Output: candidate generation result

```
if MSPCA is used then
    dist = MSL(p', q', d);
else
    dist = ||(p' − q')||_2^2;
if dist < cand-set.maxdist or |cand-set| < C then
    cand-set.insert(dist, p');                          // maintain set size as C
```

Algorithm 7: LBCandidateEva(*cand-set*, τ)

Input: candidate set *cand-set*, threshold τ
Output: candidate evaluation result

```
dist, p' = get point with minimal distance in cand-set ;
remove point p' from cand-set ;
if dist > τ then
    return false;                                       // terminate search
return true and dist;                                   // conduct DCO
```

evaluating candidates based on these lower bounds, we can safely skip DCO for a point when the lower bound calculated in candidate generation stage exceeds the current threshold. Furthermore, as the search progresses and the threshold continuously decreases, points that could not be pruned at larger thresholds may become eligible for pruning. For example, in Figure 3, as the search progresses, we can prune points G, D, and A with smaller threshold, to avoid redundant distance computations. The modified method is shown in alg. 6 and alg. 7. By substituting the corresponding algorithmic components in alg. 5, we can derive our complete optimized search method.

5.2 Data Layout Optimization

To improve cache efficiency in the candidate search, we propose a data layout optimization method that mitigates the performance bottleneck caused by cache inefficient random memory access patterns, when retrieving neighboring nodes. While co-locating vectors of neighboring nodes improves access locality, it incurs prohibitive memory footprint overhead. Our method leverages the dimension-reduced vector within SkipComputing to achieve both space efficiency and memory access optimization. Figure 4b shows the basic layout of a single node. We make three trade-offs: Space-for-Access, Precision-for-Space, and Space-for-Precision to optimize the search.

By storing subvectors of each node's neighbors adjacently, we improve access locality, yet naive replication incurs prohibitive memory footprint, necessitating further optimization (Space-for-Access). Quantization method can be used to reduce the data volume, and recent advanced methods such as DiskANN [39], NGT-QG [38] and SymphonyQG [32] highlight the potential of combining quantization with graph. Building upon this, we implement an additional quantization layer for further compression of the vector data after dimensionality reduction, achieving a more compact representation with reduced storage requirements (Precision-for-Space). The quantization inevitably introduces precision loss of the distance in candidate search, which can impact the search

Table 1. Dataset Statistics

Dataset	Size	Dimension	Query Size	Semantics
GIST	1,000,000	960	1000	Image
MSong	994,185	420	1000	Audio
DEEP	1,000,000	256	1000	Image
Tiny5M	5,000,000	384	1000	Image
OpenAI	999,000	1536	1000	Text
arXiv	2,253,000	768	1000	Text

performance. To mitigate this loss, we use a subspace with more dimensions in the quantization, which will increase memory overhead (Space-for-Precision).

Candidate search with quantization. We design our strategy based on our PCA-based candidate search. We need to balance accuracy loss and memory access optimization carefully. Although quantization enables storage efficiency, excessive quantization errors can adversely affect search performance. Therefore, careful selection of quantization methods is essential. A direct approach is to adopt SQ8 (8 bit scalar quantization) for the sub-vector. However, this may lead to significant memory overhead as there are many repeat data to be stored. RaBitQ [30] is the state-of-the-art in quantization techniques, achieving remarkable accuracy with minimal computation and space overhead. SymphonyQG [32], which combines RaBitQ with graph-based methods, has also demonstrated impressive results. Building upon this, we integrate these into our search algorithm to compress the dimensionality reduction vectors produced by PCA used in candidate search with three key components: (1) Applying RaBitQ to quantize projection data and utilizing vertex $\mathbf{p}$'s dimension-reduced data as the normalization reference for its neighbors. (2) For each vertex $\mathbf{p}$, we store the quantized neighbor data adjacent to itself (shown in 4b), enabling efficient memory access. (3) Modifying the graph structure like HNSW via SymphonyQG's refining method to ensure that out-degrees of each node are multiples of 32 for Fast-scan optimization.

Assuming each node in the graph has R neighbors and there are M nodes that need to access their neighbors (length of search path), as estimated in the HNSW[51], $M = O(logN)$, where N is the total number of data points. Without optimization, the number of random memory accesses required to visit neighboring nodes is $O(MR)$. After optimization, it requires $O(M)$ random accesses.

6 Evaluation

6.1 Experimental Settings

Datasets. We use six public datasets which are widely used in ANN search to benchmark performance [7, 29, 44]. Table 1 shows the statistical information about datasets used in experiments.

Performance metrics. To evaluate accuracy, we use recall, while efficiency is measured using queries per second (QPS). The overall performance of different methods is evaluated using the QPS-Recall curve, a standard performance measurement in ANN search. We transformed the queries via matrix multiplication using the implementation from [29], disabling hardware-specific optimizations as specified in the original paper. The query time is measured end to end including the transformation of queries.

Compared methods. The basic graph-based method we used is HNSW, consistent with ADSampling. The methods we used for comparison include ADSampling [29], PEOs [49], DDC_{res} [75], DADE [24]. We list details below:

- HNSW: The vanilla hierarchical navigable small world graph implemented in hnswlib [51].

- HNSW-AD++: ADSampling [29] method with further optimization of the search by decomposing the search queue.
- HNSW-PEOs: PEOs [49] is a probabilistic routing method to identify candidate points. This method optimizes memory access of the testing data to achieve higher performance, and we use it to compare with our data layout optimized variant.
- HNSW-DDC$_{res}$: DDC$_{res}$ [75] is a variant in DDC for euclidean distance that takes the residual dimension variance for pruning.
- HNSW-DADE [24]: A data-aware distance estimation approach using PCA to approximate the exact distance in a lower-dimensional space and conduct hypothesis testing.
- SkipComputing-ALBP: Using PCA as in [71] to conduct adaptive lower bound pruning, excluding the candidate search optimization.

For our method SkipComputing, we have the following variants:

- SkipComputing-Random: Candidate search and lower bound pruning stage use data transformed by random projection.
- SkipComputing-RandAD: Candidate search uses data transformed by random projection and using ADSampling in DCO stage.
- SkipComputing-PCA/MSPCA: The candidate search and lower bound pruning are both performed on the PCA/MSPCA transformed data.
- SkipComputing-MA: The data layout optimized method using PCA to transform data and RaBitQ [30] for quantization. MA is short for memory access.
- SkipComputing-ALBP+: Using MSPCA to conduct adaptive lower bound pruning, excluding the candidate search optimization.

Parameter Setting. For the construction of HNSW, the parameters M and $efConstruction$ which control the graph quality, are set to 16 and 500 respectively, consistent with ADSampling. For the compared methods, we use the default parameters of the respective methods. For our method, four parameters need to be set: the subspace dimensions d, error ϵ used in random variant, ef-queue size L used in candidate search, and Δ_s in lower bound pruning which controls the increase in the number of dimensions per iteration to improve the lower bound. The setting of d varies for each combination of dataset and method, while Δ_s is constant within a dataset. The ϵ is set to 0.2 by default while specific value 0.05 for MSong and 0.1 for both arXiv and OpenAI datasets. We set parameter L to $4 \cdot k$ for all methods in all datasets except the DEEP and tiny5M datasets, when requiring recall rates exceeding 95%, we assigned L to 300 and 700 for $k = 20$ and $k = 100$ respectively to ensure high recall performance. In practice, the L of some datasets can be set smaller than $4 \cdot k$. We omit the prefix SkipComputing for simplicity. The dimension parameters with different datasets are shown in Table 2.

Table 2. Setting of d and Δ_s

Dataset	Random(d)	RandAD(d)	PCA(d)	MSPCA(d)	MA (d)	Δ_s
GIST	480	480	128	128	128	64
MSong	256	256	192	192	256	32
DEEP	192	192	128	128	128	32
Tiny5M	192	192	192	192	256	64
OpenAI	768	768	512	512	512	128
arXiv	384	384	256	256	256	128

The parameters are determined by initial value assignment followed by further grid search refinement. For Random variant, we use Equation 3 to set the initial values of subspace dimension d

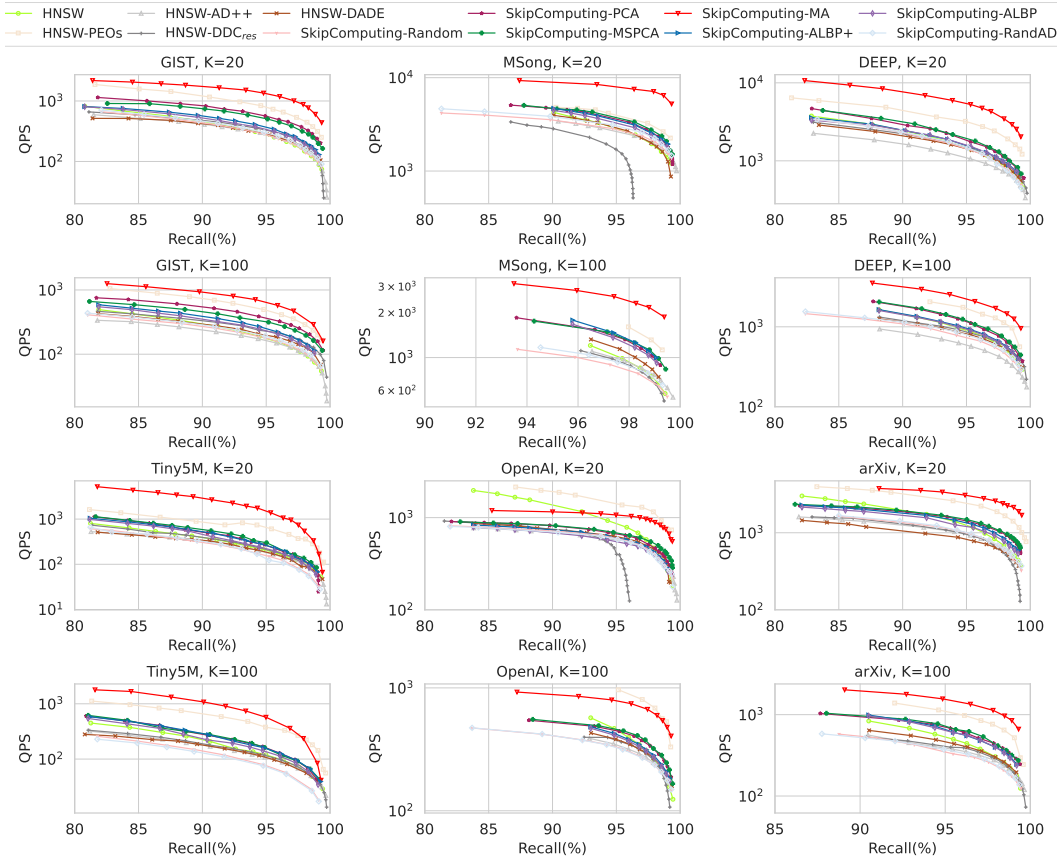


Fig. 5. Overall Performance. Our candidate search and lower bound pruning based method is superior to existing DCO optimization methods. Compared to PEOs which employs memory access optimization, our SkipComputing-MA achieves better performance in most cases.

and error ϵ to ensure a low probability (below 0.2) of estimation errors. For our optimized variants, the initial value d is determined as the smallest number of principal components that preserves 80% of the pairwise distance on the sampled vectors. The initial value of L is set to k, and Δ_s is set to twice the bit width of SIMD. After determining the initial values, we perform a grid search to refine the parameters in specific datasets.

All our C++ source codes are complied with g++ 9.5.0 using O3 optimization under Ubuntu 20.04LTS. We enable SIMD with AVX512 to calculate the distance between two points. The experiments are conducted on a machine with an Intel(R) Xeon(R) Silver 4210R CPU and 256GB memory.

6.2 Overall Search Performance

Figure 5 illustrates the overall search performance using the Recall-QPS curve, where a position closer to the upper right corner indicates better performance. We present the results for the top-20 and top-100 search performance in six datasets under two distinct scenarios: with and without memory optimizations. We have the following observations: (1) Our SkipComputing-PCA and

MSPCA variants consistently surpass ADSamping, DDC, DADE, and HNSW across all datasets. (2) SkipComputing-MA outperforms PEOs on most datasets, demonstrating the effectiveness of our optimization strategy. However, in some cases, its performance is affected by precision loss from dimensionality reduction, which increases search costs. Additionally, the disabling of hardware acceleration for matrix transformations results in longer processing times. (3) SkipComputing-Random's search performance is compromised by estimation errors and dimensional expansion, resulting in performance degradation relative to HNSW in specific cases. (4) Experimental results show that existing methods frequently exhibit inferior performance compared to HNSW, while our approaches consistently exceed HNSW's performance. This outcome provides empirical evidence supporting the validity of our cost analysis and our method's capability to effectively skip redundant distance computations. (5) SkipComputing-ALBP and SkipComputing-ALBP+ variants which exclusively employ adaptive lower bound pruning, outperform existing methods in most scenarios, demonstrating the efficacy of lower bound pruning strategy. (6) In scenarios with high dimensionality and low recall rates, the cost of query transformation cannot be overlooked, as exemplified by the OpenAI dataset when the recall rate falls below 95%. To enhance transformation speed, measures such as hardware acceleration and faster matrix computation algorithms can be employed.

6.3 Index Size

Table 3 illustrates the index size of different methods. Raw is the size of the dataset. For our PCA-based and other compared methods, the index size remains unchanged for HNSW. For the MSPCA method, each vector needs to be stored along with its mean and variance, resulting in a slight increase in storage space, which is very minimal and can be practically ignored. Our memory access optimization variant exhibits similar memory overhead compared to PEOs, while our method can achieve better performance. Combining with the performance shown in Figure 5, our PCA and MSPCA methods achieve superior performance while introducing negligible changes to the index size. Our memory access optimization method, when compared to PEOs, maintains a similar or comparable index size while outperforming PEOs in terms of performance.

Table 3. Index Size (MB) of Different Methods

Dataset	Raw	HNSW	MSPCA	MA	PEOs
DEEP	981	1119	1126	1954	2351
GIST	3666	3804	3812	4639	4624
Tiny5M	7344	8033	8071	14649	13202
OpenAI	5858	5996	6003	8293	7670
arXiv	6610	6920	6937	9901	10096
MSong	1597	1734	1742	3050	2749

6.4 Number of Dimensions Evaluated

Figure 6 shows the sum of dimensions involved in all distance computations across visited points during search. Although DADE and DDC methods can use fewer dimensions, they do not achieve better performance as shown in Figure 5. Our method achieves better performance despite evaluating more dimensions than DADE and DDC. For example, on the OpenAI dataset, our MSPCA variant outperforms competitors while utilizing more dimensions. The results indicate that our method has lower costs and can better enhance performance by using fewer dimensions involved in the computation compared to HNSW.

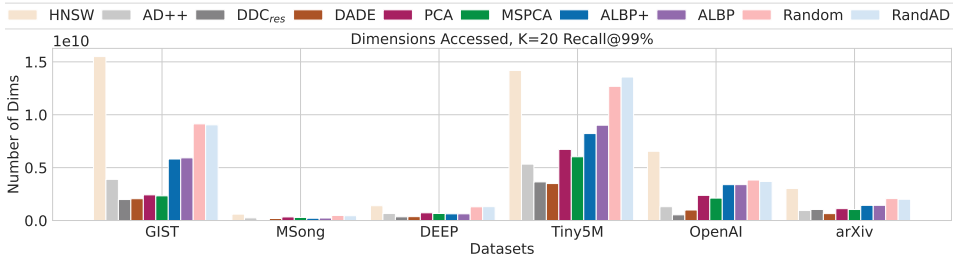
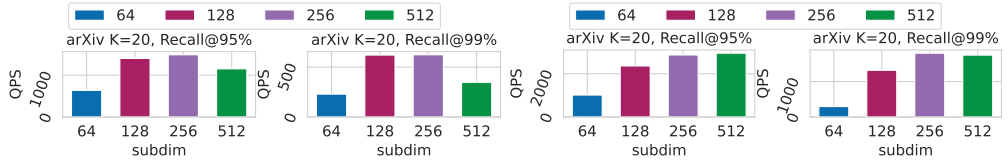


Fig. 6. Number of Dimensions.



(a) SkipComputing-MSPCA (b) SkipComputing-MA

Fig. 7. Search Performance on Different Subdim.

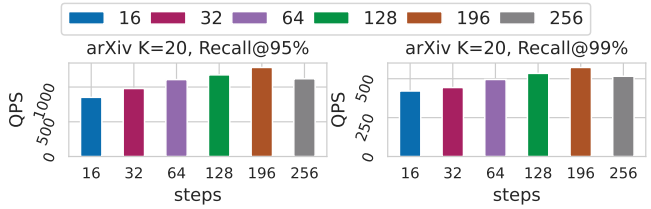


Fig. 8. Search Performance on Different Step Size.

6.5 Subspace in Candidate Search

Our method relies on dimensionality reduced subspace for candidate search. In this section, we explore the impact of different subspace dimensions on search performance. Lower dimensionality may lead to information loss, affecting the search progress in the graph and resulting in performance degradation due to more points evaluation. Conversely, higher dimensionality may introduce redundancy, increasing computational costs and also affecting performance. Using the arXiv dataset, we demonstrate the performance of SkipComputing-MSPCA and SkipComputing-MA at 95% and 99% recall rates across search subspaces of varying dimensions. As shown in Figure 7, we observe that using a subspace with different dimensions yields different search performance. For the SkipComputing-MSPCA method, increasing the dimensionality leads to more redundant dimensions participating in the distance computation, resulting in performance degradation. In contrast, for the SkipComputing-MA method, due to the use of RaBitQ, the computational cost is relatively low and there is no significant negative impact on performance. While further increasing dimensionality incurs additional space overhead, using all dimensions would render it equivalent to SymphonyQG. The selection of an appropriate subspace dimension in candidate search stage is crucial for achieving optimal search performance.

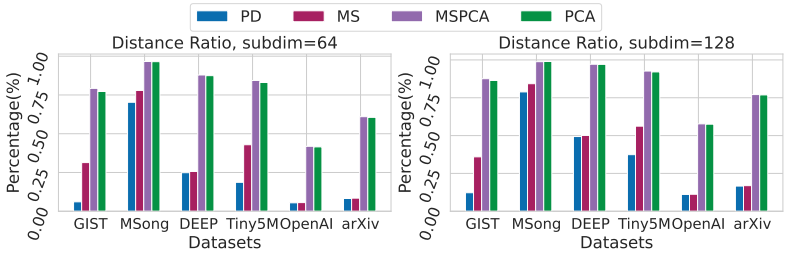


Fig. 9. Percentage of Distance.

Table 4. Points (x10^6) Evaluated in DCO (Recall@99%, k=20).

Method	GIST	MSong	DEEP	Tiny5M	OpenAI	arXiv
HNSW	15.04	1.40	5.10	41.32	3.99	3.70
MSPCA	1.20	0.13	0.27	1.61	0.32	0.28

6.6 Step Size

In this subsection, we present the impact of step size Δ_s on the performance of adaptive lower bound pruning in DCO optimization. Δ_s specifies how many dimensions are incrementally included in lower bound based pruning to check whether a candidate point exceeds the threshold as detailed in alg. 4. A smaller step size reduces redundant dimensions in computation but increases iterations, limiting processor utilization and pipeline efficiency. Conversely, a larger step size introduces additional redundant calculations, resulting in performance degradation. Figure 8 illustrates the performance of lower bound pruning on the arXiv dataset using different step sizes on MSPCA variant. We observe that as the step size increases, performance initially improves and then declines. Therefore, selecting an appropriate step size can help to optimize performance.

6.7 Proportion of Distance

Our adaptive lower bound pruning strategy iteratively improves the lower bound. For empirical validation, we present the distance proportion of four approaches: PDscanning (partial scan the origin vector) [29], MS-transform, PCA, and MSPCA under identical subspace dimensionality constraints. The analysis reveals that: (1) MS-transform achieves superior lower bounds compared to PDscanning, and (2) MSPCA demonstrates higher distance proportions than standard PCA, indicating its enhanced capability for generating tighter bounds that facilitate more effective pruning.

6.8 Points Evaluated in DCO

We perform candidate search in subspace to delay DCO for some points, allowing us to eventually exclude them from DCO. As shown in Table 4, we show that the number of points evaluated in DCO for different datasets is significantly lower for our MSPCA variant compared to HNSW. This demonstrates the effectiveness of our delay DCO strategy.

6.9 Cache Miss

The data layout optimization on MA variant can reduce cache misses in candidate search stage compared with the non-optimized PCA variant. We use perf tool to analyze the cache miss. As presented in Table 5, the proportion of cache misses during the candidate search phase compared to the

Table 5. Proportion of cache misses in candidate search stage.

Method	GIST	MSong	DEEP	Tiny5M	OpenAI	arXiv
PCA	0.82	0.87	0.89	0.89	0.86	0.78
MA	0.56	0.60	0.48	0.65	0.42	0.57

entire search process shows a significant decrease in MA compared to PCA variants, demonstrating the effectiveness of our approach.

6.10 Error Ratio

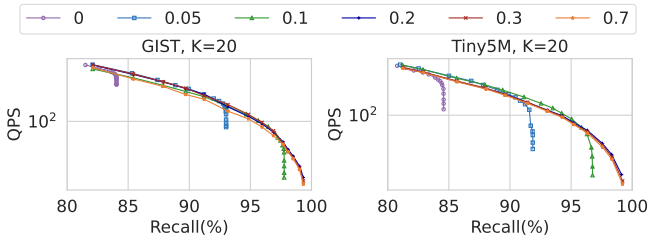


Fig. 10. Performance on different ϵ

The error ratio ϵ serves as a critical parameter in the Random variant during search, controlling the termination condition. Figure 10 presents the performance analysis of the Random variant on both GIST and Tiny5M datasets under varying ϵ values. When ϵ is too small, the search may terminate prematurely due to overestimation of distances, leading to lower recall rates. Conversely, setting ϵ too high can result in unnecessary traversals of additional points, causing performance degradation. The results indicate that an appropriate ϵ value is essential for search.

6.11 Comparing with Non-SIMD Setting

Table 6 presents the QPS comparison of different methods under SIMD and non-SIMD settings at 99% recall. Our PCA and MSPCA variants, which are optimized under SIMD setting, achieve superior performance. Although competing methods show some improvement on non-SIMD setting, their performance generally cannot surpass SIMD-accelerated full scan distance computations. Notably, applying these optimization techniques to already SIMD-optimized implementations fails to deliver additional enhancements, and in some cases results in performance degradation, a finding that aligns with our earlier discussion. Although our method generally demonstrates performance gains in non-SIMD setting, its improvement is occasionally less pronounced than competing approaches, such as on the GIST dataset where MSPCA outperforms AD++ and DADE but lags behind DDC_{res} and Random variant shows lower performance in some cases. This occurs because our method processes more dimensions in distance computations and there is a smaller performance penalty from early termination strategies in competing methods in non-SIMD setting. Our method is more effective on SIMD and can also help improve performance in non-SIMD setting.

7 Related Work

Approximate nearest neighbor search. Approximate nearest neighbor search has been extensively and deeply studied. Researchers have explored various approaches, including hashing-based [3, 4, 22, 28, 35, 62, 63, 82, 83], graph-based [25–27, 34, 48, 50, 64], tree-based [12, 18, 60, 77],

Table 6. Comparing performance (QPS) with SIMD and non-SIMD setting. We use superscript "-" to denote the method with non-SIMD settings. We omit the prefix HNSW and SkipComputing for each method.

Recall	Recall@99%, K=20					
Datasets	GIST	MSong	DEEP	Tiny5M	OpenAI	arXiv
HNSW	95	1418	590	58	255	442
HNSW$^-$	38	797	289	30	98	182
AD++	103	1613	443	58	310	377
AD++$^-$	83	1238	419	47	241	250
DDC$_{res}$	114	N/A	702	68	N/A	237
DDC$^-_{res}$	88	N/A	671	64	N/A	208
DADE	132	1385	624	66	310	395
DADE$^-$	126	1250	580	61	273	372
Random	108	1521	595	34	280	443
Random$^-$	49	815	285	21	128	209
RandAD	114	1686	430	33	273	421
RandAD$^-$	50	811	266	21	134	209
ALBP	126	1733	662	54	262	536
ALPB$^-$	70	1336	459	34	150	294
ALBP+	137	1943	693	67	282	586
ALBP+$^-$	79	1587	515	44	150	313
PCA	**225**	1807	765	64	332	685
PCA$^-$	113	968	404	40	202	308
MSPCA	197	**2011**	**823**	**79**	**403**	**727**
MSPCA$^-$	114	1136	431	47	208	405

and quantization-based [5, 9, 31, 40, 41, 43]. Among these, graph-based methods like HNSW [51] have gained widespread attention due to their excellent performance [33, 66, 76] and many improvement methods have been proposed to make it better [17, 39, 52, 55]. Numerous comprehensive surveys are available, systematically summarizing various ANN methods. A comprehensive survey of relevant ANNS methods is provided in [47], surveys of relevant graph-based methods are presented in [8, 70], a thorough review of hash algorithms for ANN is conducted in [13] and a comprehensive survey of various learning-to-hash techniques is offered in [65]. In addition, a series of benchmark tools [7, 11, 13, 71] are proposed to evaluate the performance of different methods.

Distance comparison optimization. ADSampling [29] uses random projection to estimate the distance. DDC [75] improves distance estimation accuracy by leveraging the data distribution and introduces a data-driven approach for distance correction. DDC$_{res}$ is the variant designed for the euclidean distance. DADE [24] proposed an unbiased distance estimation method. Fudist [71] is a DCO benchmark for ANNS based on hnswlib. In addition to the methods mentioned above that rely on early distance termination, there are also methods that employ routing tests to select appropriate points for distance computation. FINGER [16] introduces a fast inference technique that approximates the distance by estimating angles between neighboring residual vectors. PEOs employs a probabilistic routing method [49] based on space partition and random projection, while leveraging data locality to improve processing speed at the cost of memory consumption. The routing test data for all neighboring nodes of a node are stored contiguously, reducing random access overhead while enabling SIMD-based parallel processing of multiple points. As PEOs show superior search performance compared to FINGER, we compare our approach with PEOs.

PDX [44] is a method designed to improve the search performance of IVF under SIMD through the vertical layout of the dimensions. As noted in Section 7 of PDX [44], the proposed PDX layout and PDXearch are particularly suitable for bucket methods such as IVF. However, their applicability to graph-based method requires further exploration, as the absence of notion of blocks. Also, PDX-BOND relies on the layout's block statistics means and cannot be directly compared in graph-based method. Our method focuses on optimizing the search for graph-based method. Future work can explore the integration of our method with IVF, using PDX's layout optimization techniques to achieve better search performance, as well as how to apply the corresponding techniques on the graph and combine with our SkipComputing for improved performance.

Quantization with graph-based methods. Quantization techniques, widely studied for data compression, can be effectively combined with graph-based methods to enhance search performance. DiskANN [39] achieves disk-based search through quantization, while LVQ [2] introduces an optimized scalar quantization method to improve query performance in graph-based methods. NGT-QG [38] leverages fast scan to improve search efficiency, and SymphonyQG [32] method achieves impressive results by combining RaBitQ [30] with graph.

Orthogonal strategies. Besides DCO optimization, there are other works aim to maintain the basic data structure unchanged and adopt orthogonal strategies to optimize various methods. Several studies [15, 46] introduce learned adaptive early termination strategies to enhance the search process in graphs. In [81], a cardinality estimation method is proposed to estimate the minimum probing cardinality, to reduce the latency of the IVFPQ query. Starling [69] optimizes data layout to improve the I/O efficiency of disk-resident graph-based method, which can be used to improve the performance of various graph-based methods such as Vamana [39]. iQAN [56] is a parallel search algorithm to improve the search performance in graph-based method.

8 Conclusion

In this paper, we propose SkipComputing, a novel method to optimize the performance of ANN search via skipping redundancy distance computations. We skip redundant distance computations through two complementary strategies: First, we delay DCO for neighboring points through a two-stage candidate search method, allowing progressive excluding of low potential points during the search. Second, we employ a distance decomposition method that partitions distances into multiple components, allowing the use of tighter lower bounds to terminate the distance calculation early in DCO. Furthermore, we reduce random access cost in candidate search by data layout optimization strategy and leverage quantization techniques to further reduce memory consumption. Through these optimizations, we achieve superior search performance.

We can select different variants of SkipComputing based on three key factors: performance requirements, configuration complexity, and memory constraints. Random serves as a baseline method with more parameter tuning. RandAD, combined with ADSampling for DCO, also requires more parameter tuning. ALBP and ALBP+ focus on optimizing DCO with simple configurations but moderate performance improvements. PCA and MSPCA variants deliver superior performance, but require more configurations. When memory resources are abundant, the MA variant can provide greater performance improvement at the cost of higher memory consumption. We can explore more combinations to create more different variations.

Acknowledgments

We would like to thank all the anonymous reviewers and friends in lab for their insightful comments and suggestions. The work was partially supported by the National Key Research and Development Program of China (2024YFF0617702), the National Natural Science Foundation of China (No.U22A2025, 62232007, U23A20309, 62202091), and 111 Project (No.B16009).

References

[1] 2025. *Performance analysis tools based on Linux perf_events (aka perf) and ftrace.* Retrieved July 1, 2025 from https://github.com/brendangregg/perf-tools

[2] Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore Willke. 2023. Similarity Search in the Blink of an Eye with Compressed Indices. *Proc. VLDB Endow.* 16, 11 (July 2023), 3433–3446. doi:10.14778/3611479.3611537

[3] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008), 117–122.

[4] Alexandr Andoni and Ilya Razenshteyn. 2015. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 793–801.

[5] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9, 4 (2015), 288–299.

[6] Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based Language Models and Applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, ACL 2023, Toronto, Canada, July 9-14, 2023*, Yun-Nung Vivian Chen, Margot Mieskes, and Siva Reddy (Eds.). Association for Computational Linguistics, 41–46.

[7] Martin Aumüller, Erik Bernhardsson, and Alexander John Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.* 87 (2020).

[8] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2025. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. *Proc. ACM Manag. Data* 3, 1, Article 43 (Feb. 2025), 31 pages. doi:10.1145/3709693

[9] Artem Babenko and Victor S. Lempitsky. 2015. The Inverted Multi-Index. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 6 (2015), 1247–1260.

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

[11] Erik Bernhardsson. 2021. *big-ann-benchmarks.com.* https://big-ann-benchmarks.com/

[12] Alina Beygelzimer, Sham M. Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006 (ACM International Conference Proceeding Series, Vol. 148)*, William W. Cohen and Andrew W. Moore (Eds.). ACM, 97–104.

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

[14] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.* 15, 3 (2024), 39:1–39:45.

[15] Manos Chatzakis, Yannis Papakonstantinou, and Themis Palpanas. 2025. DARTH: Declarative Recall Through Early Termination for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 4, Article 242 (Sept. 2025), 26 pages. doi:10.1145/3749160

[16] Patrick H. Chen, Wei-Cheng Chang, Jyun-Yu Jiang, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. 2023. FINGER: Fast Inference for Graph-based Approximate Nearest Neighbor Search. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 3225–3235.

[17] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 5199–5212.

[18] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld (Eds.). Morgan Kaufmann, 426–435.

[19] Benjamin Coleman, Santiago Segarra, Alexander J. Smola, and Anshumali Shrivastava. 2022. Graph Reordering for Cache-Efficient Near Neighbor Search. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/fb44a668c2d4bc984e9d6ca261262cbb-Abstract-Conference.html

[20] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells (Eds.). ACM, 191–198.

[21] Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms* 22, 1 (2003), 60–65. doi:10.1002/RSA.10073

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

[23] Arjen P. de Vries, Nikos Mamoulis, Niels Nes, and Martin L. Kersten. 2002. Efficient k-NN search on vertically decomposed data. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki (Eds.). ACM, 322–333. doi:10.1145/564691.564729

[24] Liwei Deng, Penghao Chen, Ximu Zeng, Tianfu Wang, Yan Zhao, and Kai Zheng. 2024. Efficient Data-Aware Distance Comparison Operations for High-Dimensional Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 18, 3 (Nov. 2024), 812–821. doi:10.14778/3712221.3712244

[25] Wei Dong, Moses Charikar, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar (Eds.). ACM, 577–586.

[26] Cong Fu, Changxu Wang, and Deng Cai. 2021. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4139–4150.

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

[28] Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. 2012. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 541–552.

[29] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2 (2023), 137:1–137:27.

[30] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3, Article 167 (May 2024), 27 pages. doi:10.1145/3654970

[31] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized Product Quantization for Approximate Nearest Neighbor Search. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. IEEE Computer Society, 2946–2953.

[32] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 1, Article 80 (Feb. 2025), 26 pages. doi:10.1145/3709730

[33] Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, Zhenshan Cao, Yanliang Qiao, Ting Wang, Bo Tang, and Charles Xie. 2022. Manu: A Cloud Native Vector Database Management System. *Proc. VLDB Endow.* 15, 12 (2022), 3548–3561.

[34] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, Toby Walsh (Ed.). IJCAI/AAAI, 1312–1317.

[35] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-Aware Locality-Sensitive Hashing for Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 9, 1 (2015), 1–12.

[36] Yoonho Hwang and Hee-Kap Ahn. 2011. Convergent bounds on the euclidean distance. In *Proceedings of the 24th International Conference on Neural Information Processing Systems* (Granada, Spain) *(NIPS'11)*. Curran Associates Inc., Red Hook, NY, USA, 388–396.

[37] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, Jeffrey Scott Vitter (Ed.). ACM, 604–613.

[38] Yahoo Japan. 2018. *Neighborhood Graph and Tree for Indexing High-dimensional Data.* https://github.com/yahoojapan/NGT

[39] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information*

Processing Systems 32 (2019).

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

[41] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. 2011. Searching in one billion vectors: Re-rank with source coding. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. IEEE, 861–864.

[42] William B Johnson, Joram Lindenstrauss, et al. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics* 26, 189-206 (1984), 1.

[43] Yannis Kalantidis and Yannis Avrithis. 2014. Locally Optimized Product Quantization for Approximate Nearest Neighbor Search. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2329–2336.

[44] Leonardo Kuffo, Elena Krippner, and Peter Boncz. 2025. PDX: A Data Layout for Vector Similarity Search. *Proc. ACM Manag. Data* 3, 3, Article 196 (June 2025), 26 pages. doi:10.1145/3725333

[45] Kasper Green Larsen and Jelani Nelson. 2016. The Johnson-Lindenstrauss Lemma Is Optimal for Linear Dimensionality Reduction. *Leibniz International Proceedings in Informatics* 55 (2016), 82–1.

[46] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2539–2554.

[47] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1475–1488.

[48] Kejing Lu, Mineichi Kudo, Chuan Xiao, and Yoshiharu Ishikawa. 2021. HVS: Hierarchical Graph Structure Based on Voronoi Diagrams for Solving Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 15, 2 (2021), 246–258.

[49] Kejing Lu, Chuan Xiao, and Yoshiharu Ishikawa. 2024. Probabilistic Routing for Graph-Based Approximate Nearest Neighbor Search. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=pz4B2kHVKo

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

[51] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[52] Magdalen Dobson Manohar, Zheqi Shen, Guy E. Blelloch, Laxman Dhulipala, Yan Gu, Harsha Vardhan Simhadri, and Yihan Sun. 2024. ParlayANN: Scalable and Deterministic Parallel Graph-Based Approximate Nearest Neighbor Search Algorithms. In *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, PPoPP 2024, Edinburgh, United Kingdom, March 2-6, 2024*, Michel Steuwer, I-Ting Angelina Lee, and Milind Chabbi (Eds.). ACM, 270–285.

[53] James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of Vector Database Management Systems. *CoRR* abs/2310.14021 (2023).

[54] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Vector Database Management Techniques and Systems. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago AA, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 597–604.

[55] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27.

[56] Zhen Peng, Minjia Zhang, Kai Li, Ruoming Jin, and Bin Ren. 2023. iQAN: Fast and Accurate Vector Search with Efficient Intra-Query Parallelism on Multi-Core Architectures. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, PPoPP 2023, Montreal, QC, Canada, 25 February 2023 - 1 March 2023*, Maryam Mehri Dehnavi, Milind Kulkarni, and Sriram Krishnamoorthy (Eds.). ACM, 313–328. doi:10.1145/3572848.3577527

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

[58] Jianbin Qin, Wei Wang, Chuan Xiao, Ying Zhang, and Yaoshu Wang. 2021. High-Dimensional Similarity Query Processing for Data Science. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 4062–4063.

[59] Hanan Samet. 2005. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[60] Chanop Silpa-Anan and Richard I. Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society.

[61] Yongye Su, Yinqi Sun, Minjia Zhang, and Jianguo Wang. 2024. Vexless: A Serverless Vector Data Management System Using Cloud Functions. *Proc. ACM Manag. Data* 2, 3 (2024), 187.

[62] Yifang Sun, Wei Wang, Jianbin Qin, Ying Zhang, and Xuemin Lin. 2014. SRS: Solving c-Approximate Nearest Neighbor Queries in High Dimensional Euclidean Space with a Tiny Index. *Proc. VLDB Endow.* 8, 1 (2014), 1–12. doi:10.14778/2735461.2735462

[63] Yufei Tao, Ke Yi, Cheng Sheng, and Panos Kalnis. 2009. Quality and efficiency in high dimensional nearest neighbor search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul (Eds.). ACM, 563–576.

[64] Jingdong Wang and Shipeng Li. 2012. Query-driven iterated neighborhood graph search for large scale indexing. In *Proceedings of the 20th ACM Multimedia Conference, MM '12, Nara, Japan, October 29 - November 02, 2012*, Noboru Babaguchi, Kiyoharu Aizawa, John R. Smith, Shin'ichi Satoh, Thomas Plagemann, Xian-Sheng Hua, and Rong Yan (Eds.). ACM, 179–188.

[65] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. 2016. Learning to Hash for Indexing Big Data - A Survey. *Proc. IEEE* 104, 1 (2016), 34–57.

[66] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2614–2627.

[67] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.).

[68] Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Yifan Zhu, and Wenchao Zhou. 2025. Accelerating Graph Indexing for ANNS on Modern CPUs. *Proc. ACM Manag. Data* 3, 3 (2025), 123:1–123:29. doi:10.1145/3725260

[69] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An I/O-Efficient Disk-Resident Graph Index Framework for High-Dimensional Vector Similarity Search on Data Segment. *Proc. ACM Manag. Data* 2, 1 (2024), V2mod014:1–V2mod014:27.

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

[71] Zeyu Wang, Haoran Xiong, Zhenying He, Peng Wang, and Wei Wang. 2024. Distance Comparison Operators for Approximate Nearest Neighbor Search: Exploration and Benchmark. *CoRR* abs/2403.13491 (2024).

[72] 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'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, Ashish Gupta, Oded Shmueli, and Jennifer Widom (Eds.). Morgan Kaufmann, 194–205.

[73] Shitao Xiao, Zheng Liu, Weihao Han, Jianjin Zhang, Defu Lian, Yeyun Gong, Qi Chen, Fan Yang, Hao Sun, Yingxia Shao, and Xing Xie. 2022. Distill-VQ: Learning Retrieval Oriented Vector Quantization By Distilling Knowledge from Dense Embeddings. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (Eds.). ACM, 1513–1523.

[74] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-Chain: Interactively Enhancing Large Language Models with Search for Knowledge-intensive Tasks. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 1362–1373.

[75] Mingyu Yang, Wentao Li, Jiabao Jin, Xiaoyao Zhong, Xiangyu Wang, Zhitao Shen, Wei Jia, and Wei Wang. 2025. Effective and General Distance Computation for Approximate Nearest Neighbor Search . In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 1098–1110. doi:10.1109/ICDE65448.2025.00087

[76] Wen Yang, Tao Li, Gai Fang, and Hong Wei. 2020. PASE: PostgreSQL Ultra-High-Dimensional Approximate Nearest Neighbor Search Extension. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan,

Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2241–2253.

[77] Peter N. Yianilos. 1993. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, Vijaya Ramachandran (Ed.). ACM/SIAM, 311–321.

[78] Longhui Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. 2024. A Two-Stage Adaptation of Large Language Models for Text Ranking. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, 11880–11891. doi:10.18653/V1/2024.FINDINGS-ACL.706

[79] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. 2023. VBASE: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In *17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023, Boston, MA, USA, July 10-12, 2023*, Roxana Geambasu and Ed Nightingale (Eds.). USENIX Association, 377–395.

[80] Yunan Zhang, Shige Liu, and Jianguo Wang. 2024. Are there fundamental limitations in supporting vector data management in relational databases? A case study of PostgreSQL. In *International Conference on Data Engineering (ICDE)*.

[81] Bolong Zheng, Ziyang Yue, Qi Hu, Xiaomeng Yi, Xiaofan Luan, Charles Xie, Xiaofang Zhou, and Christian S. Jensen. 2023. Learned Probing Cardinality Estimation for High-Dimensional Approximate NN Search. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 3209–3221.

[82] Bolong Zheng, Xi Zhao, Lianggui Weng, Quoc Viet Hung Nguyen, Hang Liu, and Christian S. Jensen. 2022. PM-LSH: a fast and accurate in-memory framework for high-dimensional approximate NN and closest pair search. *VLDB J.* 31, 6 (2022), 1339–1363. doi:10.1007/S00778-021-00680-7

[83] Yuxin Zheng, Qi Guo, Anthony K. H. Tung, and Sai Wu. 2016. LazyLSH: Approximate Nearest Neighbor Search for Multiple Distance Functions with a Single Index. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 2023–2037.

[84] Yifan Zhu, Lu Chen, Yunjun Gao, and Christian S. Jensen. 2021. Pivot selection algorithms in metric spaces: a survey and experimental study. *The VLDB Journal* 31, 1 (Aug. 2021), 23–47. doi:10.1007/s00778-021-00691-4

Received April 2025; revised July 2025; accepted August 2025