TaCo: Data-adaptive and Query-aware Subspace Collision for High-dimensional Approximate Nearest Neighbor Search

JIUQI WEI, Oceanbase, Ant Group, China
ZHENYU LIAO, Huazhong University of Science and Technology, China
RUOYU HAN, Institute of Computing Technology, Chinese Academy of Sciences, China
QUANQING XU, Oceanbase, Ant Group, China
CHUANHUI YANG, Oceanbase, Ant Group, China
THEMIS PALPANAS, LIPADE, Université Paris Cité, France

Approximate Nearest Neighbor Search (ANNS) in high-dimensional Euclidean spaces is a fundamental problem with broad applications. *Subspace Collision* is a newly proposed ANNS framework that provides a novel paradigm for similarity search and achieves superior indexing and query performance. However, the subspace collision framework remains data-agnostic and query-oblivious, resulting in imbalanced index construction and wasted query overhead. In this paper, we address these limitations from two aspects: first, we design a subspace-oriented data transformation mechanism by averaging the entropies computed over each subspace of the transformed data, which ensures balanced subspace partitioning (in an information theoretical sense) and enables *data-adaptive* subspace collision; second, we present *query-aware* and scalable query strategies that dynamically allocate overhead for each query and accelerate collision probing within subspaces. Building on these ideas, we propose a novel data-adap**t**ive and query-**a**ware subspa**c**e c**o**llision method, abbreviated as TaCo, which achieves efficient and accurate ANN search while maintaining an excellent balance between indexing and query performance. Extensive experiments on real-world datasets demonstrate that, when compared to state-of-the-art subspace collision methods, TaCo achieves up to **8**× speedup in indexing and reduces to **0.6**× memory footprint, while achieving over **1.5**× query throughput. Moreover, TaCo achieves state-of-the-art indexing performance and provides an effective balance between indexing and query efficiency, even when compared with advanced methods beyond the subspace-collision paradigm.

CCS Concepts: • **Information systems** → **Nearest-neighbor search**; **Query optimization**.

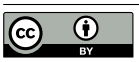
Additional Key Words and Phrases: Subspace collision, ANN search, High-dimensional spaces

ACM Reference Format:
Jiuqi Wei, Zhenyu Liao, Ruoyu Han, Quanqing Xu, Chuanhui Yang, and Themis Palpanas. 2026. TaCo: Data-adaptive and Query-aware Subspace Collision for High-dimensional Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 241 (June 2026), 28 pages. https://doi.org/10.1145/3802118

1 Introduction

Background and Problem. Nearest Neighbor Search (NNS) on high-dimensional vectors is a fundamental problem for numerous applications, such as recommendation systems [66], information retrieval [40], and data mining [69]. However, NNS in high-dimensional spaces is challenging due

*Chuanhui Yang is the corresponding author.
Authors' Contact Information: Jiuqi Wei, Oceanbase, Ant Group, China, weijiuqi.wjq@antgroup.com; Zhenyu Liao, Huazhong University of Science and Technology, China, zhenyu_liao@hust.edu.cn; Ruoyu Han, Institute of Computing Technology, Chinese Academy of Sciences, China, hanruoyu23z@ict.ac.cn; Quanqing Xu, Oceanbase, Ant Group, China, xuquanqing.xqq@oceanbase.com; Chuanhui Yang, Oceanbase, Ant Group, China, rizhao.ych@oceanbase.com; Themis Palpanas, LIPADE, Université Paris Cité, France, themis@mi.parisdescartes.fr.



ACM 2836-6573/2026/6-ART241
https://doi.org/10.1145/3802118

to the *curse of dimensionality* phenomenon [9, 15, 35, 51, 52, 54]. In practice, Approximate Nearest Neighbor Search (ANNS) is often used as an alternative approach, achieving huge efficiency gains by sacrificing some query accuracy [49, 70, 79]. With the development of Large Language Models (LLMs) [98], ANNS has received a lot of attention. For example, Retrieval Augmented Generation (RAG) leverages ANNS to add context to an LLM query [21, 29, 45], and Key-Value Cache (KVCache) adopts ANNS to accelerate the model inference process [17, 53, 97]. Depending on the storage medium of indexes and datasets, ANNS methods can be divided into in-memory methods [28, 30, 55, 89] and memory-disk hybrid methods [13, 36, 75]. In this paper, we focus on in-memory methods due to their widespread application.

Prior Work. Numerous excellent ANNS methods have been developed, falling into four categories: locality-sensitive hashing (LSH)-based methods [2, 44, 71, 87, 89, 100], vector quantization (VQ)-based methods [6, 27, 28, 30, 37, 58], tree-based methods [8, 42, 65, 77, 78, 81], and graph-based methods [4, 24, 33, 55, 72, 99]. Although each category of methods exhibits distinct strengths, they commonly face inherent limitations that restrict their ability to simultaneously perform well in index construction and query answering [49]. *Subspace Collision* is a newly proposed ANNS framework [86]. Within the subspace collision framework, the original high-dimensional space is first partitioned into N_s subspaces. Subsequently, collisions between the query point and data points are counted within each subspace. A data point is considered to *collide* with a query in a subspace if it ranks among the points with the closest distance (within that subspace) to that query. For any data point, its number of collisions in all subspaces is defined as the *SC-score* metric, an integer value ranging from 0 to N_s. Empirical analysis in various datasets shows that the SC-score metric adheres to the *Pareto principle* (also known as the 80-20 rule), which makes it an efficient proxy for the Euclidean distance between the query point and data points. *SuCo* [86] is the first approach designed for the subspace collision framework. The corresponding experiments demonstrate that SuCo achieves state-of-the-art performance.

Limitations and Motivation. Although the subspace collision framework fits the ANNS tasks well and SuCo shows superior performance, we identify some limitations that need to be fixed. (1) The subspace collision framework employs a naive, *data-agnostic* subspace partitioning mechanism: uniformly dividing the original d-dimensional space into N_s fixed-size subspaces of $s = \lfloor \frac{d}{N_s} \rfloor$ dimensions. This mechanism disregards inherent data distributions, resulting in subspaces with imbalanced statistical properties. However, the SC-score metric still assigns equal weight to all subspaces during computation despite this imbalance. Consequently, designing a *data-adaptive* subspace partitioning mechanism becomes essential to mitigate errors arising from data skew across subspaces. (2) The query strategies of the subspace collision framework are *query-oblivious*, employing the same query overhead for all queries. However, empirical studies confirm significant performance variance across queries in ANNS tasks [46, 80]. This rule also applies to the subspace collision framework. For instance, some queries demonstrate highly discriminative SC-score distributions, thus high recall can be achieved by re-ranking only the top-scoring data points. In contrast, other queries exhibit poor SC-score discriminability, requiring larger candidate sets during re-ranking to achieve comparable recall. Consequently, it is necessary to design *query-aware* query answering strategies for the subspace collision framework to dynamically adjust to the query overhead. (3) Although SuCo's inverted multi-index (IMI) demonstrates competitive performance, the dynamic activation algorithm designed for IMI is not scalable enough. By maintaining the activation list via a linear array, the *Dynamic Activation* algorithm [86] incurs linear query complexity. While the linear design performs satisfactorily with a small length of the IMI list, it becomes an efficiency bottleneck when a large list length is required for higher indexing precision. Thus, designing a more *scalable* algorithm for IMI index structure becomes essential.

Our Method. In this paper, we propose a *data-adaptive* and *query-aware* subspace collision framework for ANNS and design a novel method, named TaCo, which exhibits superior performance in both index construction and query processing compared to state-of-the-art ANNS methods. First, we analyze the data-agnostic issue of the original subspace collision framework, which fails to account for inherent data distributions, resulting in subspaces with imbalanced statistical properties. To address this issue, we formulate an *entropy-averaging optimization problem* and show that it can be solved efficiently, which leads to the design of a *subspace-oriented data transformation* mechanism. This transformation makes the subspace collision framework *data-adaptive* and enables more accurate subspace partitioning. Furthermore, the transformation reduces dimensionality by 40% to 96% across different datasets, leading to significant improvements in both indexing and query efficiency. Second, we analyze the query-oblivious issue of the original subspace collision framework, i.e., using the same strategy for all queries results in wasted query overhead. To address this issue, we design a *query-aware candidate selection* mechanism that leverages the distribution of SC-scores (an intermediate query state) to *dynamically* adjust the query overhead for each individual query. This mechanism makes the subspace collision framework *query-aware* and improves the query efficiency. Third, we analyze the scalability issue of the query algorithm for inverted multi-index (IMI) structure, revealing that its performance under fine-grained indexes is constrained by the linear query complexity. To address this issue, we design a *scalable dynamic activation* algorithm that employs a min-heap structure to reduce the query complexity to $O(1)$. This advancement supports efficient querying even with highly fine-grained indexes. Fourth, integrating the above designs, we propose a data-adap**t**ive and query-**a**ware subspa**c**e c**o**llision method, abbreviated as TaCo. To systematically evaluate the proposed optimization techniques, we also present three ablation methods based on SuCo, where SuCo [86] is the state-of-the-art subspace collision-based method. Fifth, we conduct extensive experiments on real-world datasets. Empirical results show that when compared with subspace collision-based methods, TaCo achieves up to **8**× speedups in indexing while consuming only **0.6**× memory. In terms of query performance, TaCo achieves over **1.5**× query throughput at equivalent high recall. Moreover, TaCo achieves state-of-the-art indexing performance and provides an effective balance between indexing and query efficiency, even when compared with advanced methods beyond the subspace-collision paradigm.

Our main contributions are summarized as follows.

- We design a subspace-oriented data transformation mechanism by formulating an entropy-averaging optimization problem and showing that it can be solved efficiently. This approach achieves balanced subspace partitioning with impressive dimensionality reduction, thereby enhancing the subspace collision framework with data adaptability.
- We present query-aware and scalable query strategies that dynamically adjusts the overhead for each individual query and accelerate the collision probing within each subspace, thereby enhancing the subspace collision framework with query awareness and scalability.
- We propose a data-adaptive and query-aware subspace collision method named TaCo, which supports efficient and accurate ANN search while maintaining the excellent balance between indexing and query performance inherent to subspace collision-based methods.
- We conduct extensive experiments demonstrating that when compared to state-of-the-art subspace collision methods, TaCo achieves up to **8**× speedup in indexing and reduces to **0.6**× memory footprint, while achieving over **1.5**× query throughput. Moreover, TaCo achieves state-of-the-art indexing performance and provides an effective balance between indexing and query efficiency, even when compared with advanced methods beyond the subspace-collision paradigm.

Table 1. Notations

Notation	Description
$\mathbb{R}^d$	d-dimensional Euclidean space
$\mathcal{D}$	Dataset of points in $\mathbb{R}^d$
n	Dataset cardinality $\lvert\mathcal{D}\rvert$
o, q	A data point in $\mathcal{D}$ and a query point in $\mathbb{R}^d$
o_i^*	The i-th nearest data point to q in $\mathcal{D}$
o_i	The i-th data point in $\mathcal{D}$
$\lVert o_1, o_2\rVert$	The Euclidean distance between o_1 and o_2
o', q'	o and q in a subspace
N_s	Number of subspaces
s	Dimension of each subspace
$S_i, \mathcal{D}_i$	The i-th subspace and all data points in S_i
o_j^i, q^i	The j-th data point and q in the i-th subspace
α	Collision ratio
β	Re-rank ratio
K, t	Number of K-means clusters and iterations

2 Preliminaries

2.1 Problem Definition

Definition 1 (Nearest Neighbor Search, NNS)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional Euclidean space $\mathbb{R}^d$ and a query $q \in \mathbb{R}^d$. An NNS returns a point $o^* \in \mathcal{D}$ which has the minimum Euclidean distance to q among all points in $\mathcal{D}$, i.e., for each $o \in D$ satisfying $\lVert q, o^*\rVert \leq \lVert q, o\rVert$.

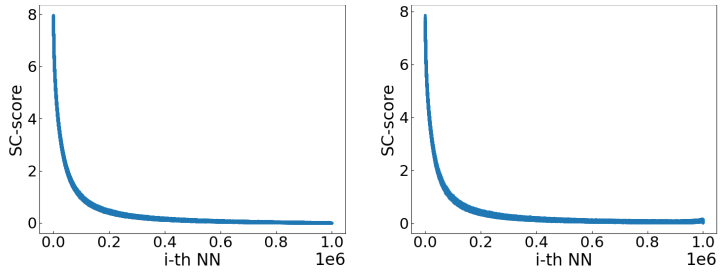
Definition 2 (k-Nearest Neighbor Search, k-NNS)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional Euclidean space $\mathbb{R}^d$, a query $q \in \mathbb{R}^d$, and an integer k. Let o_i^* be the i-th exact nearest neighbor of q in $\mathcal{D}$, and $\mathcal{B} = \{o_1^*, o_2^*, \ldots, o_k^*\}$. A k-NNS returns k points $\mathcal{R} = \{o_1, o_2, \ldots, o_k\}$, satisfying $\mathcal{R} = \mathcal{B}$.

Definition 3 (k-Approximate Nearest Neighbor Search, k-ANNS)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional Euclidean space $\mathbb{R}^d$, a query $q \in \mathbb{R}^d$, an approximation ratio $c > 1$, and an integer k. Let o_i^* be the i-th exact nearest neighbor of q in $\mathcal{D}$. A k-ANNS returns k points $o_1, o_2, \ldots, o_k$. For each $o_i \in D$ satisfying $\lVert q, o_i\rVert \leq c \cdot \left\lVert q, o_i^*\right\rVert$, where $i \in [1, k]$.

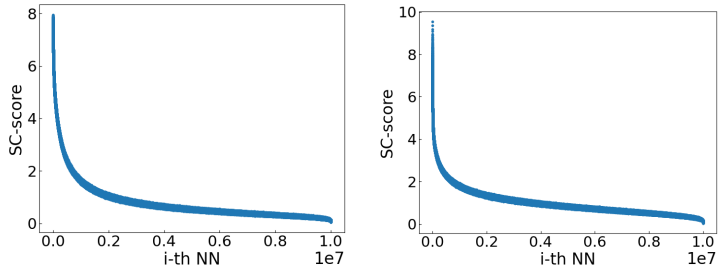
In practice, many k-ANNS implementations do not explicitly rely on an approximation ratio c during query processing; instead, they constrain the quality of the results through evaluation metrics [55, 86]. Let $\mathcal{R}^* = \{o_1^*, o_2^*, \ldots, o_k^*\}$ be the ground truth set of exact k nearest neighbors for query q in dataset $\mathcal{D}$, and $\mathcal{R} = \{o_1, o_2, \ldots, o_k\}$ represent the returned result set. k-ANNS seeks to maximize $recall = \frac{\lvert\mathcal{R} \cap \mathcal{R}^*\rvert}{k}$ to obtain higher quality results.

2.2 Subspace Collision Framework

Subspace Collision is a newly proposed ANNS framework [86]. Specifically, subspace collision is inspired by an intuition: two data points that are close in the high-dimensional original space are also more likely to be close in a common subspace. However, the distances between data points are not uniformly distributed across dimensions, which can cause the above intuition to fail. To overcome this problem, the authors proposed the definitions of *subspace sampling*, *subspace collision*, and *SC-score*.



(a) DEEP1M: N_s=8, α=0.05 (b) GIST1M: N_s=8, α=0.05



(c) Yandex DEEP10M: N_s=8, α=0.1 (d) SPACEV10M: N_s=10, α=0.1

Fig. 1. "Pareto principle" of SC-score.

Definition 4 (Subspace Sampling)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional space, we adopt a multi-round sampling strategy to obtain N_s subspaces. In round i, a number of $s = \lfloor \frac{d}{N_s} \rfloor$ dimensions are uniformly sampled without replacement to form a subspace S_i, $i = 1, 2, \ldots, N_s - 1$. For the last subspace S_{N_s}, it simply pick up all remaining dimensions.

Definition 5 (Subspace Collision)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional space, a query $q \in \mathbb{R}^d$, and a collision ratio $\alpha \in (0, 1)$. We randomly select s dimensions from all d dimensions as a subspace $\mathbb{R}^s$ ($s < d$). The dataset $\mathcal{D}$, the data point o, and the query point q in this subspace are denoted as $\mathcal{D}'$, o', and q'. If a point $o \in \mathcal{D}$ satisfies: o' is one of the $(\alpha \cdot n)$-NNs of q' in $\mathcal{D}'$, we say that o collides with q in the subspace $\mathbb{R}^s$.

Definition 6 (SC-score)**.** Given a dataset $\mathcal{D}$ of n data points in d-dimensional space, a query $q \in \mathbb{R}^d$, N_s s-dimensional subspaces, a collision ratio $\alpha \in (0, 1)$. Probe collisions of q in N_s subspaces with the collision ratio α. For a data point $o \in \mathcal{D}$, its SC-score is the number of subspaces where it collides with q. Therefore, SC-score is an integer in $[0, N_s]$.

Subspace sampling (Def. 4) provides a strategy for subspace partitioning. In practice, for convenience, the d-dimensional high-dimensional space is typically partitioned uniformly into N_s subspaces with equal dimensions $s = \lfloor \frac{d}{N_s} \rfloor$. Subspace collision (Def. 5) quantifies the similarity between data points and the query point within a single subspace, mitigating the influence of intra-subspace distance rankings on final results. Conversely, the SC-score (Def. 6) integrates collision outcomes across multiple subspaces to alleviate significant errors potentially introduced by any single subspace.

An exciting observation is that SC-score follows the "Pareto principle" (also known as the 80-20 rule), which makes it an efficient proxy for the Euclidean distance. Specifically, as shown in Figure 1,

the top 20% of data points nearest to the query exhibit discriminatively high SC-scores, while the remaining 80% of the data points have low SC-scores with poor discriminability. This phenomenon enables us to selectively process high-scoring data points to obtain high-recall ANNS results with minimal computational overhead.

2.3 SC-Linear and SuCo Methods

SC-Linear [86] is a baseline ANNS method (without index structure) designed for the subspace collision framework. The core logic of the algorithm comprises three sequential phases: (1) Collision counting: for each subspace, probing colliding data points with the query based on actual Euclidean distances; (2) Candidate selection: select a set of high-quality candidates by sorting data points according to SC-score in descending order; (3) Result refinement: re-rank all candidate points in the original space and return the top-k results. Experimental results demonstrate SC-Linear achieves over 0.99 recall on benchmark datasets, validating the subspace collision framework's efficacy for high-accuracy ANNS.

SuCo [86] is the state-of-the-art ANNS method developed upon the subspace collision framework. Compared with SC-Linear, SuCo further solves the problem of "*how to count collisions in each subspace as quickly and accurately as possible*". Specifically, SuCo adopts the *inverted multi-index (IMI)* [6] as the index structure, and designs the *Dynamic Activation Algorithm* for IMI to achieve efficient query. Experimental results confirm that SuCo achieves significantly higher efficiency than SC-Linear with an acceptable sacrifice of accuracy. SuCo also achieves superior performance in both index construction and query answering compared to state-of-the-art ANNS methods, such as HNSW [55], OPQ [30], and DET-LSH [89].

3 Data-adaptive Subspace Collision

3.1 Problem Analysis

As introduced in Section 2.2, the subspace collision framework adopts a naive, *data-agnostic* subspace partitioning mechanism that uniformly divides the original d-dimensional space into N_s fixed-size subspaces of $s = \lfloor \frac{d}{N_s} \rfloor$ dimensions. This mechanism ignores the inherent data distribution and assigns equal weight to all subspaces. However, due to the data skew across subspaces, the significance of subspaces is imbalanced in practice. Therefore, it is necessary to design a *data-adaptive* subspace partitioning mechanism to mitigate errors arising from data skew across subspaces.

Based on characteristic analysis of the subspace collision framework, two potential technical routes may support data-adaptive subspace partitioning: (1) first partition the subspace, and then assign different weights to each subspace according to underlying data distributions; or (2) first transform data points according to underlying data distribution, and then partition them into subspaces.

For the first technical pathway, we explored various weighting schemes derived from data dispersion measures, including variance, average distance, and Local Intrinsic Dimensionality (LID) [38, 41]. However, experimental attempts did not bring about query performance improvements, and even resulted in significant performance loss in some configurations. We attribute the performance degradation mainly to two factors. First, data dispersion measures have difficulty in capturing underlying feature correlations, and intra-subspace measurements discard inter-subspace correlation information. Second, weighted subspaces convert SC-score from an integer to a real-valued metric, losing the original performance advantage when counting collisions and selecting candidates.

In this paper, we adopt the second technical route: performing subspace-oriented data transformation before subspace partitioning. This route enables comprehensive feature integration

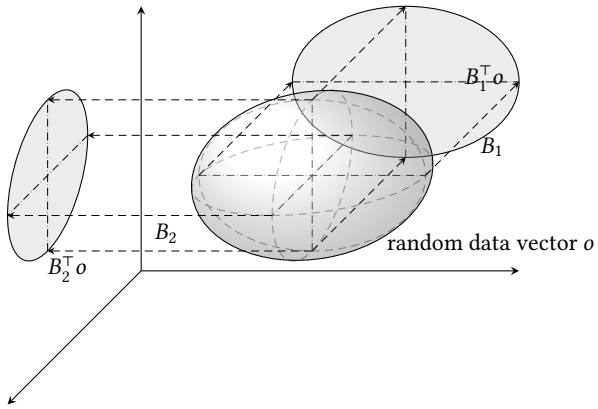


Fig. 2. Illustration of the proposed subspace-oriented entropy averaging approach, with random data vector o of mean zero and covariance Σ represented by a 3D ellipsoid.

across all dimensions during the data transformation, while preserving integer-based SC-score computation, thereby achieving an effective balance between accuracy and efficiency. The following section presents our subspace-oriented data transformation methodology for realizing data-adaptive subspace collision.

3.2 Subspace-oriented Data Transformation via Entropy Averaging

In this section, we describe the motivation and a few theoretical properties of the proposed Subspace-oriented Data Transformation method in Algorithms 1 and 2.

As motivated above in Section 3.1, here we aim to design a *data-adaptive* subspace collision framework that takes account of the *heterogeneity* in the (different subspaces of the) data. To model data having different variability in its coordinates, we consider that the input data o are of mean zero and covariance Σ (that is assumed positive definite and can be *different* from the identity matrix). Note in particular that we do *not* need to assume the data distribution (e.g., that they are Gaussian), but that the covariance exists. We aim to find a linear transformation $B \in \mathbb{R}^{d\times(N_s\cdot s)}$ that "projects" the input o into N_s subspaces of dimension s each, in a *data-adaptive* way described above.

In this paper, we adopt *differential entropy* [1] from information theory to measure the distributional differences of data in different subspaces. It is commonly used to quantify the uncertainty or information content of (the distribution of) random data. Precisely, for input data $o \in \mathbb{R}^d$, consider partitioning the linear transformation B into N_s blocks as follow:

$$B = \begin{bmatrix} B_1 & B_2 & \dots & B_{N_s} \end{bmatrix},\ B_j \in \mathbb{R}^{d\times s},\ j \in \{1, \dots, N_s\}. \tag{1}$$

Then, the j^{th} subspace after transformation is $B_j^\top o \in \mathbb{R}^s$. If the obtained N_s subspaces after transformation have approximately equal differential entropy, it signifies that the data information is equally spread over all subspaces (quantitatively measured by the area/volume of the "information ellipsoids" on the projected subspaces, as shown in Figure 2).

Since the data vector o is of mean zero and covariance Σ, each its subspace projection $B_j^\top o$ is of mean zero and covariance $B_j^\top \Sigma R_j$. Our goal is to find a linear transformation B such that, after transformation, the obtained subspaces $B_j^\top o$ are "uniform" in the sense of having balanced

Algorithm 1: Subspace-oriented Data Transformation

Input: Dataset $\mathcal{D} = \{o_1, o_2, \ldots, o_n\}$, dataset size n, data dimensionality d, subspace number N_s, subspace dimensionality s

Output: Transformed dataset $\mathcal{T}$

Initialize the transformed dataset $\mathcal{T}$ with size $(n, N_s \cdot s)$;

Compute the mean value of $\mathcal{D}$: $\bar{o} = \frac{1}{n}\sum_{i=1}^{n} o_i$;

Compute the covariance matrix of all points: $\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(o_i - \bar{o})(o_i - \bar{o})^\top$;

Perform the spectral decomposition of $\hat{\Sigma}$, obtain the eigenvectors $\mathcal{E} = \{\xi_1, \xi_2, \ldots, \xi_d\}$ and their corresponding eigenvalues $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_d\}$;

$B \leftarrow$ **call** *Eigensystem Allocation*$(\mathcal{E}, \Lambda, d, N_s, s)$;

for $i = 1$ *to* n **do**

  for $j = 1$ *to* N_s **do**

    Obtain the eigenvectors assigned to the j-th subspace: $B_j = \{\xi_1^j, \xi_2^j, \ldots, \xi_s^j\}$;

    For any $o_i \in \mathbb{R}^d$, its transformed representation at the j-th subspace is: $r_i^j \leftarrow B_j^\top(o_i - \bar{o}) = ((\xi_1^j)^\top(o_i - \bar{o}), (\xi_2^j)^\top(o_i - \bar{o}), \ldots, (\xi_s^j)^\top(o_i - \bar{o})) \in \mathbb{R}^s$;

  Concatenate the transformed representations of o_i in N_s subspaces $(r_i^1, r_i^2, \ldots, r_i^{N_s})$ to obtain the transformed data point $o_i' \in \mathbb{R}^{N_s \cdot s}$;

  $\mathcal{T}_i \leftarrow o_i'$;

return the transformed dataset $\mathcal{T}$;

differential entropies. This leads to the following constrained min-max optimization problem

$$\begin{aligned} \min_j \max_{B_j} \quad & h(B_j^\top o), \\ \text{s.t.} \quad & B_j^\top B_j = I_s, \end{aligned} \tag{2}$$

where $h(B_j^\top o)$ is the differential entropy of $B_j^\top o$ that is continuously distributed. It is known that for any random vector with given mean and covariance, the multi-variate Gaussian distribution maximizes the differential entropy [1], with

$$h(B_j^\top o) \propto \log\det(B_j^\top \Sigma B_j), \quad \text{for} \quad B_j^\top o \sim \mathcal{N}(0, \Sigma), \tag{3}$$

that is proportional to the log-determinant of the covariance $B_j^\top \Sigma B_j$, provided that B_j has full rank.

As a consequence, the optimization problem in Equation (2) can be relaxed into the following min-max problem:

$$\begin{aligned} \min_j \max_{B_j} \quad & \log\det(B_j^\top \hat{\Sigma} B_j), \\ \text{s.t.} \quad & B_j^\top B_j = I_s, \end{aligned} \tag{4}$$

where we use the sample covariance $\hat{\Sigma}$ instead of its population counterpart Σ, and impose the orthonormal constraint so that each block transformation preserves scale: the data covariance is neither artificially amplified nor diminished within each subspace.

The min-max optimization problem in (4) can be effectively solved using the Eigensystem Allocation method described in Algorithm 2, per the following result.

Theorem 1 (Performance Guarantee for Algorithm 2)**.** *Assume that the data sample covariance* $\hat{\Sigma}$ *has distinct eigenvalues that are all greater than or equal to one. Then, the Eigensystem Allocation method in Algorithm 2 solves the optimization problem in Equation* (4)*.*

Algorithm 2: Eigensystem Allocation

Input: Eigenvectors $\mathcal{E} = \{\xi_1, \xi_2, \ldots, \xi_d\}$ and eigenvalues $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ of $\hat{\Sigma}$, data dimensionality d, subspace number N_s, subspace dimensionality s

Output: A set of buckets holding eigenvectors assigned to each subspace $B = \{B_1, B_2, \ldots, B_{N_s}\}$

Initialize a set of N_s empty buckets $B = \{B_1, B_2, \ldots, B_{N_s}\}$ to hold the eigenvectors assigned to each subspace;

Initialize N_s values $p_1 = p_2 = \ldots = p_{N_s} = 1$ to hold the product of the eigenvalues assigned to each subspace;

Scale the eigenvalues proportionally such that $\forall \lambda_i \in \Lambda,\ \lambda_i \geq 1$;

Reorder the eigenvalues and eigenvectors in descending order: $\mathcal{E}' = \{\xi'_1, \xi'_2, \ldots, \xi'_d\}$, $\Lambda' = \{\lambda'_1, \lambda'_2, \ldots, \lambda'_d\}$, such that $\lambda'_1 \geq \lambda'_2 \geq \ldots \geq \lambda'_d \geq 1$;

for $i = 1$ *to* $N_s \cdot s$ **do**

  $bucket \leftarrow \underset{j \in \{1,2,\ldots,N_s\},\ |B_j| < s}{\operatorname{argmin}} p_j$;

  $B_{bucket} \leftarrow B_{bucket} \cup \{\xi'_i\}$;

  $p_{bucket} \leftarrow p_{bucket} \cdot \lambda'_i$;

return the eigenvectors allocation set B;

Proof of Theorem 1. To show that Eigensystem Allocation in Algorithm 2 solves the optimization problem in (4), we first focus on the inner maximization problem: Denote $\lambda_1 > \ldots > \lambda_d$ and $\mu_1 \geq \ldots \geq \mu_s$ the eigenvalues of $\hat{\Sigma}$ and $B_j^\top \hat{\Sigma} B_j$, respectively (in descending order), we have, by the Poincaré separation theorem (also known as general Cauchy interlacing theorem) that

$$\lambda_i \geq \mu_i \geq \lambda_{d-s+i}, \quad i = 1, \ldots, s, \tag{5}$$

that is, each eigenvalue μ_i of $B_j^\top \hat{\Sigma} B_j$ lies between two eigenvalues of $\hat{\Sigma}$. Since the logarithm function is monotonically increasing, maximizing the (log-)determinant in (4) is equivalent to maximizing the product of eigenvalues. We formally have, by (5), that

$$\max_{B_j^\top B_j = I_s} \det(B_j^\top \hat{\Sigma} B_j) = \prod_{i=1}^{s} \lambda_i, \tag{6}$$

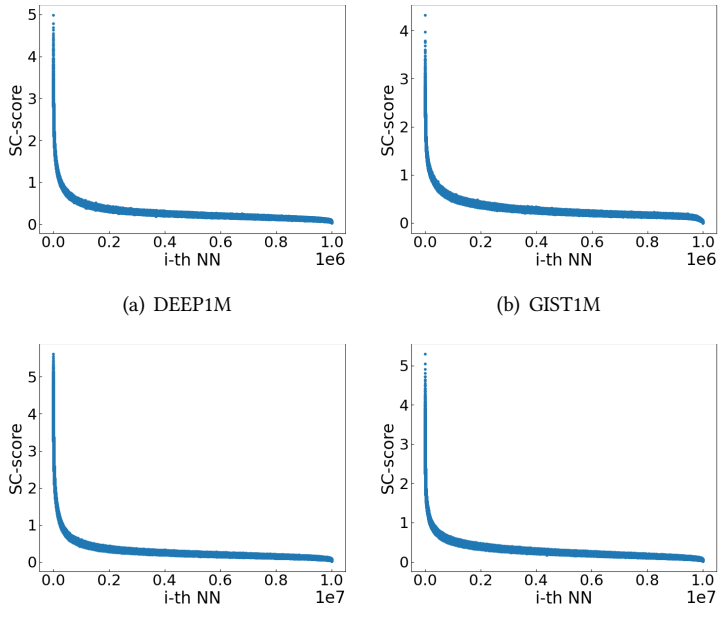
and the maximum is achieved by taking B_j to be the associated eigenvectors, i.e., $B_j = [\xi_1, \ldots, \xi_s]$.

Further note that the greedy procedure in Algorithm 2 finds the optimal "balanced" partition $B_1, \ldots, B_{N_s}$ that solves the outer minimization problem of (4). This concludes the proof. □

3.3 SC-score Distribution after Transformation

In this section, we explore the neighborhood relationship of the data after the subspace-oriented transformation. We first analyze the SC-score distributions of the transformed data on four datasets under a unified experimental setting, with $N_s = 6$, $s = 8$, and $\alpha = 0.05$. Figure 3 shows that the SC-score of the transformed data still follows the *Pareto principle*, exhibiting the same statistical behavior as the original (untransformed) data shown in Figure 1. Experimental results indicate that, the transformed data preserves the good neighborhood relationships of the original data.

To investigate the underlying mechanism behind this phenomenon, in the remainder of this section, we perform rigorous theoretical analyses showing that the transformation introduced



(a) DEEP1M (b) GIST1M

(c) Yandex DEEP10M (d) Microsoft SPACEV10M

Fig. 3. SC-score of the transformed data still follows the Pareto principle as the original data in Figure 1.

in Algorithm 2 preserves (pairwise) local distances (Lemma 1), and consequently the relative neighborhood ordering (Theorem 2).

Lemma 1 (Local Distance Preservation for Algorithm 2). *Let $\mathcal{D} = \{o_i\}_{i=1}^n \in \mathbb{R}^d$ be a dataset, and let $B \in \mathbb{R}^{d\times(N_s\cdot s)}$ denote the transformation returned by Algorithm 2, i.e., $B = \begin{bmatrix} B_1 & B_2 & \dots & B_{N_s} \end{bmatrix}$, with $B_j \in \mathbb{R}^{d\times s}$ and $N_s \cdot s \le d$. For any point $o_i \in \mathcal{D}$ and its neighbor o_j, assume that the local difference vector satisfies*

$$\|(I_d - BB^\top)(o_i - o_j)\|^2 \le \varepsilon \|o_i - o_j\|^2, \tag{7}$$

for some $\varepsilon \in (0,1)$. Then, the distance after transformation satisfies[1]

$$(1-\varepsilon)\|o_i - o_j\|^2 \le \|B^\top(o_i - o_j)\|^2 \le \|o_i - o_j\|^2. \tag{8}$$

Proof of Lemma 1. Recall from Algorithm 2 that $B \in \mathbb{R}^{d\times(N_s\cdot s)}$ contains eigenvectors of $\hat{\Sigma}$ as its columns, so that $BB^\top$ is an orthogonal projection matrix. As a consequence, $o_i - o_j \in \mathbb{R}^d$ can be decomposed as the sum of the projection $BB^\top(o_i - o_j)$ and the corresponding residue $(I_d - BB^\top)(o_i - o_j)$, with

$$\|o_i - o_j\|^2 = \|B^\top(o_i - o_j)\|^2 + \|(I_d - BB^\top)(o_i - o_j)\|^2. \tag{9}$$

Rearranging (9) gives

$$\|B^\top(o_i - o_j)\|^2 = \|o_i - o_j\|^2 - \|(I_d - BB^\top)(o_i - o_j)\|^2. \tag{10}$$

By (7), the second term on the right-hand side is bounded by $\varepsilon\|o_i - o_j\|^2$, which yields the lower bound in (8). The upper bound follows directly from the non-expansiveness of orthogonal projection. □

[1]The assumption of the form in (7) has been established for random data points drawn from the so-called spiked random matrix model, see [15, 39, 64].

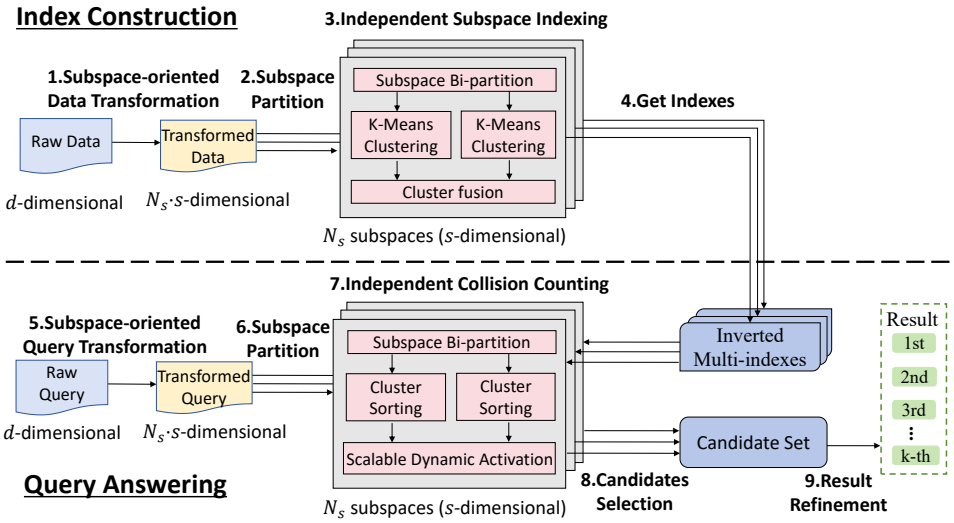


Fig. 4. Overview of the TaCo workflow.

With Lemma 1 at hand, we show next in Theorem 2 that Algorithm 1 is guaranteed to preserve relative neighborhood ordering under the conditions of Lemma 1.

Theorem 2 (Relative Neighborhood Ordering Preservation for Algorithm 1)**.** *Under the notations and conditions of Lemma 1, consider two points $o_j, o_z \in \mathcal{D}$ such that*

$$\|o_i - o_j\|^2 < (1-\varepsilon)\,\|o_i - o_z\|^2. \tag{11}$$

Then, their relative ordering with respect to o_i is preserved after the transformation through B, in the sense that

$$\|B^\top(o_i - o_j)\| < \|B^\top(o_i - o_z)\|. \tag{12}$$

Proof of Theorem 2. It follows from Lemma 1 that

$$\|B^\top(o_i - o_j)\|^2 \le \|o_i - o_j\|^2 \quad \text{and} \quad \|B^\top(o_i - o_z)\|^2 \ge (1-\varepsilon)\,\|o_i - o_z\|^2.$$

Combining these two inequalities with (11) yields

$$\|B^\top(o_i - o_j)\|^2 < \|B^\top(o_i - o_z)\|^2.$$

This concludes the proof of Theorem 2. □

4 The TaCo Method

In this section, we describe a data-adaptive and query-aware subspace collision method called TaCo. Figure 4 illustrates the TaCo workflow. During index construction, subspace-oriented data transformation is first applied, reducing data dimensionality from d to $N_s \cdot s$. The transformed data is then partitioned into N_s subspaces, and an inverted multi-index (IMI) is constructed in each subspace. During query answering, the query undergoes transformation consistent with the index construction and is subsequently partitioned into N_s subspaces. The novel *Scalable Dynamic Activation* algorithm is then executed on each subspace's IMI to enable efficient collision counting. Next, the *Query-aware Candidates Selection* algorithm identifies candidate points based on the collision distribution of each query. The final step involves refining the candidate set and returning the top-k results.

Algorithm 3: Create Index

Input: Dataset $\mathcal{D}$, dataset size n, data dimensionality d, subspace number N_s, subspace dimensionality s, number of K-means clusters K, K-means iterations t

Output: Centroid list *centroids* and inverted multi-index list *IMIs*

1 $\mathcal{T} \leftarrow$ **call** *Subspace-oriented Data Transformation*$(\mathcal{D}, n, d, N_s, s)$;
2 Divide the transformed dataset $\mathcal{T}$ into N_s subspaces: $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_s}$, each with s dimensions;
3 Initialize a list *centroids* of length $2N_s$ and a list *IMIs* of length N_s;
4 **for** $i = 1$ *to* N_s **do**
5 Initialize a map IMI_i as the inverted multi-index;
6 Further split $\mathcal{T}_i$ into two parts: $\mathcal{T}_i^1$ and $\mathcal{T}_i^2$;
7 $centroids_i^1, assignments_i^1 \leftarrow$ **call** $Kmeans(\mathcal{T}_i^1, \sqrt{K}, t)$;
8 $centroids_i^2, assignments_i^2 \leftarrow$ **call** $Kmeans(\mathcal{T}_i^2, \sqrt{K}, t)$;
9 $centroids.append(centroids_i^1, centroids_i^2)$;
10 **for** $j = 0$ *to* $n - 1$ **do**
11 $IMI_i[assignments_i^1[j], assignments_i^2[j]].append(j)$;
12 $IMIs.append(IMI_i)$;
13 **return** *centroids* and *IMIs*;

4.1 Index Construction

As analyzed in Section 3, subspace-oriented data transformation is a prerequisite for adapting the subspace collision framework to the data distribution. This process is formalized in Algorithm 1. First, compute the mean vector and covariance matrix of data points, and perform the spectral decomposition to obtain eigenvalues and eigenvectors (lines 2-4). Algorithm 2 (detailed subsequently) is then invoked to generate the subspace-oriented eigenvector allocation set (line 5). Within each subspace, all data points are transformed using the assigned s eigenvectors (lines 6-9). The concatenation of these subspace-specific transformed representations produces the final transformed dataset of dimensionality $N_s \cdot s$ (lines 10-12).

Algorithm 2 presents the eigensystem allocation procedure, which provides the optimal solution for balanced subspace partition, as proven in Section 3.2. Specifically, N_s eigenvector buckets and eigenvalue product trackers are created (lines 1-2). Eigenvalues are scaled for ease of calculation (line 3), then reordered with corresponding eigenvectors in descending sequence (line 4). The eigensystem allocation sequentially processes the top $N_s \cdot s$ eigenvectors in descending eigenvalue order (line 5). For each eigenvector being processed (corresponding to the largest available eigenvalue), the bucket with the minimum product of the eigenvalues is selected. Each bucket contains no more than s eigenvectors (line 6). The eigenvector is then assigned to the selected bucket and the corresponding eigenvalue product tracker is updated (lines 7-8). The finalized eigenvector allocation set is returned (line 9).

Figure 4 illustrates the workflow of TaCo for index construction, while Algorithm 3 provides the corresponding pseudocode. The process begins by invoking the *Subspace-oriented Data Transformation* algorithm to the original dataset, resulting in a transformed dataset with reduced dimensionality $N_s \cdot s$ (line 1). This transformed dataset is then partitioned into N_s subspaces, each of dimensionality s (line 2). For each subspace, the algorithm constructs a separate inverted multi-index (lines 4-12). First, the subspace data is split into two disjoint parts in terms of dimension (line 6). Each part is then clustered using K-means with $\sqrt{K}$ centroids and t iterations, generating two sets of centroids and corresponding assignment labels for all data points (lines 7-8). To enable efficient retrieval during

Algorithm 4: Scalable Dynamic Activation

Input: Collision ratio α, dataset size n, number of K-means clusters K, distances and indices of the first and second parts in a subspace $dists_1, idx_1, dists_2, idx_2$, the inverted multi-index IMI

Output: Clusters containing data points that collide with q

1 Initialize a list $retrieved\_clusters$, a min heap $active\_clusters$, an array $active\_idx$ of length $\sqrt{K}$ and set to 0;
2 $retrieved\_num \leftarrow 0$;
3 $active\_clusters.push(pair < dists_1[idx_1[0]] + dists_2[idx_2[0]], 0 >)$;
4 **while** *TRUE* **do**
5 $selected\_cell \leftarrow active\_clusters.top()$;
6 $pos \leftarrow selected\_cell.second$;
7 $cluster \leftarrow IMI[idx_1[pos], idx_2[active\_idx[pos]]]$;
8 $retrieved\_clusters.append(cluster)$;
9 $retrieved\_num$ += $sizeof(cluster)$;
10 **if** $retrieved\_num \geq \alpha \cdot n$ **then**
11 break;
12 **if** $active\_idx[pos] == 0$ *and* $pos < \sqrt{K} - 1$ **then**
13 $active\_clusters.push(pair < dists_1[idx_1[pos + 1]] + dists_2[idx_2[0]], pos + 1 >)$;
14 $active\_clusters.pop()$;
15 **if** $active\_idx[pos] < \sqrt{K} - 1$ **then**
16 $active\_idx[pos]$++;
17 $selected\_cell.first \leftarrow dists_1[idx_1[pos]] + dists_2[idx_2[active\_idx[pos]]]$;
18 $active\_clusters.push(selected\_cell)$;
19 **return** $retrieved\_clusters$;

query processing, a map structure is constructed as the inverted multi-index for each subspace. For each point, its assignment labels from both clustering operations form a composite key within the map, under which the point's original identifier is appended to the corresponding cell (lines 10-12).

We note that the main difference between TaCo and SuCo [86] in index construction is that TaCo achieves balanced subspace partitioning by invoking the *Subspace-oriented Data Transformation* function (Algorithm 1), which addresses the data-agnostic subspace partitioning problem encountered by SuCo's index structure. The subsequent steps of building an IMI within each subspace follow the same procedure in both TaCo and SuCo.

4.2 Collision Counting

The *Dynamic Activation* algorithm was proposed along with SuCo to facilitate efficient collision counting [86]. However, its scalability remains limited. The algorithm maintains an activation list via a linear array, resulting in $O(l)$ query complexity and $O(1)$ update complexity, where $l = \sqrt{K}$ denotes the length of the IMI list. While this design offers satisfactory performance when l is small, it becomes a bottleneck when a larger l is required to improve indexing precision. The linear query complexity constrains overall efficiency under such conditions. To address this limitation, we propose the *Scalable Dynamic Activation* algorithm (Algorithm 4). Experimental results in Section 5.2.2 demonstrate that the *Scalable Dynamic Activation* algorithm improves efficiency by

up to 30% compared to the original *Dynamic Activation* algorithm. The *Scalable Dynamic Activation* algorithm is not only applicable to TaCo, but can also be migrated to *any* method that uses the IMI index structure.

Algorithm 4 gives the pseudocode of the *Scalable Dynamic Activation* algorithm. It begins by pushing the initial cluster pair with the smallest distance sum into a min heap (line 3). In each iteration, the cluster combination with the smallest current distance is popped from the min heap and added to the result set (lines 6-9). The number of data points retrieved is accumulated, and the algorithm terminates once the accumulated count meets the threshold (lines 10-11). If the current cluster from the first subspace is being activated for the first time and subsequent clusters remain available, the next cluster is activated and pushed into the min heap (lines 12-13). The current cluster's activation index is incremented, and its next combination in the second subspace is computed and pushed back into the min heap for further evaluation (lines 14-18). The algorithm returns all retrieved clusters once the collision requirement is satisfied (line 19). This mechanism ensures that cluster combinations are retrieved in ascending order of their distance sums, enabling *early termination* when sufficient collisions are collected.

The main difference between TaCo and SuCo in the *Collision Counting* step is that TaCo adopts a min-heap to manage the activation list in its *Scalable Dynamic Activation* algorithm, achieving $O(1)$ query complexity, thereby addressing the linear query complexity problem faced by SuCo's *Dynamic Activation* algorithm. This new data structure introduces corresponding changes to the algorithm logic, including the use of a pair structure (lines 3, 6, 13, and 17 in Algorithm 4) and the management of the heap (lines 3, 5, 13, and 18 in Algorithm 4). The logic of sequentially retrieving data points by dynamically activating cells in the IMI is the same for both TaCo and SuCo.

4.3 Candidate Selection

4.3.1 Overview. The query strategies of SuCo are *query-oblivious*, employing same query overhead for all queries [86]. To enhance query efficiency, we aim to design *query-aware* query strategies for TaCo. The entire query pipeline offers two opportunities for per-query customization: collision counting and candidate selection. Collision counting already leverages an *early-termination* strategy through the *Scalable Dynamic Activation* algorithm, achieving optimal overhead adaptation for distinct queries. However, designing a *query-aware* candidate selection strategy remains an open problem. Through statistical analysis of intermediate query states (specifically, the distribution of SC-scores), we observe that certain queries exhibit highly discriminative SC-score distributions, where high recall can be achieved by re-ranking only the top-scoring data points. In contrast, other queries show poor SC-score discriminability, requiring a larger candidate set for re-ranking to achieve comparable recall. Thus, unlike SuCo, which selects a fixed number $\beta \cdot n$ of candidates for all queries, the *Query-aware Candidates Selection* algorithm designed for TaCo dynamically determines the candidate set size per query based on its SC-score distribution.

4.3.2 Algorithm Design. Algorithm 5 provides the pseudocode of the *Query-aware Candidates Selection* algorithm, which identifies candidate points based on the collision distribution of each query. The algorithm begins by initializing an empty candidate set C and an array $collision\_num$ of size $N_s + 1$ to count the number of data points for each possible SC-score (lines 1-2). It then iterates through all data points to populate $collision\_num$ according to their SC-scores (lines 3-4). Next, the algorithm determines a threshold $last\_collision$ starting from the maximum possible score N_s (line 5). It selects candidates from higher to lower SC-scores (lines 7-8) until the accumulated number of candidates exceed $\beta \cdot n$ (lines 9-12). The algorithm collects all data points whose SC-scores are no less than $last\_collision$, forming the candidate set C (lines 13-15). Finally, it returns C together with the total number of candidates (line 16).

Algorithm 5: Query-aware Candidates Selection

Input: Dataset $\mathcal{D} = \{o_1, o_2, \ldots, o_n\}$, SC-score distribution: SC_scores, re-rank ratio β, subspace number N_s, dataset size n
Output: Selected candidates C and the candidate number

Initialize an empty candidate set $C \leftarrow \varnothing$;
Initialize an array $collision\_num$ of size $N_s + 1$ and set to 0;
for $i = 1$ *to* n **do**
    $collision\_num[SC\_scores[i-1]]$++;
$last\_collision \leftarrow N_s$;
$candidate\_num \leftarrow 0$;
for $j = N_s$ *to* 0 **do**
    $candidate\_num$ += $collision\_num[j]$;
    if $collision\_num[j] \leq \beta \cdot n - candidate\_num$ **then**
        $last\_collision$ -= 1;
    else
        break;
for $i = 1$ *to* n **do**
    if $SC\_scores[i-1] \geq last\_collision$ **then**
        $C = C \cup \{o_i\}$;
return the candidate set C and $candidate\_num$;

4.3.3 Theoretical Analysis. We now provide a theoretical justification for how the query discriminability affects SC-score distribution and leads to different candidate selection behaviors.

Definition 7 (Query Discriminability)**.** Given a query q and the number of subspaces N_s. As a consequence of the uniformity of the subspaces after the subspace-oriented data transformation in Theorem 1, we denote p^* the *collision probability* such that subspace collision happens for a true nearest neighbor $o^* \in \mathcal{R}^* = \{o_1^*, \ldots, o_k^*\}$, and p the probability that subspace collision happens for a non-neighbor $o \notin \mathcal{R}^*$, with $p^* > p$. The Discriminability Gap of q is defined as $\Delta(q) = p^* - p > 0$.

Intuitively, for queries with larger discriminability gap $\Delta(q)$, the SC-score of a true nearest neighbor $SC(o)$ and that of a non-neighbor $SC(o^*)$ are further separated than queries with smaller discriminability gap. As a consequence, the Type-I error (i.e., the probability that a non-neighbor is incorrectly classified as a neighbor) and Type-II error (i.e., the probability that a true neighbor is incorrectly classified as a non-neighbor) are expected to decrease rapidly as $\Delta(q)$ grows large. This intuition is made precise in the following result.

Lemma 2 (SC-score Separation)**.** *For a given query q and its discriminability gap $\Delta(q)$ as in Definition 7, denote o^* a true nearest neighbor of q and $o \notin \mathcal{R}^*$ a non-neighbor. Then, the optimal Type-I and Type-II errors both decay at least at the rate* $\exp\left(-\frac{N_s \Delta^2(q)}{8p(1-p)}\right)$*. That is, both errors decay exponentially fast to zero as N_s or $\Delta(q)$ becomes large.*

Proof of Lemma 2. As a consequence of Theorem 1, we have that $SC(o)$ the SC-score of o follows a binomial distribution with parameters N_s and p, and $SC(o^*)$ the SC-score of o^* follows a binomial distribution with parameters N_s and p^*. The associated Bernoulli KL divergence writes,

Algorithm 6: k-ANNS Query

Input: Dataset $\mathcal{D}$, dataset size n, data dimensionality d, a query point q, number of results k, subspace number N_s, subspace dimensionality s, collision ratio α, re-rank ratio β, number of K-means clusters K, the centroid list $centroids$, the IMI list $IMIs$

Output: k nearest points to q in $\mathcal{D}$

1 Transform q with the eigenvalues obtained from the *Subspace-oriented Data Transformation* algorithm;
2 Divide q into N_s subspaces: $q^1, \ldots, q^{N_s}$;
3 **for** $i = 1$ *to* N_s **do**
4 Further split q^i into two parts q_1^i and q_2^i;
5 Calculate the distance between q^i and centroids in each part, obtain $dists_1^i$, $dists_2^i$ and sorted indices idx_1^i, idx_2^i;
6 $retrieved\_clusters \leftarrow$ **call** *Scalable Dynamic Activation* $(\alpha, n, K, dists_1^i, idx_1^i, dists_2^i, idx_2^i, IMIs[i-1])$;
7 $SC\_scores \leftarrow$ Derived from $retrieved\_clusters$;
8 $C, candidate\_num \leftarrow$ **call** *Query-aware Candidates Selection*$(\mathcal{D}, SC\_scores, \beta, N_s, n)$;
9 **return** the refined *top-k* points closest to q in C;

for small $\Delta(q)$ as

$$D(p||p^*) = \frac{\Delta^2(q)}{2p(1-p)} + O(\Delta^3(q)). \tag{13}$$

Using the Bayes optimal decision rule (that balances the two log-likelihoods) and applying the (relative entropy) Chernoff bound, we conclude the proof of Lemma 2. □

By operating directly on observable SC-score levels and scanning in descending order, Algorithm 5 implements a query-driven stopping rule, where the decision to stop collecting candidates is governed solely by the available SC-score ordering and the refinement budget. In summary, Algorithm 5 is both theoretically justified by the SC-score separation properties and practically effective as it adapts to the intrinsic discriminability of each query without extra estimation overhead.

4.4 k-ANNS Query

Algorithm 6 illustrates how TaCo supports *k-ANNS* queries. The procedure starts by transforming the query point q using the eigenvalues obtained from the *Subspace-oriented Data Transformation* algorithm, followed by partitioning the transformed query into N_s subspaces (lines 1–2). For each subspace, the distances between the query point and all centroids in both of its divided parts are calculated, and the indices corresponding to these distances sorted in ascending order are obtained (lines 3-5). Subsequently, the *Scalable Dynamic Activation* algorithm is invoked to retrieve clusters containing data points that collide with query q in each subspace (line 6). The collision counts for all data points are then aggregated accordingly (line 7). After processing all subspaces, the *Query-aware Candidates Selection* algorithm is invoked to generate a candidate set based on the provided SC-score distribution (line 8). Finally, all candidates are refined by their exact distances to q, and the top-k points with the smallest distances are returned (line 9).

Algorithm 6 reflects the query pipeline under the subspace collision framework. Since both TaCo and SuCo are instances of this framework, their overall pipelines are structurally similar. TaCo's key innovation lies in the design of the internal modules within the pipeline (Algorithms 1, 4, 5), which leads to a significant performance gain over the query algorithm used in SuCo.

5 Experimental Evaluation

In this section, we evaluate the performance of TaCo through self-evaluations and comparative experiments against state-of-the-art ANNS methods, including both subspace collision-based and non-subspace collision-based methods. All methods are implemented in C/C++ and parallelized using OpenMP and/or Pthreads, with SIMD instructions utilized to accelerate computation. TaCo is developed in C++ and compiled with -O3 optimization. All experiments are performed on a server with two AMD EPYC 9554 CPUs operating at 3.10 GHz and 756 GB of memory, running Ubuntu 22.04.

5.1 Experimental Setup

Datasets and Queries. We use five public real-world datasets with varying sizes and dimensionalities, which are widely adopted in ANNS studies [28, 55, 86, 89]. The datasets include: 256-dimensional DEEP1M, 960-dimensional GIST1M, 128-dimensional SIFT10M, 96-dimensional Yandex DEEP10M, and 100-dimensional Microsoft SPACEV10M. Note that SIFT10M, Yandex DEEP10M, and Microsoft SPACEV10M are randomly sampled subsets of their corresponding billion-scale datasets[2]. For each dataset, we randomly select 100 points as query points and remove them from the original datasets.

Benchmark Methods. We compare TaCo with eleven state-of-the-art in-memory ANN methods, covering both subspace collision-based and non-subspace collision-based methods. (1) For subspace collision-based methods: **SuCo** [86] is the state-of-the-art method under the subspace collision framework before our work. To evaluate the impact of the our proposed data transformation and query strategy optimizations on algorithm performance, we propose three ablation methods: **SuCo-DT** (SuCo with subspace-oriented data transformation), **SuCo-CS** (SuCo with query-aware candidates selection), and **SuCo-QS** (SuCo with all optimized query strategies). **SC-Linear** serves as a baseline that performs linear scan without any index structure. (2) For non-subspace collision-based methods: **IMI-OPQ** [30] uses the same IMI structure as TaCo and is recognized for its strong performance in the FAISS library [18]. **DET-LSH** [89] follows a query pipeline similar to TaCo, involving data projection, candidate selection, and final re-ranking. **IVF-RaBitQ** [27] is the state-of-the-art quantization method, achieves unbiased estimation and has been widely used in the industry. **HNSW** [55] is a widely adopted graph-based method known for its effectiveness in query processing. **MIRAGE** [72] and **SHG** [32] are optimized based on HNSW and demonstrate strong performance.

Evaluation Measures. We evaluate all methods using six performance measures: indexing time, index memory footprint, query time, queries per second (QPS), recall, and mean relative error (MRE) [3, 62, 63]. Among these, indexing time, query time, and QPS reflect efficiency; index memory footprint indicates storage overhead; and recall and MRE assess the quality of the returned results. Note that offline data preprocessing steps (e.g., data projection for DET-LSH, data rotation for IMI-OPQ, and data transformation for TaCo) are not included in the indexing time, which is a common practice in previous works for evaluating experiments [30, 47, 71, 89]. For a query q, let $\mathcal{R} = \{o_1, \ldots, o_k\}$ be the returned result set and $\mathcal{R}^* = \{o_1^*, \ldots, o_k^*\}$ be the exact k-NN set. Recall is defined as $\frac{|\mathcal{R} \cap \mathcal{R}^*|}{k}$, and mean relative error (MRE) is defined as $\frac{1}{k} \sum_{i=1}^{k} \frac{\|q,o_i\| - \|q,o_i^*\|}{\|q,o_i^*\|}$.

Parameter Settings. Both the indexing and query phases of ANNS methods involve configurable parameters. For indexing parameters, we adhere to the optimal configurations recommended in previous studies [19, 20, 49, 76, 86, 89] and empirically validate these configurations to ensure all methods achieve optimal indexing performance. For query parameters, we dynamically adjust their

[2]http://corpus-texmex.irisa.fr; https://big-ann-benchmarks.com/neurips21.html

Table 2. Comparison on TaCo and SC-Linear

Method	Dataset	Query Time (ms)	Speedup	Recall
SC-Linear	DEEP1M	273.7	\	0.968
	SIFT10M	4786.1	\	0.9906
TaCo	DEEP1M	1.266	216.2	0.9358
	SIFT10M	6.705	713.8	0.9726

Table 3. Parameter settings for each dataset

Dataset	Dim.	N_s	s	Dim. Reduction
DEEP1M	256	6	8	81.25%
GIST1M	960	4	10	95.83%
SIFT10M	128	6	6	71.88%
Yandex Deep10M	96	6	8	50%
Microsoft SPACEV10M	100	6	10	40%

values to examine the trade-offs between query efficiency and accuracy, enabling a comprehensive evaluation across different settings. For TaCo, SuCo, SuCo-DT, SuCo-CS, and SuCo-QS, $N_s \in [4, 10]$, $s \in [6, 12]$, $\alpha \in [0.01, 0.1]$, $\beta \in [0.001, 0.05]$. For IMI-OPQ, $M = 2$, $K = 2^{18}$, $\beta \in \left[10^{-4}, 10^{-1}\right]$. For DET-LSH, $L = 4$, $K = 16$, $\beta \in [0.005, 0.2]$, $c = 1.5$. For IVF-RaBitQ, we adopt the default settings of the RaBitQ library, i.e., $K = 2^{12}$ and $B = 3$. For HNSW, $efConstruction = 200$, $M = 25$, $efSearch \in [300, 3000]$. For MIRAGE, $S = 32$, $R = 4$, $iter = 15$. For SHG, $M = 48$, $efConstruction = 80$. The value of k in k-ANNS is set to 50 by default.

5.2 Self-evaluation of TaCo

5.2.1 TaCo vs. SC-Linear. We first compare the query performance between TaCo and SC-Linear under same parameter settings ($\alpha = 0.05$, $\beta = 0.005$). As shown in Table 2, TaCo achieves a significant improvement in query efficiency: exceeding **700**× speedup on the SIFT10M dataset, with only a marginal decrease in recall. The speedup ratio increases substantially with the size of the dataset, demonstrating that TaCo has strong scalability. These results confirm that TaCo's index structure and query strategies can be effectively applied to the subspace collision framework for ANNS.

5.2.2 Scalable Dynamic Activation vs. Dynamic Activation. The *Scalable Dynamic Activation* algorithm, introduced in Section 4.2, addresses scalability issues in IMI collision counting. Figure 5 compares the efficiency of the *Scalable Dynamic Activation* and the original *Dynamic Activation* algorithms on the SIFT10M dataset under varying numbers of clusters K and collision ratios α. When K is small, both algorithms perform similarly, with the *Dynamic Activation* algorithm even slightly outperforming in some cases. However, as K increases, the *Scalable Dynamic Activation* algorithm exhibits a notable performance improvement, achieving up to a 30% reduction in query time. This gain comes from its min-heap structure, which reduces query complexity to $O(1)$ versus the original's $O(\sqrt{K})$. For small K, heap operations (push and pop) slightly reduce performance due to update overhead. As K increases, the linear query cost of *Dynamic Activation* becomes a bottleneck, while the constant-time queries of *Scalable Dynamic Activation* provide significant advantages, confirming its superior scalability.

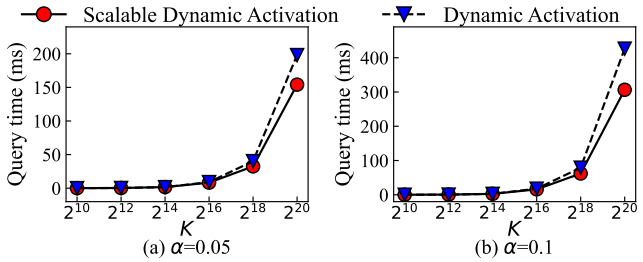


Fig. 5. Efficiency comparison between Scalable Dynamic Activation and Dynamic Activation algorithms on SIFT10M.

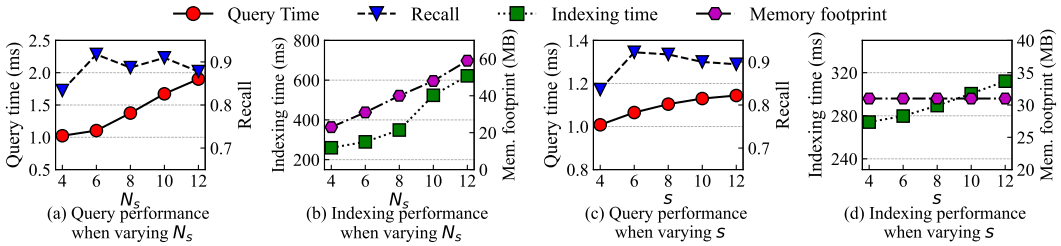


Fig. 6. Performance of TaCo when varying the number of subspaces N_s and the subspace dimensionality s on DEEP1M.

5.2.3 Parameter study. Parameters N_s and s critically affect the indexing and query performance of TaCo. As shown in Figure 6 (a) and (b), increasing N_s generally leads to higher indexing time and memory consumption. While query performance improves initially with N_s and reaches an optimum at $N_s = 6$, beyond which further increasing N_s leads to a gradual rise in query time and may even reduce recall. As shown in Figure 6 (c) and (d), increasing s leads to higher computational costs, consequently resulting in increased indexing and query time. Furthermore, recall reaches its optimal value at $s = 6$. Table 3 summarizes the optimal parameter settings of TaCo for each dataset, where $1 - \frac{N_s \cdot s}{d}$ represents the dimensionality reduction rate. It can be observed that the dimensionality reduction rates achieved across datasets are impressive, ranging from 40% to as high as 95.83%. The dimensionality reduction is a key factor contributing to TaCo's superior indexing and query efficiency compared to subspace collision-based methods.

5.3 TaCo vs. Subspace Collision Methods

5.3.1 Indexing performance. Figure 7 shows the indexing time and index memory footprint of all subspace collision-based methods. Note that the indexing time does not include preprocessing time. Owing to the use of subspace-oriented data transformation in both TaCo and SuCo-DT, these two methods achieve same indexing performance. Similarly, SuCo-QS, SuCo-CS, and SuCo, which do not incorporate this transformation, also exhibit same indexing performance. Therefore, we focus our analysis on comparing TaCo and SuCo. Experimental results indicate that TaCo achieves up to **8**× speedups in indexing while consuming only **0.6**× memory footprint when compared with SuCo. This significant improvement can be attributed to three main factors: dimensionality reduction, fewer subspaces, and code-level optimizations. These results collectively confirm the effectiveness of the proposed subspace-oriented data transformation in enhancing indexing performance.

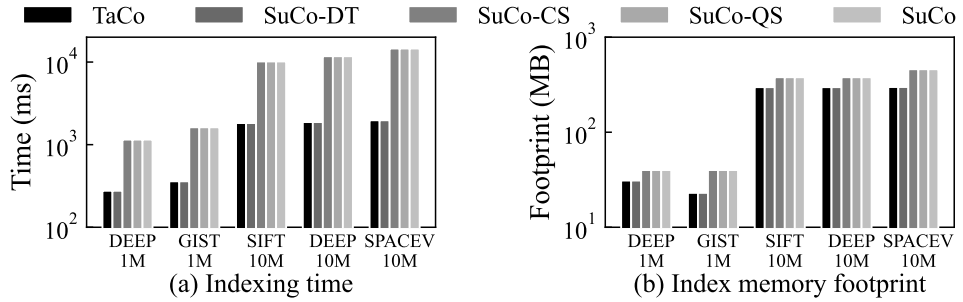


Fig. 7. Indexing performance comparison between TaCo and subspace collision-based methods.

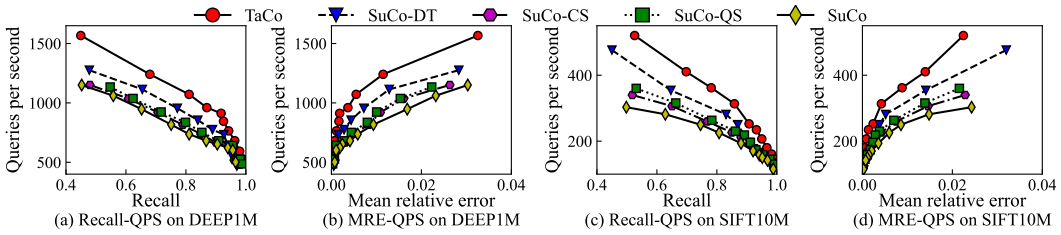


Fig. 8. Query performance comparison between TaCo and subspace collision-based methods.

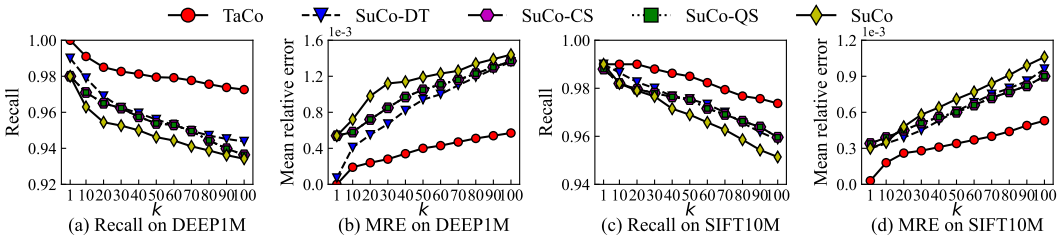


Fig. 9. Performance under different k: TaCo vs. subspace collision-based methods.

5.3.2 Query performance. In k-ANNS tasks, a fundamental trade-off exists between query efficiency and accuracy: longer query processing times generally yield higher-quality results, whereas an excessive focus on efficiency may compromise result quality. Therefore, a comprehensive evaluation of query performance must consider both efficiency (measured by QPS) and accuracy (measured by recall and MRE). Figure 8 presents the recall-QPS and MRE-QPS curves for all subspace collision-based methods. The results show that TaCo achieves over **1.5**× QPS at 0.9 recall when compared with SuCo. Furthermore, SuCo-DT, SuCo-CS, and SuCo-QS exhibit better query performance than the original SuCo method. These ablation experiments confirm that the three optimization techniques we proposed can effectively improve query performance.

5.3.3 Performance under different k. The ability of a k-ANNS method to maintain stable performance across varying values of k is an important aspect of its scalability. Figure 9 illustrates the query performance of all subspace collision-based methods under different k values ranging from 1 to 100. SuCo-CS and SuCo-QS overlap in the figure because, under the same parameter settings, the Scalable Dynamic Activation and the Dynamic Activation algorithms return identical results and therefore do not affect recall. As k increases, all methods exhibit a slight decline in query

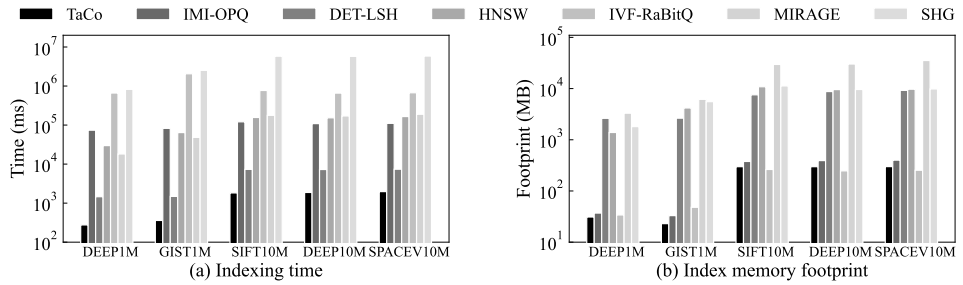


Fig. 10. Indexing performance comparison between TaCo and non-subspace collision-based methods.

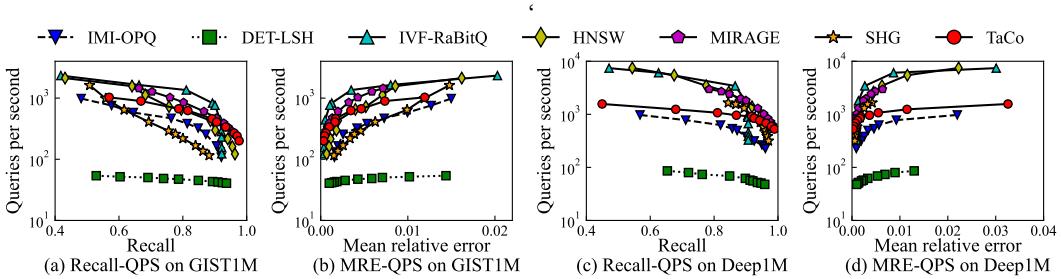


Fig. 11. Query performance comparison between TaCo and non-subspace collision-based methods.

accuracy. This is primarily because the number of candidate points remains fixed irrespective of k, a larger k thus increases the probability of missing true nearest neighbors. Nevertheless, TaCo consistently outperforms all other subspace collision-based methods across all evaluated settings of k, demonstrating its robustness and superior scalability.

5.4 TaCo vs. Non-Subspace Collision methods

This section presents a comparative analysis between TaCo and six representative methods from different ANNS paradigms: IMI-OPQ and IVF-RaBitQ (VQ-based); DET-LSH (LSH-based); HNSW, MIRAGE, and SHG (graph-based). Although cross-paradigm comparisons are non-intuitive due to fundamental architectural differences, this evaluation offers valuable insights into TaCo's competitive positioning relative to non-subspace collision methods.

5.4.1 Indexing performance. Figure 10(a) shows that TaCo achieves the best indexing efficiency, while Figure 10(b) shows that TaCo, IMI-OPQ, and IVF-RaBitQ achieve the lowest memory consumption among all evaluated methods. Note that the reported indexing time excludes preprocessing, and the index memory footprint does not include the storage cost of the dataset. TaCo, IMI-OPQ, and IVF-RaBitQ utilize the inverted file-based structure (IMI and IVF), whose lightweight design lead to significantly lower memory usage compared to DET-LSH (tree-based index) and HNSW, MIRAGE, SHG (graph-based index). However, due to differences in design objectives, TaCo performs coarse-grained collision probing within each subspace under the subspace collision framework, while IMI-OPQ and IVF-RaBitQ require fine-grained partitioning of all data points in the original space. As a result, compared with IMI-OPQ and IVF-RaBitQ, TaCo reduces indexing time by 2~3 orders of magnitude. In contrast, DET-LSH relies on fine-grained tree-based partitioning, and HNSW, MIRAGE, SHG connect each point to its nearest neighbors in a graph structure. These approaches result in heavier index structures, requiring longer indexing times and larger memory footprints.

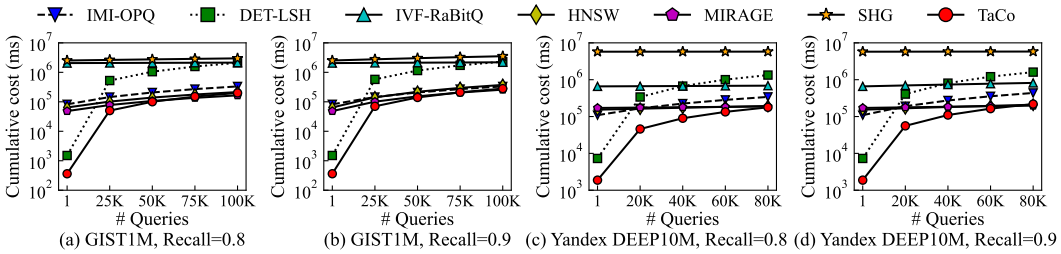


Fig. 12. Cumulative query cost (start with the indexing time): TaCo vs. non-subspace collision-based methods.

In summary, TaCo achieves superior indexing performance due to its efficient subspace-oriented design and lightweight index.

5.4.2 Query performance. Figure 11 presents the recall–QPS and MRE–QPS curves for TaCo and non-subspace collision-based methods. Experimental results indicate that IVF-RaBitQ, HNSW, and MIRAGE perform well at moderate recall levels (0.6~0.9), whereas TaCo exhibits better stability at higher recall levels (>0.9), particularly on the hard datasets such as GIST1M. Graph-based methods (HNSW, MIRAGE, SHG) achieve high query efficiency at the cost of substantial indexing time and memory consumption. During indexing, they identify and connect near neighbors for each data point, while queries follow a gradually converging path through the graph. These methods perform better on large and simple datasets like Deep1M. However, on hard datasets such as GIST1M, their greedy search strategy is prone to becoming trapped in local optima, limiting query performance. In contrast, TaCo employs a subspace collision probing strategy that offers greater robustness and fault tolerance. This allows it to perform consistently well even on hard datasets. IVF-RaBitQ leverages SIMD operations to accelerate computations on quantized codes, thus achieving QPS similar to graph-based methods; however, the limited number of quantization bits makes it challenging to attain very high recall. In summary, TaCo achieves state-of-the-art indexing performance, while simultaneously attaining top-tier query performance, establishing it as a lightweight and effective solution for k-ANNS tasks.

5.4.3 Overall Evaluation. Given the considerable differences in indexing and query performance across different categories of ANNS methods, it is essential to conduct a comprehensive evaluation that integrates both indexing and query metrics. Figure 12 shows the cumulative query costs of all methods under the same recall, where the cost starts with the indexing time. The experimental results demonstrate the advantages of TaCo: it completes index construction and processes 50K-80K queries before the best competitor (i.e., HNSW and MIRAGE) answers its first query. In contrast, the slow index construction of IVF-RaBitQ and SHG, as well as the slow query time of DET-LSH, make them less competitive in this evaluation. In summary, while different categories of methods are suited to particular scenarios due to their distinctive characteristics, TaCo achieves an effective balance between indexing and query efficiency, making it especially suitable for applications that require rapid deployment and immediate query responses, such as retrieval-based sparse attention for LLM inference acceleration [53, 97].

6 Related Work

Approximate Nearest Neighbor Search. With data being generated at an unprecedented rate and datasets increasing in scale [22, 25, 26, 96, 102, 103], more efficient management of large-scale data is required to facilitate advanced analysis [50, 59, 60, 85, 88]. A wide range of effective ANNS methods have been proposed [79, 83]: locality-sensitive hashing (LSH)-based methods [2, 44, 71, 89, 100],

vector quantization (VQ)-based methods [6, 27, 28, 30, 37, 58], tree-based methods [8, 42, 65, 77, 78, 81, 82, 93], graph-based methods [5, 24, 32, 49, 55, 72], and hybrid methods [4, 12, 33, 99]. Subspace collision is a newly proposed framework, which achieves a better balance between index construction and query answering [86]. Our work advances the subspace collision framework by introducing subspace-oriented data transformation and optimizing query strategies, thereby extending the boundaries of efficient and scalable ANNS.

AI for ANNS and ANNS for AI. Traditional ANNS methods rely on heuristic designs, limiting their ability to adapt indexing and query strategies to data distributions or query characteristics. In contrast, deep learning can capture intrinsic features that are difficult to manually design [43], motivating a range of learning-enhanced ANNS approaches, including learn-to-index [14, 34, 48, 95], learn-to-query [7, 23], learned early termination [11, 46], and learned cardinality estimation [68, 101]. Furthermore, with advances in large language models (LLMs) [98], ANNS has gained significant attention for enhancing AI applications [90]. For instance, retrieval-augmented generation (RAG) uses ANNS to provide contextual information for LLM queries [21, 29, 45], and key-value cache (KVCache) employs ANNS to accelerate model inference [17, 53, 67, 97].

Hybrid Search. Traditional ANNS primarily focuses on vector search. However, how to achieve hybrid search by integrating ANNS with traditional database systems has become a key concern in both academia [31, 61, 92, 104] and industry [56, 73, 84, 94]. Hybrid search enhances traditional ANNS by enabling vectors to satisfy additional attribute-based constraints [74, 91], greatly expanding its applicability in real-world applications. For example, in e-commerce, users can specify attributes such as price range and product style while still leveraging semantic similarity search [10, 92]. In search engines, results can be filtered by domain or user privileges without compromising retrieval relevance [10]. In RAG systems, vector databases exploit metadata-based filtering to improve retrieval accuracy, thereby enhancing generative AI applications [21, 29]. This integration of vector-based retrieval and structured data filtering offers a practical paradigm for next-generation information retrieval systems.

7 Conclusions

In this paper, we first designed the subspace-oriented data transformation mechanism to enable data-adaptive subspace collision framework. Next, we presented query-aware and scalable query strategies that dynamically allocate overhead for each query and accelerate collision probing within subspaces. Then, we proposed the TaCo method, which achieves efficient and accurate ANN search while maintaining an excellent balance between indexing and query performance. Finally, we conducted extensive experiments, and the results demonstrate the superiority of TaCo. In future work, we will combine deep learning to explore learn-based index structures and query strategies under the subspace collision framework, as well as randomized numerical linear algebra techniques [16, 57].

Acknowledgments

T. Palpanas supported by EU Horizon projects TwinODIS (101160009) and DataGEMS (101188416), and by *ΥΠΑΙΘΑ* & NextGenerationEU project HARSH (*ΥΠ*3*TA* – 0560901) that is carried out within the framework of the National Recovery and Resilience Plan "Greece 2.0" with funding from the European Union – NextGenerationEU. Z. Liao would like to acknowledge the National Natural Science Foundation of China (via fund NSFC-12571561) and the Fundamental Research Support Program of HUST (2025BRSXB0004) for providing partial support.

References

[1] 2005. *Differential Entropy*. John Wiley & Sons, Ltd, Chapter 8, 243–259. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/047174882X.ch8 doi:10.1002/047174882X.ch8

[2] 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.

[3] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* 87 (2020), 101374.

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

[5] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2025. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–31.

[6] Artem Babenko and Victor Lempitsky. 2014. The inverted multi-index. *IEEE transactions on pattern analysis and machine intelligence* 37, 6 (2014), 1247–1260.

[7] Dmitry Baranchuk, Dmitry Persiyanov, Anton Sinitsin, and Artem Babenko. 2019. Learning to route in similarity graphs. In *International Conference on Machine Learning*. PMLR, 475–484.

[8] Erik Bernhardsson. 2015. *Approximate Nearest Neighbors in C++/Python optimized for memory usage and loading/saving to disk*. https://github.com/spotify/annoy

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

[10] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating labels and vectors: A unified approach to filtered approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–27.

[11] Manos Chatzakis, Yannis Papakonstantinou, and Themis Palpanas. 2026. DARTH: Declarative Recall Through Early Termination for Approximate Nearest Neighbor Search. *SIGMOD* (2026).

[12] Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. 2018. SPTAG: A library for fast approximate nearest neighbor search.

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

[14] Chih-Yi Chiu, Amorntip Prayoonwong, and Yin-Chih Liao. 2019. Learning to index for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 42, 8 (2019), 1942–1956.

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

[16] Michal Derezinski, Zhenyu Liao, Edgar Dobriban, and Michael Mahoney. 2021. Sparse Sketches with Small Inversion Bias. In *Proceedings of Thirty Fourth Conference on Learning Theory*. PMLR, 1467–1510.

[17] Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. [n. d.]. HashAttention: Semantic Sparsity for Faster Inference. In *Forty-second International Conference on Machine Learning*.

[18] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281* (2024).

[19] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The lernaean hydra of data series similarity search: an experimental evaluation of the state of the art. *Proceedings of the VLDB Endowment* 12, 2 (2018), 112–127.

[20] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2019. Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. *Proc. VLDB Endow.* 13, 3 (2019), 403–420. doi:10.14778/3368289.3368303

[21] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 6491–6501.

[22] Ziyu Fei, Ying Li, Jiuqi Wei, Yufan Fu, Botao Peng, and Xiaodong Li. 2023. Flexauth: A decentralized authorization system with flexible delegation. In *TrustCom*.

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

[24] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment* 12, 5 (2019), 461–474.

[25] Yufan Fu, Xiaodong Lee, Jiuqi Wei, Ying Li, and Botao Peng. 2024. Securing the internet's backbone: A blockchain-based and incentive-driven architecture for DNS cache poisoning defense. *Computer Networks* (2024).

[26] Yufan Fu, Jiuqi Wei, Ying Li, Botao Peng, and Xiaodong Li. 2023. Ti-dns: A trusted and incentive dns resolution architecture based on blockchain. In *TrustCom*.

[27] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2025. Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[28] Jianyang Gao and Cheng Long. 2024. Rabitq: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.

[29] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* 2, 1 (2023).

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

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

[32] Zengyang Gong, Yuxiang Zeng, and Lei Chen. 2025. Accelerating Approximate Nearest Neighbor Search in Hierarchical Graphs: Efficient Level Navigation with Shortcuts. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3518–3530.

[33] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–26.

[34] Gaurav Gupta, Tharun Medini, Anshumali Shrivastava, and Alexander J Smola. 2022. Bliss: A billion scale index using iterative re-partitioning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 486–495.

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

[36] 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).

[37] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[38] Kerstin Johnsson, Charlotte Soneson, and Magnus Fontes. 2014. Low bias local intrinsic dimension estimation from expected simplex skewness. *IEEE transactions on pattern analysis and machine intelligence* 37, 1 (2014), 196–202.

[39] Iain M. Johnstone and Debashis Paul. 2018. PCA in High Dimensions: An Orientation. *Proc. IEEE* 106, 8 (2018), 1277–1292. doi:10.1109/jproc.2018.2846730

[40] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[41] Adam Kołacz and Przemysław Grzegorzewski. 2016. Measures of dispersion for multidimensional data. *European Journal of Operational Research* 251, 3 (2016), 930–937.

[42] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. 2018. Coconut: A Scalable Bottom-Up Approach for Building Data Series Indexes. *Proceedings of the VLDB Endowment* 11, 6 (2018).

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[44] Yifan Lei, Qiang Huang, Mohan Kankanhalli, and Anthony KH Tung. 2020. Locality-sensitive hashing scheme based on longest circular co-substring. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2589–2599.

[45] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[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 ACM SIGMOD International Conference on Management of Data*. 2539–2554.

[47] Hui Li, Shiyuan Deng, Xiao Yan, Xiangyu Zhi, and James Cheng. 2025. SAQ: Pushing the Limits of Vector Quantization through Code Adjustment and Dimension Segmentation. *Proceedings of the ACM on Management of Data* 3, 6 (2025), 1–25.

[48] Wuchao Li, Chao Feng, Defu Lian, Yuxin Xie, Haifeng Liu, Yong Ge, and Enhong Chen. 2023. Learning balanced tree indexes for large-scale vector retrieval. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1353–1362.

[49] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge*

and Data Engineering 32, 8 (2019), 1475–1488.
[50] Ying Li, Jiuqi Wei, Ziyu Fei, Yufan Fu, and Xiaodong Lee. 2024. DiSAuth: A DNS-based secure authorization framework for protecting data decoupled from applications. *Computer Networks* 254 (2024), 110774.
[51] Zhenyu Liao, Romain Couillet, and Michael W Mahoney. 2021. A Random Matrix Analysis of Random Fourier Features: Beyond the Gaussian Kernel, a Precise Phase Transition, and the Corresponding Double Descent. *Journal of Statistical Mechanics: Theory and Experiment* 2021, 12 (2021), 124006. arXiv:2006.05013 doi:10.1088/1742-5468/ac3a77
[52] Zhenyu Liao and Michael W. Mahoney. 2025. Random Matrix Theory for Deep Learning: Beyond Eigenvalues of Linear Models. arXiv:2506.13139 [stat] doi:10.48550/arXiv.2506.13139
[53] Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. 2024. RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval. arXiv:2409.10516 [cs.LG] https://arxiv.org/abs/2409.10516
[54] Cosme Louart, Zhenyu Liao, and Romain Couillet. 2018. A Random Matrix Approach to Neural Networks. *Annals of Applied Probability* 28, 2 (2018), 1190–1248. doi:10.1214/17-aap1328
[55] 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.
[56] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-throughput vector similarity search in knowledge graphs. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.
[57] Chengmei Niu, Zhenyu Liao, Zenan Ling, and Michael W. Mahoney. 2025. Fundamental Bias in Inverting Random Sampling Matrices with Application to Sub-sampled Newton. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, 46649–46692.
[58] Mohammad Norouzi and David J Fleet. 2013. Cartesian k-means. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 3017–3024.
[59] Themis Palpanas. 2015. Data Series Management: The Road to Big Sequence Analytics. *SIGMOD Record* (2015).
[60] Themis Palpanas and Volker Beckmann. 48(3), 2019. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGREC* (48(3), 2019).
[61] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. Acorn: Performant and predicate-agnostic search over vector embeddings and structured data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[62] Marco Patella and Paolo Ciaccia. 2008. The many facets of approximate similarity search. In *First International Workshop on Similarity Search and Applications (sisap 2008)*. IEEE, 10–21.
[63] Marco Patella and Paolo Ciaccia. 2009. Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms* 7, 1 (2009), 36–48.
[64] Debashis Paul. 2007. Asymptotics of Sample Eigenstructure for a Large Dimensional Spiked Covariance Model. *Statistica Sinica* 17, 4 (2007), 1617–1642. jstor:24307692
[65] Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2020. Messi: In-memory data series indexing. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 337–348.
[66] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 291–324.
[67] Chuxu Song, Zhencan Peng, Jiuqi Wei, and Chuanhui Yang. 2026. CSAttention: Centroid-Scoring Attention for Accelerating LLM Inference.
[68] Ji Sun, Guoliang Li, and Nan Tang. 2021. Learned cardinality estimation for similarity queries. In *Proceedings of the 2021 International Conference on Management of Data*. 1745–1757.
[69] Yukihiro Tagami. 2017. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 455–464.
[70] Yao Tian, Ziyang Yue, Ruiyuan Zhang, Xi Zhao, Bolong Zheng, and Xiaofang Zhou. 2023. Approximate Nearest Neighbor Search in High Dimensional Vector Databases: Current Research and Future Directions. *IEEE Data Engineering Bulletin* 47, 3 (2023).
[71] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2023. DB-LSH 2.0: Locality-Sensitive Hashing With Query-Based Dynamic Bucketing. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[72] Sairaj Voruganti and M Tamer Özsu. 2025. MIRAGE-ANNS: Mixed Approach Graph-based Indexing for Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–27.
[73] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 international conference on management of data*. 2614–2627.
[74] 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. *Advances in Neural Information*

Processing Systems 36 (2023), 15738–15751.

[75] 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. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–27.

[76] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment* 14, 11 (2021), 1964–1978.

[77] Qitong Wang, Ioana Ileana, and Themis Palpanas. 2025. LeaFi: Data Series Indexes on Steroids with Learned Filters. *Proc. ACM Manag. Data* (2025).

[78] Qitong Wang and Themis Palpanas. 2021. Deep learning embeddings for data series similarity search. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1708–1716.

[79] Zeyu Wang, Peng Wang, Themis Palpanas, and Wei Wang. 2023. Graph- and Tree-based Indexes for High-dimensional Vector Similarity Search: Analyses, Comparisons, and Future Directions. *IEEE Data Eng. Bull.* 47, 3 (2023), 3–21.

[80] Zeyu Wang, Qitong Wang, Xiaoxing Cheng, Peng Wang, Themis Palpanas, and Wei Wang. 2024. ***Steiner***-Hardness: A Query Hardness Measure for Graph-Based ANN Indexes. *PVLDB* (2024).

[81] Zeyu Wang, Qitong Wang, Peng Wang, Themis Palpanas, and Wei Wang. 2023. Dumpy: A Compact and Adaptive Index for Large Data Series Collections. *Proc. ACM Manag. Data* 1, 1 (2023), 111:1–111:27.

[82] Zeyu Wang, Qitong Wang, Peng Wang, Themis Palpanas, and Wei Wang. 2024. DumpyOS: A data-adaptive multi-ary index for scalable data series similarity search. *The VLDB Journal* (2024), 1–25.

[83] Zeyu Wang, Haoran Xiong, Qitong Wang, Zhenying He, Peng Wang, Themis Palpanas, and Wei Wang. 2024. Dimensionality-Reduction Techniques for Approximate Nearest Neighbor Search: A Survey and Evaluation. *IEEE Data Eng. Bull.* 48, 3 (2024), 63–80. http://sites.computer.org/debull/A24sept/p63.pdf

[84] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3152–3165.

[85] Jiuqi Wei, Xiaodong Lee, Yufan Fu, Ying Li, and Botao Peng. 2025. Dominate data by yourself: a decentralized scheme for data interoperation when data is decoupled from applications. *World Wide Web* 28, 3 (2025), 1–39.

[86] Jiuqi Wei, Xiaodong Lee, Zhenyu Liao, Themis Palpanas, and Botao Peng. 2025. Subspace collision: an efficient and accurate framework for high-dimensional approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–29.

[87] Jiuqi Wei, Xiaodong Lee, Botao Peng, Quanqing Xu, Chuanhui Yang, and Themis Palpanas. 2026. PDET-LSH: Scalable In-Memory Indexing for High-Dimensional Approximate Nearest Neighbor Search with Quality Guarantees. *IEEE Transactions on Knowledge and Data Engineering* (2026).

[88] Jiuqi Wei, Ying Li, Yufan Fu, Youyi Zhang, and Xiaodong Li. 2023. Data Interoperating Architecture (DIA): Decoupling Data and Applications to Give Back Your Data Ownership. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 438–447.

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

[90] Jiuqi Wei, Quanqing Xu, and Chuanhui Yang. 2026. The Virtuous Cycle: AI-Powered Vector Search and Vector Search-Augmented AI. arXiv:2603.09347 [cs.DB] https://arxiv.org/abs/2603.09347

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

[92] Yuexuan Xu, Jianyang Gao, Yutong Gou, Cheng Long, and Christian S Jensen. 2024. irangegraph: Improvising range-dedicated graphs for range-filtering nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–26.

[93] Djamel Edine Yagoubi, Reza Akbarinia, Florent Masseglia, and Themis Palpanas. 2020. Massively Distributed Time Series Indexing and Querying. *IEEE Trans. Knowl. Data Eng.* 32, 1 (2020), 108–120.

[94] Zhenkun Yang, Chuanhui Yang, Fusheng Han, Mingqiang Zhuang, Bing Yang, Zhifeng Yang, Xiaojun Cheng, Yuzhong Zhao, Wenhui Shi, Huafeng Xi, et al. 2022. OceanBase: a 707 million tpmC distributed relational database system. *Proceedings of the VLDB Endowment* (2022).

[95] Ximu Zeng, Liwei Deng, Penghao Chen, Xu Chen, Han Su, and Kai Zheng. 2025. LIRA: A Learning-based Query-aware Partition Framework for Large-scale ANN Search. In *Proceedings of the ACM on Web Conference 2025*. 2729–2741.

[96] Aiyao Zhang, Xiaodong Lee, Zhixian Zhuang, Jiuqi Wei, Yufan Fu, and Botao Peng. 2025. POLARIS: Cross-Domain Access Control via Verifiable Identity and Policy-Based Authorization. *arXiv preprint arXiv:2511.22017* (2025).

[97] Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–30.

[98] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023).

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

[100] Bolong Zheng, Zhao Xi, Lianggui Weng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S Jensen. 2020. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. *Proceedings of the VLDB Endowment* 13, 5 (2020), 643–655.

[101] 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 *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 3209–3221.

[102] Zhixian Zhuang, Xiaodong Lee, Jiuqi Wei, Yufan Fu, and Aiyao Zhang. 2024. CBCMS: a compliance management system for cross-border data transfer. In *BigData*.

[103] Zhixian Zhuang, Xiaodong Lee, Aiyao Zhang, Jiuqi Wei, Yufan Fu, and Botao Peng. 2026. Structured policy modeling and context-aware generation for multi-jurisdictional compliance in global software systems. *Information and Software Technology* (2026), 108041.

[104] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: segment graph for range-filtering approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–26.

Received October 2025; revised January 2026; accepted February 2026