# FAVOR: Efficient Filter-Agnostic Vector ANNS Based on Selectivity-Aware Exclusion Distances

JUNJIE SONG, Huazhong University of Science and Technology, China
YU LIU*, Huazhong University of Science and Technology, China and National University of Singapore, Singapore
GUOYU HU, National University of Singapore, Singapore
ZHONGLE XIE, Zhejiang University, China
MING YANG, Wuhan Technical University, China and National University of Singapore, Singapore
BENG CHIN OOI, Zhejiang University, China
KE ZHOU, Huazhong University of Science and Technology, China

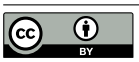
Modern retrieval systems increasingly require integrating approximate nearest neighbor search (ANNS) with complex attribute filtering to handle hybrid queries in applications such as recommendation systems and retrieval-augmented generation (RAG). While HNSW-based inline-filtering methods show promise, existing approaches struggle to deliver high throughput under low-selectivity scenarios while balancing search efficiency, filtering generality, and index connectivity. To address these challenges, we propose FAVOR, an efficient filter-agnostic vector ANNS that supports arbitrary filtering conditions while maintaining stable performance across varying selectivity levels. FAVOR introduces three novel features: (1) an integrated architecture that unifies selectivity estimation and filtered ANNS execution, providing a cohesive solution for hybrid vector–attribute queries; (2) a HNSW-based inline-filtering algorithm that introduces an exclusion distance mechanism to dynamically reshape the vector distance distribution, pushing non-target vectors away from the query while promoting valid candidates toward the query, thus improving search efficiency without compromising generality or graph connectivity; and (3) a selectivity-driven search selector that estimates query selectivity and dynamically routes queries between a pre-filtering brute-force algorithm for low-selectivity cases and an optimized HNSW-based search algorithm for other scenarios, ensuring consistent performance. Extensive experiments on real-world datasets demonstrate that FAVOR achieves a 1.3–5× higher QPS at *Recall*@10 = 95% compared to state-of-the-art methods for arbitrary filtering conditions, while maintaining competitive performance even against tailored solutions in some filtering conditions.

CCS Concepts: • **Information systems** → **Top-k retrieval in databases**; **Similarity measures**; *Query processing*; *Data structures*.

Additional Key Words and Phrases: Approximate nearest neighbor search, Vector databases, AIxDB

*Corresponding author.

Authors' Contact Information: Junjie Song, Huazhong University of Science and Technology, China, d202381493@hust.edu.cn; Yu Liu, Huazhong University of Science and Technology, China, liu_yu@hust.edu.cn and National University of Singapore, Singapore; Guoyu Hu, National University of Singapore, Singapore, guoyu.hu@u.nus.edu; Zhongle Xie, Zhejiang University, China, xiezl@zju.edu.cn; Ming Yang, Wuhan Technical University, China and National University of Singapore, Singapore, mingyang@wtc.edu.cn; Beng Chin Ooi, Zhejiang University, China, ooibc@zju.edu.cn; Ke Zhou, Huazhong University of Science and Technology, China, zhke@hust.edu.cn.




ACM 2836-6573/2026/6-ART183
https://doi.org/10.1145/3802060

**ACM Reference Format:**
Junjie Song, Yu Liu, Guoyu Hu, Zhongle Xie, Ming Yang, Beng Chin Ooi, and Ke Zhou. 2026. FAVOR: Efficient Filter-Agnostic Vector ANNS Based on Selectivity-Aware Exclusion Distances. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 183 (June 2026), 25 pages. https://doi.org/10.1145/3802060

## 1 INTRODUCTION

Recent advances in deep learning have enabled the encoding of rich semantic information from unstructured data into high-dimensional vectors [1, 2]. These embeddings support semantic retrieval through nearest neighbor search in vector space. However, the paradigm often falls short in real-world scenarios where queries involve both semantic relevance and structured constraints. For example, when users search for videos on YouTube, they may provide a textual description while also specifying filters such as upload date or video duration. Such hybrid queries, combining semantic similarity with attribute-based filtering, are increasingly common in recommendation systems, search engines, and retrieval-augmented generation (RAG) applications [3, 4].

Approximate nearest neighbor search (ANNS) utilizing graph-based indexes has gained widespread adoption due to its efficient search and superior retrieval quality [5, 6]. Within this category, Hierarchical Navigable Small World (HNSW) has emerged as the predominant methodology, distinguished by its capacity for incremental construction and relatively few tuning parameters compared with alternative methods [7]. Yet, integrating HNSW with attribute filtering remains challenging because the subset of vectors satisfying filtering conditions often forms a sparsely connected subgraph. Specifically, there are three primary challenges in this domain.

- **Challenge 1 (C1): Supporting arbitrary filtering conditions.** The need for attribute-tailored indexes to support diverse filtering types creates a fundamental tension between query efficiency and the practical costs of index construction and maintenance. For example, tailored approaches [8–10] redesign the index structure to explicitly connect vectors satisfying specific attributes, thereby avoiding unnecessary similarity computations. However, these tailored indexes lack scalability and flexibility, as they cannot efficiently generalize to new filtering conditions.
- **Challenge 2 (C2): Balancing search efficiency and recall.** Preserving graph connectivity requires expensive similarity computations on vectors that violate filtering conditions, making it exceptionally challenging to optimize for both efficiency and recall. Filter-agnostic methods [7, 11] address this by modifying the search process while retaining the index structure, but they often fail to balance optimal path exploration and computational efficiency, leading to substantial performance gaps compared with tailored approaches.
- **Challenge 3 (C3): Maintaining stable performance under varying selectivity.** The inherent fluctuation in query selectivity poses a significant risk to search performance, which can degrade sharply with low-selectivity queries. For instance, under low selectivity, ACORN [11] suffers from connectivity issues, whereas result-set filtering [7] struggles to terminate efficiently due to the cost of finding sufficient filtering-valid vectors closer to the query vector than those in the candidate set.

To tackle these challenges, we propose FAVOR, an efficient Filter-Agnostic VectOR (FAVOR) ANNS. FAVOR seamlessly integrates the estimation of query selectivity with two distinct search algorithms, each adaptive for different selectivity levels. It includes a selectivity-driven search selector, an FAVOR search algorithm, and a pre-filtering brute-first algorithm. By employing a standard proximity graph and a unified query processing workflow, FAVOR supports arbitrary filtering conditions without incurring the construction and maintenance overhead of tailored indexes (addressing **C1**). At the heart of our search algorithm is an exclusion distance mechanism. By dynamically adjusting vector distributions with a selectivity-aware exclusion distance based on filtering compliance and enabling filtering-valid vectors to be closer to the query vector than

their neighbors, it steers the search towards valid results while preserving connectivity, effectively balancing the connectivity and efficiency (addressing **C2**). The integrated selectivity-driven search selector determines the optimal choice between a pre-filtering and FAVOR search algorithm for each query, ensuring stable performance across varying selectivity levels (addressing **C3**). Together, FAVOR enables selectivity-aware, filter-agnostic ANNS with performance on par with or superior to tailored methods.

Our contributions are summarized as follows:

- We propose FAVOR, a filter-agnostic vector ANNS scheme, which integrates a selectivity-driven search selector and two complementary search algorithms. It achieves stable query performance across the full selectivity spectrum, leveraging a pre-filtering brute-force algorithm for low-selectivity queries and an HNSW-based filtered ANNS algorithm for other selectivity levels.
- We devise an inline-filtering search algorithm integrating HNSW with a *exclusion distance* mechanism. This mechanism adaptively repels filtering-violated vectors from the query while promoting valid candidates along the search path, thereby improving efficiency without sacrificing recall.
- Our selectivity-driven search selector estimates query selectivity on the fly, dynamically routing each query to the optimal algorithm. This ensures stable performance under arbitrary filtering conditions by choosing between the pre-filtering brute-force method and our proposed inline-filtering search algorithm.
- We conduct extensive experiments on real-world datasets with various filtering conditions. FAVOR achieves a 1.3–5× improvement in QPS at $Recall@10 = 95\%$ compared to filter-agnostic approaches while exhibiting comparable performance compared to state-of-the-art tailored approaches in some filtering conditions.

We integrate FAVOR into HAKES [12][1], a scalable and modular vector database for supporting AI workloads. FAVOR enables HAKES to efficiently support hybrid vector–attribute queries with arbitrary filtering conditions, providing stable performance across varying query selectivities.

The paper is structured as follows. Section 2 provides the background on Filtered ANNS and state-of-the-art methods. Section 3 discusses the design guidelines and provides an overview of FAVOR. Section 4 and Section 5 detail the core components, the selectivity-driven search selector, and the FAVOR search algorithm. Section 6 evaluates FAVOR. Section 7 reviews related works and Section 8 concludes.

## 2 PRELIMINARY AND BACKGROUND

We first establish the notation and formally define the problem of filtered ANNS. Table 1 lists the key symbols used in this paper.

### 2.1 Definition

*2.1.1 Approximate Nearest Neighbor Search (ANNS).* For a query vector $\boldsymbol{q}$ and a vector dataset $\hat{\mathcal{D}} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}\} \in \mathbb{R}^l$, where $l$ is the dimensionality of the vector space, the purpose of approximate nearest neighbor search is to find the $k$ approximate nearest vectors to $\boldsymbol{q}$. ANNS addresses this task by using index structures (*e.g.*, hash[13–15], tree[16–19], graph[7, 20]) to achieve sublinear search time while trading off minimal accuracy for efficiency gains.

*2.1.2 Filtered ANNS.* Let each vector $\boldsymbol{v_i} \in \hat{\mathcal{D}} (i = 1, 2, \ldots, n)$ be associated with a $m$-dimensional scalar attribute set $A_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\}$, where each $a_{ij} (j = 1, 2, \ldots, m)$ is a scalar value. The vector dataset with attributes can be formally defined as

[1] https://github.com/nusdbsystem/HAKES

Table 1. Key notations.

| Symbol | Description |
|---|---|
| $\hat{\mathcal{D}}$ | Vector set |
| $A$ | Attribute set |
| $\mathcal{D}$ | Vector dataset with $A$ |
| $\mathcal{F}$ | Set of filtering conditions |
| $\boldsymbol{q}$ | Query vector in $\mathbb{R}^l$ |
| $\boldsymbol{v}^T$ / $\boldsymbol{v}^N$ | Vector whose $A \in \mathcal{F}$ / $A \notin \mathcal{F}$ |
| TD / NTD | Set of $(\boldsymbol{v}_i^T, A_i)$ / $(\boldsymbol{v}_j^N, A_j)$ in $\mathcal{D}$ for a $(\boldsymbol{q}, \mathcal{F})$ |
| $\mathcal{V}$ | Visited list: set of vectors already visited |
| $\mathcal{C}$ | Candidate set: set of vectors to be explored |
| $\mathcal{R}$ | Result set: the $ef$ nearest vectors to $\boldsymbol{q}$ in $\mathcal{C}$ |
| $\mathcal{S}$ | Target result set: the $k$ nearest vectors to $\boldsymbol{q}$ in $\mathcal{R}$ |
| $p$ | Selectivity: proportion of TD in $\mathcal{D}$ |

$\mathcal{D} = \{(\boldsymbol{v_1}, A_1), (\boldsymbol{v_2}, A_2), \ldots, (\boldsymbol{v_n}, A_n)\}$. Given a query vector $\boldsymbol{q}$ and a set of filtering conditions $\mathcal{F}$, the target subset is defined as

$$\mathcal{S} = \{(\boldsymbol{v_i}, A_i) \in \mathcal{D} \mid A_i \in \mathcal{F}\}.$$

Filtered ANNS aims to acquire the approximate $k$ nearest vectors to $\boldsymbol{q}$ in $\mathcal{S}$. This paper focuses on filtered ANNS with arbitrary $\mathcal{F}$. Generally, the $\mathcal{F}$ can be categorized into four primary types, including *Equality* [8, 21–24], *Inclusion* [8, 24], *Range* [10, 25–28], and *Logic* [9, 11, 29, 30].

### 2.2 Hierarchical Navigable Small World (HNSW)

HNSW [7] is an efficient graph-based indexing approach. It constructs a multi-layered proximity graph that supports logarithmic-scale search complexity. The index comprises $L + 1$ layers (from layer $L$ down to layer 0), where higher layers contain sparsely connected nodes functioning as "highways" across the data space, while the base layer (layer 0) maintains detailed local neighborhood structures for precise retrieval.

During the index construction phase, each vector is inserted into the base layer and added to higher layers with exponentially decreasing probability. At each layer, it connects to its approximate nearest neighbors. Specifically, for a vector to insert $\boldsymbol{v}$, the topmost layer level $l$ is sampled from an exponentially decaying probability distribution. Then, starting from an entry point designated at the top layer, greedy search is performed to locate the nearest vectors to $\boldsymbol{q}$, and the nearest $\boldsymbol{v}$ serves as the entry point for the next layer. Iteratively, the neighbors at layer $l$ to $L$ are identified and connections are built with $\boldsymbol{v}$ being inserted to those layers.

During the query search phase, the search starts from a designated entry point at the top layer and navigates downward through successive layers, progressively refining a candidate set, $C$, until reaching the base layer, where the final $k$ nearest neighbors are identified. Specifically, the algorithm executes $\mathit{GreedySearch}(\boldsymbol{q}, 1, ep, \mathcal{G})$ at each non-base layer to iteratively select the optimal entry node $ep$ for transitioning to the subsequent layer. Once reaching the base layer, the algorithm transitions from single-node selection to $\mathit{GreedySearch}(\boldsymbol{q}, ef, ep, \mathcal{G})$, where $ef > k$ is to ensure comprehensive exploration of the vector space. Finally, the top-$k$ vectors nearest to the $\boldsymbol{q}$ in $\mathcal{R}$ are selected as the query result.

In addition to the concepts presented in Table 1, we focus on two key concepts of HNSW that are crucial for our analysis and the improvement of existing limitations.

**Algorithm 1:** GreedySearch($\boldsymbol{q}, ef, ep, \mathcal{G}$)

**Input:** query vector $\boldsymbol{q}$, entry point $ep$, graph index $\mathcal{G}$
**Output:** result set of $ef$ approximate nearest neighbors $\mathcal{R}$

$\mathcal{V} \leftarrow ep$ ; // visited list
$C \leftarrow ep$ ; // candidate set
$\mathcal{R} \leftarrow ep$ ; // result set
**while** $C \neq \emptyset$ **do**
  $v_\alpha \leftarrow \arg\min_{v \in C} Dis(\boldsymbol{q}, v)$;
  $v_\beta \leftarrow \arg\max_{v \in \mathcal{R}} Dis(\boldsymbol{q}, v)$;
  **if** $Dis(\boldsymbol{q}, v_\alpha) > Dis(\boldsymbol{q}, v_\beta)$ **then**
    **break**;
  **foreach** $v \in$ *neighbors of* $v_\alpha$ **do**
    **if** $v \notin \mathcal{V}$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \bigcup v$;
      $v_\beta \leftarrow \arg\max_{v \in \mathcal{R}} d(\boldsymbol{q}, v)$;
      **if** $(Dis(\boldsymbol{q}, v) < Dis(\boldsymbol{q}, v_\beta)) \parallel (|\mathcal{R}| < ef)$ **then**
        $C \leftarrow C \bigcup v$;
        $\mathcal{R} \leftarrow \mathcal{R} \bigcup v$;
        **if** $|\mathcal{R}| > ef$ **then**
          $\mathcal{R} \leftarrow \mathcal{R} \setminus \arg\max_{v \in \mathcal{R}} Dis(\boldsymbol{q}, v)$;
**return** $\mathcal{R}$;

Table 2. Summary table w.r.t filtered ANNS solutions.

| Methods | Graph Type | Filter Support | | | | Vectors on search path | Termination efficiency | General QPS | QPS under low selectivity |
|---|---|---|---|---|---|---|---|---|---|
| | | Equality | Inclusion | Range | Logic | | | | |
| **SeRF[10]** | Attribute-tailored | × | × | ✓ | × | TD | High | High | High |
| **UNG[8]** | Attribute-tailored | ✓ | ✓ | × | × | TD | High | High | High |
| **ACORN-$\gamma$[11]** | Dense-Conventional | ✓ | ✓ | ✓ | ✓ | TD | Medium | Medium | Low |
| **RSF[7]** | Conventional | ✓ | ✓ | ✓ | ✓ | TD & NTD | Low | Medium | Low |
| **FAVOR** | Conventional | ✓ | ✓ | ✓ | ✓ | TD & NTD | High | High | High |

- **Search Path**: the sequence of data selected to extend the traversal path during the search process, starting from an entry point, along with the data points whose vectors are closer to the $\boldsymbol{q}$ compared to their neighbor vectors.
- **Termination Condition**: the criterion for the termination of the search process. In HNSW, termination is triggered when all vectors in $C$ are farther from $\boldsymbol{q}$ than all vectors in $\mathcal{R}$, or when $C$ is empty.

### 2.3 Variants for filtered ANNS

Typically, the search process of HNSW could be extended with an inline-filtering approach to support filtered ANNS without modifying the graph construction process. Meanwhile, there also exist solutions that integrate the attributes with the proximity graph to achieve high efficiency.

*2.3.1 Representative Approaches.* We summarize four representative variants of HNSW-based inline-filtering approaches that inspired our work. Approaches such as *UNG* [8] and *SeRF* [10] support filtering for specialized condition sets $\mathcal{F}$, while *ACORN* [11] and *Result-Set-Filtering* [7] enable filtering for arbitrary $\mathcal{F}$. All of them adopt termination conditions similar to those in HNSW.

- **UNG** constructs a directed acyclic Label Navigation Graph (LNG) to organize label semantics. It builds a local proximity graph for each label set and interconnects them into a global index

through cross-group edges derived from LNG containment relationships. At query time, UNG locates the minimal superset covering the query label set within the LNG and initiates graph traversal from the corresponding vector nodes. Guided by cross-group edges, the traversal is restricted to the subgraph that satisfies the filter condition, ensuring that $C$ and $\mathcal{R}$ consist solely of TD.

- **SeRF** constructs an HNSW graph by inserting nodes in ascending order of attribute values and assigning validity intervals to edges using insertion or deletion timestamps. This design compresses multiple range-specific indexes into a single unified structure. For a given query, SeRF dynamically prunes edges whose validity intervals fall outside the query's range boundaries, forming a tailored subgraph for efficient filtered traversal. Consequently, its $C$ and $\mathcal{R}$ all contain only TD.
- **ACORN-$\gamma$** performs filtered ANNS by augmenting a dense HNSW index. During index construction, it introduces additional neighbors for each data point to enhance connectivity. At query time, the algorithm filters out NTD neighbors but computes distances only between $\boldsymbol{v}^T$ and $\boldsymbol{q}$. The TD point with the shortest distance extends the search path, effectively forming a subgraph composed entirely of TD.
- **RSF** is identical to HNSW in index construction and termination conditions. The only difference is that it only permits TD to enter $\mathcal{R}$ during the query search process.

*2.3.2 Discussion and Rethinking.* Table 2 summarizes the performance ratings and search strategies of representative methods. Several consistent observations can be drawn. By comparing the columns "Graph Type" and "Filter Support", it becomes evident that attribute-tailored graphs generally achieve better performance but are limited to specific types of filtering conditions. This limitation arises because the topological structures of these tailored indexes vary across filter types, preventing them from generalizing to other filtering conditions. Consequently, UNG cannot support range filtering, while SeRF is only applicable to range-based filters. In contrast, ACORN-$\gamma$ and RSF, which employ conventional graph structures, are capable of handling arbitrary filtering conditions. However, this generality comes at the cost of lower efficiency compared to tailored methods.

It can also be seen from the table that most approaches adopt a TD-preferred strategy, where distance computations are performed only between $\boldsymbol{v}^T$ and $\boldsymbol{q}$, and only TD are collected into $C$. While this design reduces redundant similarity evaluations, it also requires a tailored index structure to maintain sufficient graph connectivity. Without such structural support, connectivity becomes fragile, leading to inefficiency in search. For instance, ACORN-1, an extreme setting for ACORN-$\gamma$, often encounters cases where no TD exist within the neighborhood or second-order neighborhood of the current node. Although this issue could be alleviated by increasing node out-degrees in ACORN-$\gamma$, it introduces additional overhead in both index construction and neighbor exploration. In contrast, RSF avoids connectivity loss by always extending the search path toward the nearest neighbor to $\boldsymbol{q}$, regardless of whether the neighbor is TD or NTD. However, since $C$ is frequently populated with NTD close to $\boldsymbol{q}$ while $\mathcal{R}$ accepts only TD, RSF struggles to terminate efficiently. It must locate TD sufficiently close to $\boldsymbol{q}$, leading to unnecessary distance computations for NTD. Thus, the tradeoff between the graph connectivity and efficiency is a crucial design point for HNSW-based solutions to filtered ANNS.

The table also reveals an interesting phenomenon: the absence of attribute-tailored index structures leads to unstable performance when selectivity is low. In practical scenarios, queries are continuously issued with diverse filtering conditions, leading to highly variable selectivity levels. When a low-selectivity query triggers throughput degradation, it can propagate delays throughout the request queue, causing a sharp rise in end-to-end latency and severely impairing user experience.

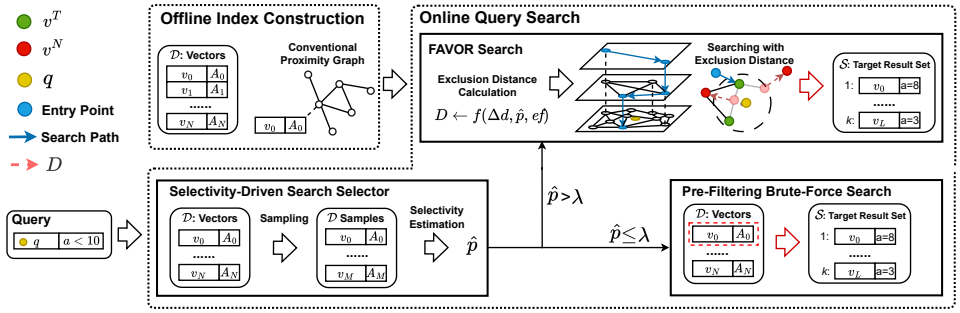



Fig. 1. Workflow of FAVOR.

This highlights the necessity of addressing low-selectivity robustness as a fundamental requirement for real-world ANNS systems.

In summary, while each search strategy demonstrates strengths under certain conditions, ensuring stable and efficient performance across dynamic query selectivity and variant filter types remains an open research problem that motivates our work.

## 3 FAVOR

We present the design guidelines and the overview of our work in this section.

### 3.1 Key idea

We aim to develop an efficient filtered ANNS method that supports arbitrary $\mathcal{F}$ while maintaining high QPS across the variant selectivity range.

*3.1.1 Key Idea.* Our key idea is that high performance for filtered ANNS within the HNSW framework can be achieved by increasing the number of TD points that are closer to $\boldsymbol{q}$ than their neighbors on the search path. Existing approaches either construct tailored graph structures (*e.g.*, UNG [8] and SeRF [10]) or modify search strategies (*e.g.*, ACORN [11]) to favor TD points. Tailored structures explicitly connect TD points and ensure traversal only among them, thereby maximizing proximity to $\boldsymbol{q}$ but sacrificing flexibility across different filtering conditions. Conversely, passive search modification strategies, such as those used in ACORN, force the selection of TD points to extend the search path and enlarge node out-degrees to increase TD encounter probability. However, they cannot ensure that TD points are actually closer to $\boldsymbol{q}$ than their neighbors, leading to insufficient exploration and degraded accuracy.

Our approach aims to balance two essential objectives: actively prioritizing TD points during search and comprehensively exploring the query neighborhood, while preserving full flexibility for arbitrary filtering conditions. The key lies in dynamically reshaping the distance distribution during search to make TD points closer to $\boldsymbol{q}$ than their neighbors, thereby achieving high performance without sacrificing generality.

*3.1.2 Design Guidelines.* Building upon the discussion and rethinking in Section 2.3, we summarize several key design guidelines to guide the development of our method:

- **G.1:** Employ a general proximity graph to organize data and perform query search, ensuring adaptability to arbitrary filtering conditions.
- **G.2:** Compute distances between all neighbors and $\boldsymbol{q}$ so that the search path always extends through the data point nearest to $\boldsymbol{q}$.
- **G.3:** Allow NTD points to enter $\mathcal{R}$ to enable accurate and efficient termination decisions.
- **G.4:** Identify the selectivity of each query and adapt the search strategy to improve efficiency in low-selectivity scenarios.

### 3.2 Design of FAVOR

We propose FAVOR based on the guidelines and our key idea. It estimates selectivity and selects a suitable search strategy correspondingly (**G.4**). Based on the standard HNSW graph and query search strategy, it requires no knowledge of queries beforehand and supports arbitrary $\mathcal{F}$ at predetermined recall rates (**G.1-G.3**). Note that it enhances the search algorithm by dynamically making TD closer to the $\boldsymbol{q}$ compared to their neighbors.

*3.2.1 Architecture.* As shown in Figure 1, FAVOR follows the two-phase search paradigm as vanilla HNSW: offline index construction and online query search. The offline phase prepares a general proximity graph without any filtering-specific design, while the online phase adaptively executes one of two search algorithms based on the estimated query selectivity. The core of FAVOR lies in its selectivity-driven search strategy and the two complementary algorithms that ensure stable, high-performance ANNS across diverse filtering conditions.

- *Selectivity-driven search selector.* The selector first conducts a sampling on the vector dataset $D$ and then estimates the query selectivity. It determines the appropriate search mode to balance efficiency and accuracy. The estimation and selection strategy details are described in Section 4.
- *FAVOR search algorithm.* The algorithm is an HNSW-based inline-filtering ANNS method with dynamic distance computation. An exclusion distance mechanism achieves this dynamic computation during the search process. FAVOR search algorithm is activated to perform a focused and efficient vector search over the most relevant index regions when the query exhibits qualified selectivity. The details of the search process are presented in Section 5.
- *Pre-filtering brute-force search algorithm.* The algorithm is a linear scanning search method. That is, a pre-filtering step narrows the candidate space before applying a lightweight brute-force search, ensuring comprehensive yet efficient retrieval for low-selectivity queries. The algorithm will be introduced in Section 4.

*3.2.2 Processing Workflow.* We explain the processing workflow of FAVOR in its two phases below.

In the offline index construction phase, FAVOR builds an index over the vectors in $\mathcal{D}$ using a conventional graph connectivity algorithm [31]. Each vector is stored together with its attribute set $A$ to support efficient inline filtering at query time. During construction, FAVOR records the average pairwise distance between vectors, denoted as $\Delta d$ (see Section 5.3), which captures the intrinsic distance distribution of $\mathcal{D}$. This statistic is stored in the index metadata and later used to compute exclusion distances and guide search path optimization during online query processing. Note that this phase is performed entirely offline and does not rely on any prior query information.

In the online query search phase, the incoming query $\boldsymbol{q}$ and its filtering condition $\mathcal{F}$ are first processed by the selectivity-driven search selector. This component estimates the query selectivity $\hat{p}$ based on $\mathcal{F}$ (see Section 4.2) and routes the query according to a predefined threshold $\lambda$ (see Section 4.1). If $\hat{p} < \lambda$, the system executes Pre-Filtering Brute-Force Search, which performs a lightweight scan over a reduced candidate set to return the top-$k$ results. Otherwise, the FAVOR search algorithm is activated. FAVOR computes the exclusion distance using the pre-recorded $\Delta d$ and the estimated $\hat{p}$ (see Section 5.3). It then performs a guided traversal over the HNSW index based on this exclusion distance until the termination criteria are met (see Section 5.4), after which the top-$k$ results are returned.

## 4 SELECTIVITY-DRIVEN SEARCH SELECTOR

### 4.1 Selector Design by Selectivity

HNSW-based methods are highly sensitive to query selectivity $p$. When $p$ is high, a dense distribution of TD allows the navigation-based search mechanism to efficiently locate the top-$k$ data

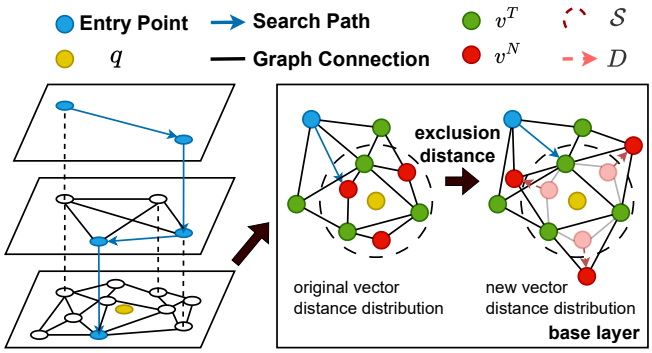



Fig. 2. FAVOR search process.

points nearest to the $\boldsymbol{q}$. However, when $p$ is low, TD becomes sparsely distributed, causing the search to diverge into irrelevant regions of the graph. This leads to numerous ineffective visits to NTD and severely reduced search efficiency. In contrast, the pre-filtering brute-force approach (PreFBF), which applies filtering before similarity computation, could perform more effectively under low-selectivity conditions [10].

To adaptively select the two search strategies, we introduce a selectivity threshold $\lambda$ that determines when to use PreFBF or the HNSW-based algorithm (see Section 5). A conservative threshold ensures that PreFBF is chosen when selectivity falls below $\lambda$, while the HNSW-based algorithm is used otherwise. We determine the $\lambda = 1\%$ based on empirical measurement. We refer to the interval in which $p$ is close to but slightly exceeds 1% as the middle selectivity range. Since it is a less representative scenario [8, 11], we adopt a conservative retrieval strategy that prioritizes graph-based search methods because they incur lower potential error costs and thus provide more robust retrieval quality (see Section 6.2).

### 4.2 Selectivity Estimation

When a query $q$ arrives, we estimate $p$ first to determine the corresponding search strategy. A straightforward way to obtain accurate selectivity is to scan the entire dataset and count TD points. Yet, it is costly and cannot scale with the dataset size. Given that the estimation is an online operation, FAVOR prioritizes efficiency and generality over accuracy and employs a sampling approach. Given a query, we collect the proportion of TD points in a randomly drawn subset from the dataset and express it by $\hat{p}$. To assess the reliability of the estimation method, we discuss the relative error of the random sampling approach. Since the number of TD points follows a hyper-geometric distribution [32], the relative error of $\hat{p}$ is

$$\textit{Relative Error} = \frac{\sqrt{\mathrm{Var}(\hat{p})}}{p} = \sqrt{\frac{(1-p)}{np}\left(1-\frac{n}{N}\right)}, \tag{1}$$

where $n$ is the sample size and $N$ is the size of the total dataset. The relative error decreases as the sample size $n$ and selectivity $p$ increase. For million-scale datasets, the relative error remains around 1% even when $p \approx 1\%$ with a sampling rate of $n/N = 1\%$. The error only becomes significant when $p << 1\%$, for which the strategy selects PreFBF as the search strategy based on $\lambda = 1\%$. Since PreFBF's search process does not require $\hat{p}$, it does not affect search performance.

## 5 FAVOR SEARCH ALGORITHM

In this section, we detail the basic idea of FAVOR search algorithm, the filtered search workflow, the determination of exclusion distance, and the optimization of termination conditions.

### 5.1 Basic Idea

Existing HNSW-based filter-agnostic ANNS methods treat the inter-vector distance distribution as static during search, preventing them from achieving the essential goal of making TD appear closer to the query vector $\boldsymbol{q}$ than their NTD neighbors. While tailored approaches explicitly reshape this distribution through index reconstruction during the pre-search process, such modifications are infeasible for filter-agnostic designs. Therefore, our idea is to dynamically adjust the distribution during the retrieval process itself to support different filter conditions. The key intuition is to perform real-time distance correction for each visited vector based on its compliance with the filtering condition, thereby continuously reshaping the target distance distribution to emulate the behavior of tailored indexes. As the central innovation of this work, we introduce an exclusion distance mechanism that penalizes non-target data (NTD), enabling this adaptive correction in an online manner.

*5.1.1 Definition.* In our approach, the distance function $Dis(\boldsymbol{q}, \boldsymbol{v})$ represents the Euclidean distance between two vectors. For filtered ANNS, we first utilize $\mathcal{F}$ to distinguish TD and NTD. To reduce the impact of NTD during search, we impose an exclusion distance that pushes NTD farther from $\boldsymbol{q}$, thereby increasing the likelihood that nearby TD points become the top candidates. Formally, the distance function for the exclusion distance mechanism is formulated as

$$\overline{Dis}\,(\boldsymbol{q}, \boldsymbol{v}) = \begin{cases} Dis(\boldsymbol{q}, \boldsymbol{v}^T), & A \in \mathcal{F}, \\ Dis(\boldsymbol{q}, \boldsymbol{v}^N) + D, & A \notin \mathcal{F}, \end{cases} \tag{2}$$

where $\overline{Dis}(\cdot)$ denotes the adjusted distance after distribution correction, $D$ is the exclusion distance, $\boldsymbol{v}$ represents a visited data point, and $A$ denotes its associated attribute set.

*5.1.2 Illustration.* As illustrated in the base layer of Figure 2, under the original vector distance distribution, a NTD data point ($\boldsymbol{v}^N$), colored in red, is likely to be selected to extend the search path. After applying the exclusion distance, however, the NTD points are effectively pushed farther from the query $\boldsymbol{q}$, making a TD point ($\boldsymbol{v}^T$) more likely to be selected instead. This adjustment brings TD points closer to $\boldsymbol{q}$ than their neighboring NTD, enabling more efficient search expansion and potentially yielding higher query throughput (QPS). Building on this intuition, we develop FAVOR search algorithm, which operates on the standard HNSW graph without modifying its construction process. Instead, our algorithm dynamically redefines the distances between $\boldsymbol{q}$ and visited data points during search and refines the termination condition, thereby significantly enhancing search efficiency.

### 5.2 Search Workflow

Based on the offline-built index, the overall search workflow is depicted in Figure 2 and Algorithm 2. The process starts from the entry point at the top layer and proceeds layer by layer, where a greedy search algorithm identifies the data point nearest to the query $\boldsymbol{q}$ within the current layer. This point then serves as the entry point for the next lower layer. During this hierarchical descent, attribute filtering is temporarily omitted to enable rapid convergence toward the region near $\boldsymbol{q}$. Once the entry point reaches the bottom layer, the *OptiGreedySearch* procedure (Algorithm 3) is invoked to populate $C$ and $\mathcal{R}$. Finally, the top-$k$ TD points nearest to $\boldsymbol{q}$ are selected from $\mathcal{R}$ to construct the $\mathcal{S}$.

Algorithm 3 describes the process of how FAVOR implements search using the exclusion distance at the base layer. The algorithm employs an array, a min-heap, and a max-heap to maintain $\mathcal{V}$, $C$,

**Algorithm 2:** FAVORSearch($q, p, k, \mathcal{G}$)

**Input:** query vector $\boldsymbol{q}$, selectivity $p$, graph index $\mathcal{G}$
**Output:** result set of $k$ approximate nearest neighbors

$\mathcal{R} \leftarrow \emptyset$ ; // result set
$\mathcal{S} \leftarrow \emptyset$ ; // target result set
$ep \leftarrow$ entry point in $\mathcal{G}$;
$L \leftarrow$ number of layers in $\mathcal{G}$;
**for** $l = L - 1$ **to** 1 **do**
  $\mathcal{R} \leftarrow \text{GreedySearch}(\boldsymbol{q}, 1, ep, \mathcal{G}_l)$;
  $ep \leftarrow \arg\min_{\boldsymbol{v} \in \mathcal{R}} Dis(\boldsymbol{q}, \boldsymbol{v})$;
$\mathcal{R} \leftarrow \text{OptiGreedySearch}(\boldsymbol{q}, \mathcal{F}, ef, ep, \mathcal{G}_0)$;
$\mathcal{S} \leftarrow k$ nearest vectors to $\boldsymbol{q}$ in $\mathcal{R}$;
**return** $\mathcal{S}$;

and $\mathcal{R}$, respectively, while initializing a count of TD in $\mathcal{R}$. Note that the order of vectors in $C$ and $\mathcal{R}$ is maintained according to the vector distance distribution with an added exclusion distance. Consequently, when a data point enters them, it is first checked against $\mathcal{F}$, then the distance is calculated based on Equation (2).

In the beginning, the entry point enters into $\mathcal{V}$, $C$ and $\mathcal{R}$. When $C$ is not empty, $C$'s root node, *i.e.*, the nearest data point to $\boldsymbol{q}$ in $C$ under the new vector distance distribution, is removed from $C$ and used to extend the search path. If $C$'s root node is farther from $\boldsymbol{q}$ compared to $C$'s root node, *i.e.*, the farthest data point to $\boldsymbol{q}$ in $\mathcal{R}$ under the new vector distribution, while the count of TD in $\mathcal{R}$ is more than $ef/2$, the search process terminates. The top-$k$ nearest vectors to $\boldsymbol{q}$ in $\mathcal{R}$ are the results. Algorithm 3 (line 1-9) outlines the above process.

When the termination condition is not satisfied, the algorithm begins to traverse the neighbors of the point used to extend the path. If a neighbor hasn't been previously visited, it is added to $\mathcal{V}$. Then, it is added to $C$ and $\mathcal{R}$ if it is closer to $\boldsymbol{q}$ than the current root of $\mathcal{R}$, or $\mathcal{R}$ is not full. In this case, if its attributes meet $\mathcal{F}$, the count of TD in $\mathcal{R}$ adds 1. Furthermore, when the population operation makes the size of $\mathcal{R}$ beyond $ef$, the root node of $\mathcal{R}$ is removed from $\mathcal{R}$. If this removed data point's attributes meet $\mathcal{F}$, the count of TD in $\mathcal{R}$ subtracts 1. The above process iterates until all neighbors have been explored. Algorithm 3 (line 10-24) outlines the above process. During this process, we can enable more TD points to pave the search path.

### 5.3 Exclusion Distance Determination

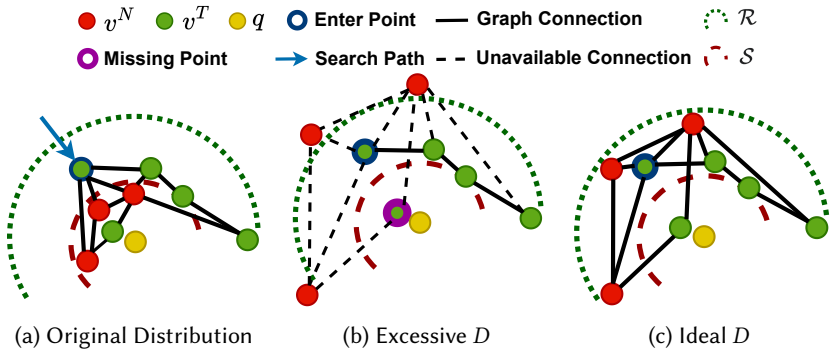
*5.3.1 Intuition.* Although repelling NTD a sufficient distance would easily make TD closer to $\boldsymbol{q}$, the HNSW traversal mechanism still needs neighbors to maintain graph connectivity, ensuring the sufficient traversal in nearest neighbor search. Thus, the exclusion distance must be narrowly bounded within a delicate range.

As shown in Figure 3a, the search path directs to a new vector and $\mathcal{R}$ is full. This vector connects a $\boldsymbol{v}^T$ and three $\boldsymbol{v}^N$. If the current vector distance distribution remains unchanged or if $D$ is insufficient, the three NTD points will enter $\mathcal{R}$. Due to their extreme proximity to $\boldsymbol{q}$, they persistently occupy the positions in $\mathcal{R}$, preventing other promising TD points from entering. When three NTD points violate the condition shown in line 15 of Algorithm 3 due to excessive $D$, they cannot enter the $\mathcal{R}$. As shown in Figure 3b, their connections are unavailable. Consequently, graph traversal operations initiated from any direction cannot reach the TD point (purple circle) nearest to the $\boldsymbol{q}$ under this compromised graph connectivity. The ideal $D$ appears in Figure 3c, where all NTD points remain confined within the region of $\mathcal{R}$ but lie well beyond the radius of $\mathcal{S}$. This vector distance distribution

**Algorithm 3:** OptiGreedySearch($\boldsymbol{q}, \mathcal{F}, ef, ep, \mathcal{G}$)

**Input:** query vector $\boldsymbol{q}$, filtering conditions $\mathcal{F}$, entry point $ep$, graph index $\mathcal{G}$
**Output:** result set of $ef$ approximate nearest neighbors

$\mathcal{V} \leftarrow ep$ ; // visited list
$C \leftarrow ep$ ; // candidate set
$\mathcal{R} \leftarrow ep$ ; // result set
$n \leftarrow 0$ ; // count for TD in $\mathcal{R}$
**while** $C \neq \emptyset$ **do**
  $\boldsymbol{v}_\alpha \leftarrow$ extract $\arg\min_{\boldsymbol{v} \in C} \overline{Dis}(\boldsymbol{q}, \boldsymbol{v})$;
  $\boldsymbol{v}_\beta \leftarrow \arg\max_{\boldsymbol{v} \in \mathcal{R}} \overline{Dis}(\boldsymbol{q}, \boldsymbol{v})$;
  **if** $\overline{Dis}(\boldsymbol{q}, \boldsymbol{v}_\alpha) > \gamma \cdot \overline{Dis}(\boldsymbol{q}, \boldsymbol{v}_\beta)$ *and* $\overline{p} > 0.5$ **then**
    **break**;
  **foreach** $\boldsymbol{v} \in$ *neighbors of* $\boldsymbol{v}_\alpha$ **do**
    **if** $\boldsymbol{v} \notin \mathcal{V}$ **then**
      $\mathcal{V} \leftarrow \mathcal{V} \bigcup \boldsymbol{v}$;
      $\boldsymbol{v}_\beta \leftarrow \arg\max_{\boldsymbol{v} \in \mathcal{R}} \overline{Dis}(\boldsymbol{q}, \boldsymbol{v})$;
      $A \leftarrow$ attribute of $\boldsymbol{v}$;
      **if** $(\overline{Dis}(\boldsymbol{q}, \boldsymbol{v}) < \overline{Dis}(\boldsymbol{q}, \boldsymbol{v}_\beta)) \parallel (|\mathcal{R}| < ef)$ **then**
        $C \leftarrow C \bigcup \boldsymbol{v}$;
        $\mathcal{R} \leftarrow \mathcal{R} \bigcup \boldsymbol{v}$;
        **if** $A \in \mathcal{F}$ **then**
          $n \leftarrow n + 1$;
        **if** $|\mathcal{R}| > ef$ **then**
          $A \leftarrow$ attribute of $\arg\max_{\boldsymbol{v} \in \mathcal{R}} \overline{Dis}(\boldsymbol{q}, \boldsymbol{v})$;
          $\mathcal{R} \leftarrow \mathcal{R} \setminus \arg\max_{\boldsymbol{v} \in \mathcal{R}} \overline{Dis}(\boldsymbol{q}, \boldsymbol{v})$;
          **if** $A \in \mathcal{F}$ **then**
            $n \leftarrow n - 1$;
**return** $\mathcal{R}$;




(a) Original Distribution (b) Excessive $D$ (c) Ideal $D$

Fig. 3. The vector distance distributions under various $D$.

supports both the index connectivity and the accuracy of top-$k$ query results. To this end, an ideal

$D$ for NTD should be in the following range:

$$\max_{\boldsymbol{v}^T \in \mathcal{S}} Dis(\boldsymbol{q}, \boldsymbol{v}^T) < Dis(\boldsymbol{q}, \boldsymbol{v}^N) + D < \max_{\boldsymbol{v} \in \mathcal{R}} Dis(\boldsymbol{q}, \boldsymbol{v}), \tag{3}$$

where $\max_{\boldsymbol{v}^T \in \mathcal{S}} Dis(\boldsymbol{q}, \boldsymbol{v}^T)$ is the farthest $\boldsymbol{v}^T$ from $\boldsymbol{q}$ in $\mathcal{S}$, and $\max_{\boldsymbol{v} \in \mathcal{R}} Dis(\boldsymbol{q}, \boldsymbol{v})$ is the farthest $\boldsymbol{v}$ from $\boldsymbol{q}$ in $\mathcal{R}$. The $D$ in this range can balance ANNS efficiency with index connectivity.

*5.3.2 Theoretical Derivation.* Based on the above analysis, we derive the optimal value of $D$ by identifying the key parameters that significantly affect retrieval performance, including the cardinality of the candidate set $ef$, the cardinality of the result set $k$, the selectivity $p$, and the average distance between data points $\Delta d$. To systematically examine the effect of each parameter on $D$ and theoretically derive its optimal value, we adopt a control variate methodology and model the influence of unmeasured factors through a residual function $x$. The theoretical derivation in this section assumes that the attribute distribution among data points has no correlation with their positions in the vector space.

The derivation of $D$ begins with $\Delta d$, a parameter obtained from the dataset's vector distance distribution during indexing. This formulation is theoretically grounded in a classical spatial statistical model that assumes data points are independently and uniformly distributed in a $d$-dimensional Euclidean space, loosely corresponding to a homogeneous Poisson point process [33]. Under this idealized model, the expected distance $D_m$ from an arbitrary point to its $m$-th nearest neighbor satisfies the asymptotic relationship $d_m \propto m^{1/d}$. To achieve a feasible approximation, we perform a Taylor expansion of the function $f(m) = m^{1/d}$ around a reference point $m_0$, where $m_0$ is deemed as $ef$ (around $ef = 100$). The expansion is given by

$$f(m) = m_0^{1/d} + \frac{1}{d} m_0^{\frac{1}{d}-1}(m - m_0) + \frac{1}{2d} m_0^{\frac{1}{d}-2}(m - m_0)^2 + \dots. \tag{4}$$

Given that $d > 1$ and $m_0$ is sufficiently large, the higher-order Taylor coefficients (proportional to $m_0^{1/d-n}$ for $n \geq 2$) are tiny compared to the zeroth- and first-order terms, which can be safely ignored in the local approximation. Therefore, the relationship between $d_m$ and $m$ is well approximated by a linear function within a neighborhood of $m_0$.

Although real-world spatial distributions often deviate from an ideal Poisson process, the general rule still holds: nearest-neighbor distances increase monotonically as a function of $m$. We observe that the linear fitting strategy can model the spatial distribution with acceptable empirical error. This practical operation enables and grounds the robust estimation of local density $\Delta d$ through a linear model derived from the distance distribution. Theoretically, since different queries induce distinct $\Delta d$, they could benefit from customized $D$ values for optimal retrieval performance. However, in practice, determining an optimal $D$ for a query requires counting $\Delta d$, which incurs prohibitively high costs in real-time query serving. To address this issue, we determine $D$ with a strategy using the dataset's global $\Delta d$, which is calculated during the offline index construction phase. That avoids computing the local density $\Delta d$ online. The global $\Delta d$ is defined as

$$\Delta d = \frac{d_\alpha - d_\beta}{\alpha - \beta}, \tag{5}$$

where $d_\alpha$ and $d_\beta$ represent the average distances to the $\alpha$-th and the $\beta$-th nearest vectors from a vector, respectively. Different from the calculation on averaging the distance to all the neighbors, our definition is practical for ANNS, as shown in Section 6.5.2.

Due to the fact that a larger $\Delta d$ necessitates a correspondingly larger $D$ to ensure that an NTD point is pushed to a position equivalent to that of a TD point, $D \propto \Delta d$. We define $D$ as

$$D = x\Delta d, \tag{6}$$

where $x$ is a function related to $ef$, $k$, and $p$. Based on Equation (5), when the $\boldsymbol{q}$ is deemed as the anchor vector, the distance between $\boldsymbol{q}$ and $m$-th nearest vector to $\boldsymbol{q}$ can be expressed by

$$d_m \approx d_0 + (m-1)\Delta d,$$

where $d_0$ is $Dis(\boldsymbol{q}, \boldsymbol{v}_0)$ and $\boldsymbol{v}_0$ is the nearest vector to the $\boldsymbol{q}$. Therefore, the coverage of a nearest vector set based on the $\boldsymbol{q}$ can be expressed by $d_m$, where $\boldsymbol{v}_m$ is the $m$-th nearest vector to the $\boldsymbol{q}$. Given $\mathcal{S}$ and $\mathcal{R}$ contain $k$ and $ef$ vectors, respectively, the radii of $\mathcal{S}$ and $\mathcal{R}$ for the $\boldsymbol{q}$ shown in Figure 3 can be expressed by

$$\begin{cases} R(\boldsymbol{q}, \mathcal{S}) = d_0 + (k-1)\Delta d, & (7) \\ R(\boldsymbol{q}, \mathcal{R}) = d_0 + (ef-1)\Delta d, & (8) \end{cases}$$

where $R(\cdot)$ denotes the radius.

To meet the left part of Inequality (3) based on Equation (7), all NTD points in $\mathcal{S}$ must be excluded from $\mathcal{S}$ after the $D$ is applied. In the extreme case where the $\boldsymbol{v}_0$ is an NTD point, the inf $D = R(\boldsymbol{q}, \mathcal{S})$. Meanwhile, to satisfy $|\mathcal{S}| = k$, $R(\boldsymbol{q}, \mathcal{S})$ must expand to cover the top $\frac{k}{p}$ nearest TD before the vector distance distribution changes. This process aims to find the support vector from the $\boldsymbol{v}_k$ to the $\boldsymbol{v}_{\frac{k}{p}}$ based on an expansion factor of $\frac{1}{p}$. Similar to the right part of Inequality (3) based on Equation (8), the $R(\boldsymbol{q}, \mathcal{R})$ also expands after the $D$ is applied. Consequently, the support vector for $\mathcal{R}$ alters to the $\boldsymbol{v}_{\frac{ef}{p}}$. As a result, the radii of $\mathcal{R}$ and $\mathcal{S}$ under the new vector distance distribution become

$$\begin{cases} R(\boldsymbol{q}, \mathcal{S}) = d_0 + (\frac{k}{p} - 1)\Delta d, & (9) \\ R(\boldsymbol{q}, \mathcal{R}) = d_0 + (\frac{ef}{p} - 1)\Delta d. & (10) \end{cases}$$

In addition, the $D$ is applied only to NTD points that constitute a proportion of $(1-p)$ of $\mathcal{D}$. Therefore, the additional average distance assigned to each $\boldsymbol{v}^N$ and $\boldsymbol{v}^T$ are $\frac{D}{1-p}$ and $\frac{pD}{1-p}$, respectively. It causes the $R$ in Equation (10) and Equation (9) to be systematically inflated. To ensure accuracy, the calculation of $R(\boldsymbol{q}, \mathcal{S})$ and $R(\boldsymbol{q}, \mathcal{R})$ incorporates an offset of $\frac{pD}{1-p}$ applied to the global average distance.

Based on the above analysis, Inequality (3) can be derived to the form as

$$\begin{cases} d_0 + (\frac{k}{p} - 1)\Delta d + D < d_0 + (\frac{ef}{p} - 1)\Delta d - \frac{Dp}{1-p}, & (11) \\ d_0 + D > d_0 + (\frac{k}{p} - 1)\Delta d - \frac{Dp}{1-p}, & (12) \end{cases}$$

where Inequality (11) ensures that the NTD point in $\mathcal{S}$ with the farthest distance to $\boldsymbol{q}$ remains in $\mathcal{R}$ even when its distance is augmented by $D$, while Inequality (12) ensures that the NTD point in $\mathcal{S}$ with the nearest distance to $\boldsymbol{q}$ is excluded from $\mathcal{S}$ after augmentation. Through the derivation of two inequalities, we establish that

$$(1-p)(\frac{k}{p} - 1)\Delta d < D < (1-p)(\frac{ef}{p} - \frac{k}{p})\Delta d. \quad (13)$$

According to the minimax principle in optimization theory, selecting the average of the two bounds provided by the inequalities yields the most robust estimator. Therefore, we strategically recommend the intermediate value for $D$. It is described as

$$D = \frac{1}{2p}(1-p)(ef-p)\Delta d. \quad (14)$$

Empirically, normalizing this value by $ef$ is found to further enhance robustness across different search parameters. Equation (14) demonstrates that $D$ is sensitive to $p$ and their relationship is consistent with intuitive reasoning. For example, a negative correlation exists between $p$ and $D$. When $p \to 0$, $D \to \infty$, while as $p \to 1$, $D \to 0$.

### 5.4 Termination Condition Optimization

The conventional termination condition of HNSW is satisfied when $\min_{\boldsymbol{v}\in C} Dis(\boldsymbol{q},\boldsymbol{v}) > \max_{\boldsymbol{v}\in \mathcal{R}} Dis(\boldsymbol{q},\boldsymbol{v})$. Since the vector distance distribution has changed by $D$ in our approach, we adopt a similar condition to terminate the query search, *i.e.*,

$$\min_{\boldsymbol{v}\in C} \overline{Dis}(\boldsymbol{q},\boldsymbol{v}) > \max_{\boldsymbol{v}\in \mathcal{R}} \overline{Dis}(\boldsymbol{q},\boldsymbol{v}).$$

However, such a condition could introduce the issue of premature termination.

*5.4.1 Issue of Premature Termination.* The conventional termination condition effectively balances search efficiency and recall under the implicit assumption that all candidate vectors are TD points. In the filtered ANNS setting, this assumption no longer holds. Even when the termination condition is satisfied, further exploration may still reveal TD points that are closer to $\boldsymbol{q}$. If a sequence of NTD points is consecutively added to both $C$ and $\mathcal{R}$ during traversal, the decision to terminate will be dominated by the distances among these NTD points. Since the $\Delta d$ between NTD points remains unchanged, the termination condition may trigger prematurely once no nearer NTD points are found while some TD points remain undiscovered. This premature termination leads to an incomplete search and, consequently, degraded recall.

*5.4.2 Optimization.* To mitigate premature termination caused by consecutive NTD traversals, we adopt a conservative search strategy. This strategy aims not only to obtain $k$ TD points in the final $\mathcal{S}$, but also to ensure that sufficient TD points are included in the $\mathcal{R}$ to make termination decisions reliable. Let $\overline{p}$ denote the proportion of TD points within $\mathcal{R}$. To guarantee that at least $k$ TD points can be retrieved, the candidate distribution must satisfy $ef \cdot \overline{p} - k > 0$, which can be rewritten as

$$\overline{p} > \frac{k}{ef}. \tag{15}$$

Combining Equation (11) and Equation (12), this constraint is satisfied only when $ef > 2k$, which ensures $\overline{p} < 0.5$. However, increasing $\overline{p}$ generally leads to higher computational overhead due to longer search paths. To balance efficiency and recall, we empirically set $\overline{p} = 0.5$, requiring that at least half of the points in $\mathcal{R}$ are TD points. Finally, we refine the original termination condition by incorporating this additional constraint to form a more robust termination criterion. FAVOR treats this optimization as an optional component for high-precision search. When activated, it can effectively improve the recall rate while keeping the $ef$ parameter unchanged.

## 6 EVALUATION

We comprehensively evaluate the performance of FAVOR across multiple datasets and various filtering scenarios against state-of-the-art methods. Specifically, Section 6.1 describes the experimental setup. Section 6.2 evaluates search performance across different scenarios. Section 6.3 analyzes the indexing construction overhead, Section 6.4 presents an ablation study to evaluate the effectiveness of each component, and Section 6.5 validates the correctness of our theoretical assumptions. All experiments are conducted on a server equipped with two Intel(R) Xeon(R) Gold 6326 CPUs @ 2.90GHz (32 cores and 64 threads in total) and 512 GB of memory.

Table 3. Dataset statistics.

| Dataset | # Base | # Query | Dimension | Source |
|---|---|---|---|---|
| **SIFT1M** | 1000000 | 10000 | 128 | Image |
| **GIST1M** | 1000000 | 1000 | 960 | Image |
| **DEEP1M** | 1000000 | 10000 | 96 | Image |
| **Msong** | 992272 | 200 | 420 | Audio |
| **Paper** | 2029997 | 10000 | 200 | Text |
| **Words** | 8000 | 200 | 3072 | Text |
| **LAION25M** | 25011200 | 1000 | 512 | Text & Image |

## 6.1 Experimental Setup

*6.1.1 Filtering Conditions.* We employ four types of filtering conditions as experimental scenarios. The specific settings are as follows:

- **Equality:** The attribute value must exactly match the queried value. We consider two equality conditions: (1) a boolean equality condition, *Equality_bool*, with a selectivity of 50%; and (2) an integer equality condition, *Equality_int*, with a selectivity of 10%.
- **Inclusion:** The value of an attribute belongs to the set specified by the filtering condition. We use a set of three values for an integer attribute, corresponding to a selectivity of 30%.
- **Range:** The value of an attribute lies in a given value range. We draw a range with 10% and 50% selectivity as *Range_10* and *Range_50*.
- **Logic:** It combines the above filtering conditions using logical operators (AND, OR, and NOT) to form a composite filtering condition. In our experiments, we evaluate a logical AND combination of *Equality_int* and *Range_50*, yielding an overall selectivity of approximately 5%.

*6.1.2 Datasets.* Table 3 presents all datasets used in our evaluation. For each dataset, we attach the vectors with attributes to support diverse testing scenarios. Each vector is assigned three types of attributes: *bool*, *int*, and *float*. For *bool* attributes, true and false are assigned with equal probability. For *int* attributes, values are uniformly sampled from the integers 0 to 9. For *float* attributes, values are uniformly drawn from the range [0, 100].

*6.1.3 Baselines.* We compare our approach against six state-of-the-art filtered ANNS methods and two vector databases. We use their open-sourced codes unless otherwise noted. We configure the vector databases to use HNSW. Below, we describe the configuration of each baseline.

- **ACORN-$\gamma$**: We follow the original paper [11], setting $M = 32$ and $M_\beta = 64$. $\gamma$ is set to 12 for all datasets but GIST1M. Due to its higher dimensionality, we set $\gamma = 50$ for GIST1M.
- **ACORN-1**: We set $M = M_\beta = 32$ and $\gamma = 1$ for all datasets, according to [11].
- **Result Set Filtering (RSF)**: We implement this baseline on `hnswlib`[2]. The HNSW index is constructed with $M = 32$ and $efc = 40$ for `SIFT1M`, `DEEP1M`, `Paper`, and `Words`, and with $M = 64$ and $efc = 200$ for `GIST1M`, `Msong` and `LAION25M`.
- **Milvus** [5]: As a widely used vector database supporting multiple indexing algorithms, we configure Milvus to use HNSW as the underlying index. We set the same $M$ and $efc$ as in the Result Set Filtering for the fair comparison.
- **Vearch** [34]: It is a distributed vector database developed by JD.com. We also configure it to use HNSW indexing with identical parameters to those used in Result-Set-Filtering.
- **UNG**: We implement UNG following the methodology in [8], which builds proximity graphs using the Vamana algorithm. The parameters are set as $\alpha = 1.2$, degree bound $R = 32$, and maximum queue length $L = 100$, cross-group edges $\delta = 6$ and the number of entry points is

[2]https://github.com/nmslib/hnswlib

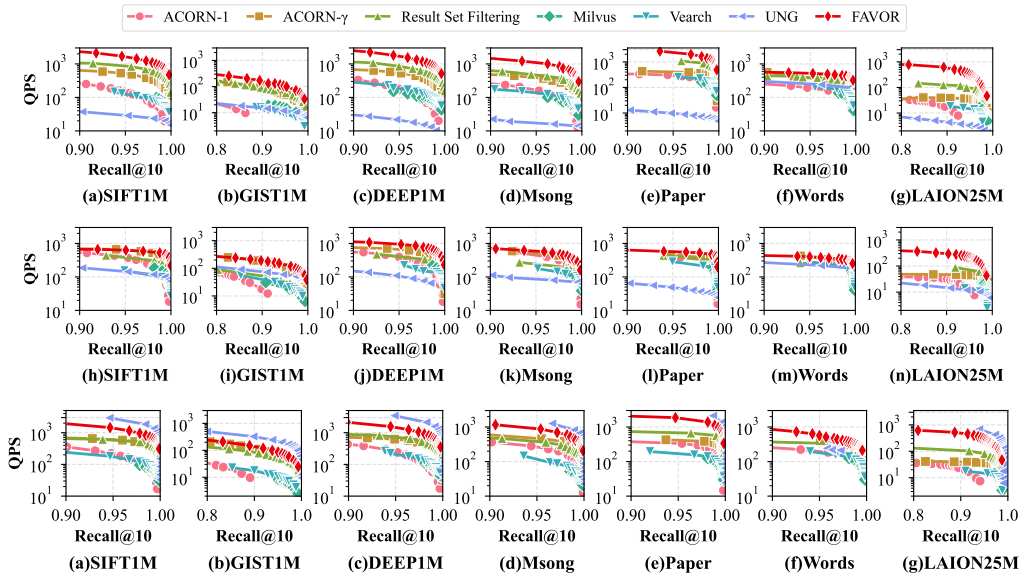



Fig. 5. Search performance in *Inclusion*.

$\sigma = 16$. Note that UNG requires attribute values to start from 1. Therefore, we shift the original bool attributes (0/1) to the range [1, 2], and int attributes (0–9) to [3, 12]. Since UNG does not support range queries on float attributes, we omit float attributes during indexing. We adopt the containment scenario for implementing *Equality* and the overlap scenario for *Inclusion*.

- **SeRF**: For range query evaluation, we follow the 2DSegmentGraph construction method in [10]. The graph is built on top of HNSW. For `SIFT1M`, `DEEP1M`, `Paper` and `Words`, we set $index_k = 32$, $ef\_con = 40$; for the remaining datasets, we use $index_k = 64$, $ef\_con = 200$. For all datasets, we set $efc\_max = 500$. SeRF only supports range filtering and does not support float attributes. Therefore, we convert float attributes into ordered integers and assign them as int attributes for SeRF.

We released our code on GitHub[3]. FAVOR is implemented in C++ and we use the same HNSW construction parameters as the above baselines for fairness. Additionally, we set $\alpha = 10$ and $\beta = efc$ for computing $\Delta d$. The search performance is evaluated under single-threaded execution with SIMD optimizations disabled since not all methods support multi-threading or SIMD.

## 6.2 Performance Evaluation

*6.2.1 QPS-recall tradeoff comparison.* Figure 4, 5, 8, and 9 present the QPS-recall curve at high recall for the four filtering conditions.

As shown in Figure 4, in the *Equality* scenario, FAVOR outperforms all other baselines across all datasets. Under the condition of $Recall@10 = 95\%$, the performance improvement of FAVOR over the best baseline ranges from 1.3× to 5×.

Figure 5 presents the results for the *Inclusion* scenario. FAVOR still outperforms all filtering-agnostic baselines, achieving a performance improvement of 1.5× to 3× when $Recall@10 = 95\%$. Note that the performance of FAVOR is also competitive when compared to the filter-specific method UNG.

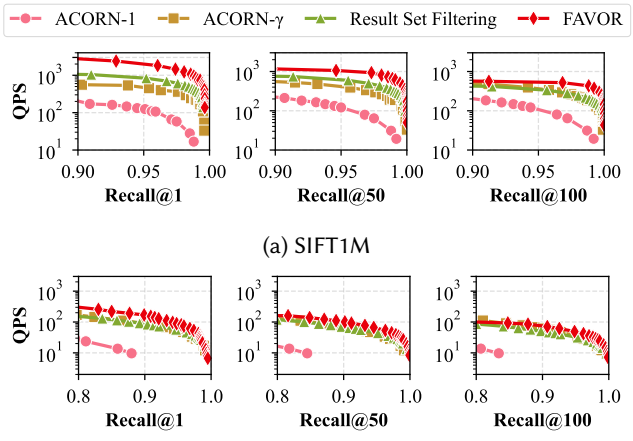
[3]https://github.com/JunjieSong123/FAVOR

In the *Range* scenario, as shown in Figure 8, FAVOR outperforms all filtering-agnostic methods. Remarkably, FAVOR achieves performance comparable to SeRF on the SIFT, Msong, Paper, and LAION25M datasets within the *Range_50* setting, despite SeRF's specialized optimization for range queries.

In the *Logic* scenario, as shown in Figure 9, FAVOR ranks among the top-performing methods, trailing only slightly behind ACORN-$\gamma$ on GIST1M. This minor gap arises because ACORN-$\gamma$ (with $\gamma = 50$) employs a densely connected index structure that incurs substantial additional indexing overhead (see Table 4), yielding higher performance in this specific case.

Overall, across nearly all experimental scenarios, FAVOR consistently outperforms all filtering-agnostic baselines, including ACORN-1, ACORN-$\gamma$, and result-set filtering. At *Recall*@10 = 95%, FAVOR improves QPS by up to 5× over the best baseline. Although UNG and SeRF demonstrate QPS advantages in certain datasets under the *Inclusion* and *Range* conditions, respectively, these benefits arise from their specialized optimizations. In contrast, FAVOR sustains robust performance across diverse query scenarios by effectively balancing filtering generality and search efficiency without requiring extensive index modification. Notably, FAVOR sustains consistently high performance across diverse scenarios while avoiding the structural overhead associated with index-enhanced approaches.

*6.2.2 Performance at different recall levels.* We assess the proposed method in the *equality_bool* scenario across multiple recall levels. Experiments on SIFT1M and GIST1M datasets evaluate *Recall*@1, *Recall*@50, and *Recall*@100. As shown in Figure 6, for *Recall* > 95%, FAVOR substantially outperforms the baseline, achieving speedups of up to 3.5× at *Recall*@1, 2.4× at *Recall*@50, and 2.0× at *Recall*@100. Performance gains are consistent across all recall levels and align closely with the trend observed at *Recall*@10.




(a) SIFT1M

(b) GIST1M

Fig. 6. Search performance at different recall levels.

*6.2.3 Performance under Varying Selectivity.* We assess FAVOR's performance across different selectivity values to validate our search strategy. Under the *Range* scenario with *Recall*@10 = 95%, we compare the QPS of FAVOR, including graph-based and brute-force search, against the baseline on SIFT1M and GIST1M datasets. Results in Figure 7 show how selectivity affects performance.

The two search strategies provided by FAVOR, *i.e.*, brute-force search and graph-based search, exhibit an interesting phenomenon across different experimental setups. Their performance curves

Table 4. Index construction time (s).

| Method \ Dataset | SIFT1M | GIST1M | DEEP1M | Msong | Paper | Words | LAION25M |
|---|---|---|---|---|---|---|---|
| **ACORN-1** | 7.2 | 31.8 | 5.5 | 17.2 | 22 | 0.8 | 697.9 |
| **ACORN-$\gamma$** | 150 | 3452 | 105 | 289 | 330 | 16.8 | 12791 |
| **Result Set Filtering** | 9.6 | 62.2 | 7.6 | 32.6 | 30.7 | 1.2 | 1147 |
| **UNG** | 11.4 | 82.1 | 10.4 | 26.6 | 31.4 | 1.5 | 1216 |
| **SeRF** | 33.9 | 327 | 33.2 | 107.7 | 103.5 | 3.8 | 3782 |
| **FAVOR** | 11.2 | 64.7 | 9.7 | 34.1 | 32.5 | 1.3 | 1148 |

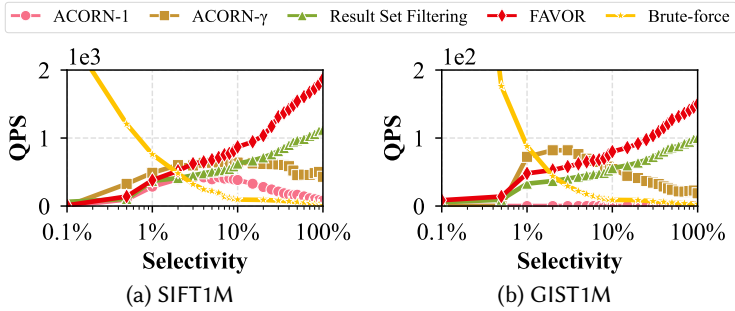
Table 5. Storage overhead for Index (GB).

| Method \ Dataset & size | SIFT1M | GIST1M | DEEP1M | Msong | Paper | Words | LAION25M |
|---|---|---|---|---|---|---|---|
| | 0.49 | 3.58 | 0.37 | 1.56 | 1.52 | 0.094 | 47.8 |
| **ACORN-1** | 0.81 | 3.89 | 0.69 | 1.86 | 2.15 | 0.098 | 55.5 |
| **ACORN-$\gamma$** | 0.97 | 4.20 | 0.85 | 3.02 | 2.78 | 0.102 | 59.6 |
| **Result Set Filtering** | 0.755 | 4.08 | 0.63 | 2.05 | 2.04 | 0.098 | 56.2 |
| **UNG** | 0.70 | 3.74 | 0.58 | 1.71 | 1.88 | 0.098 | 54.3 |
| **SeRF** | 0.78 | 4.33 | 0.69 | 2.13 | 2.36 | 0.10 | 57.4 |
| **FAVOR** | 0.755 | 4.08 | 0.63 | 2.05 | 2.04 | 0.098 | 56.2 |

consistently intersect within $1\% < p < 3\%$. When $p < 1\%$, brute-force achieves higher QPS, and the QPS metric improves as $p$ decreases. Conversely, for $p > 3\%$, the graph-based method delivers higher QPS, with performance improving as $p$ increases. Furthermore, within the intersection region, the QPS of the graph-based method varies by less than 8%, while the brute-force method shows fluctuations exceeding 50%. This implies that the graph-based search is less sensitive to estimation error of $p$ than the brute-force search in this region.

Based on this observation, we conservatively set the switching threshold at $p = 1\%$. At $p = 1\%$, the graph-based method enters its stable performance region, allowing the system to transition to a more robust search strategy upon reaching moderate selectivity levels. Even within the intersecting performance range, the QPS degradation remains below 8% due to the flat response curve of the graph-based method. Therefore, the selected threshold is not only empirically supported but also strikes a balance between efficiency, stability, and robustness.

With the same settings, we also compare the performance of FAVOR against baseline approaches. Although the QPS decreases as $p$ lowers, FAVOR mitigates the risk of sustained performance degradation by proactively employing brute-force search rather than graph-based search under low-selectivity conditions. Therefore, FAVOR consistently outperforms the state-of-the-art baseline methods across most selectivity levels. It achieves an average QPS improvement of 2.5× on the `SIFT1M` dataset and 2× on the higher-dimensional `GIST1M` dataset. These results demonstrate that FAVOR remains effective and robust over a wide range of selectivity levels.




(a) SIFT1M (b) GIST1M

Fig. 7. QPS yielded by different methods across varying selectivity at *Recall*@10 = 95%.

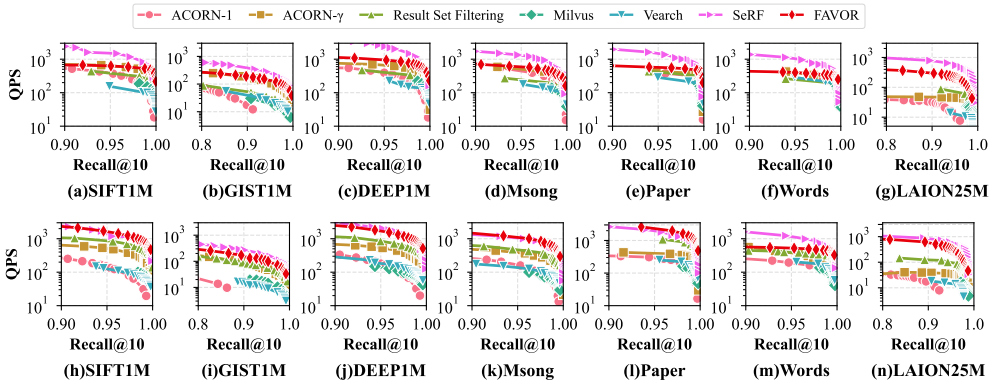



Fig. 8. Search performance in *Range_10* ( (a) – (g) ) and *Range_50* ( (h) – (n) ).

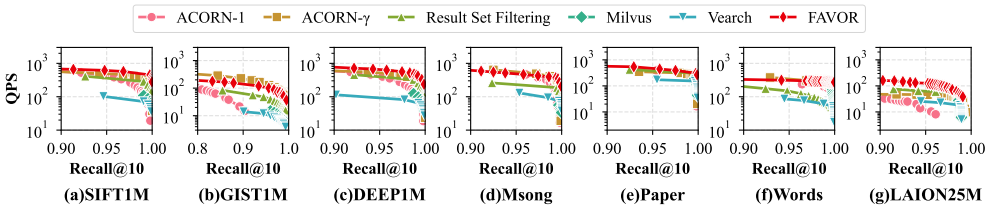



Fig. 9. Search performance in *Logic*.

### 6.3 Index Construction

*6.3.1 Construction Time.* Table 4 shows a comparison of index construction times for different methods under a 64-thread environment, where FAVOR is built upon HNSW. During the construction process, we only utilize neighbor nodes within the *efc* range as approximate $\alpha$-th and $\beta$-th nearest neighbors to estimate distances. This introduces only a minor additional cost when computing $\Delta d$, resulting in an average construction time about 2*s* higher than that of Result Set Filtering, whose construction time is equivalent to vanilla HNSW. This overhead is significantly lower than that of ACORN-$\gamma$ and SeRF, and is comparable to UNG. Considering FAVOR 's superior adaptability and runtime performance, this modest overhead highlights its overall competitiveness.

*6.3.2 Storage Overhead.* Table 5 compares the index sizes of different methods with respect to the original dataset size. Noted that the reported index sizes include the original vectors. FAVOR, built upon HNSW, only introduces small additional storage overhead for attribute information and the data distribution parameter $\Delta d$, without incorporating any extra links or expanded neighbor lists. Consequently, its index size remains comparable to that of Result Set Filtering. Although the graph-tree hybrid index of UNG mitigates the overhead inherent to pure graph constructs, it, as well as SeRF, suffers from inflexibility in handling arbitrary predicates, necessitating compensatory storage. ACORN-$\gamma$, in contrast, depends on dense edge connections to maintain graph connectivity, resulting in substantial storage overhead. FAVOR maintains connectivity without such redundancy, ensuring a more storage-efficient design.

### 6.4 Ablation Study

We conduct an ablation study on the key components of our FAVOR search algorithm, including the exclusion distance mechanism and termination condition, to demonstrate their effectiveness.

*6.4.1 Exclusion Distance.* We conduct a comparative analysis of three exclusion distances strategies using *Equality_int* on three datasets. $D_0$ denotes setting the zero exclusion distance. The second strategy is $D_{max} = \max_{\boldsymbol{v}^T \in \mathcal{D}} Dis(\boldsymbol{q}, \boldsymbol{v}^T) - \min_{\boldsymbol{v}^N \in \mathcal{D}} Dis(\boldsymbol{q}, \boldsymbol{v}^N)$, as making all TD closer to the

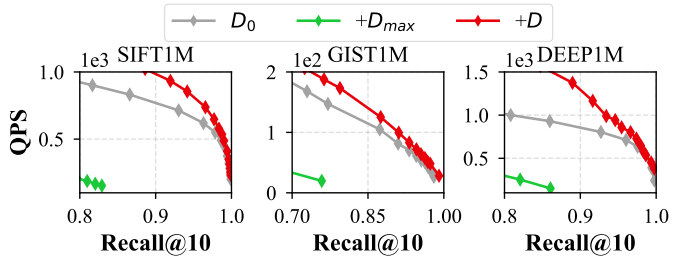



Fig. 10. Effect of exclusion distance strategies.

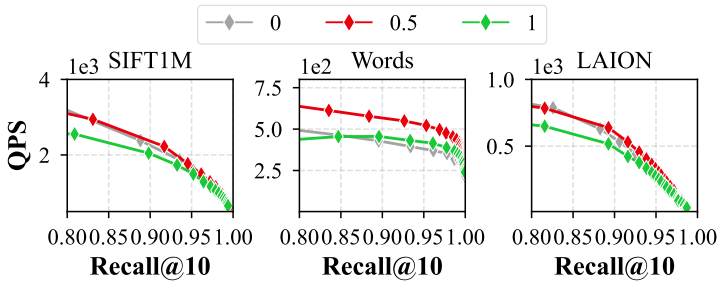



Fig. 11. Impact of optimized termination condition.

$\boldsymbol{q}$ than any NTD point. $D$ is calculated by Equation (14) and used in FAVOR. Figure 10 shows that adopting $D$ achieves the best performance. Compared to the method with $D_0$, it achieves an average 1.6× improvement in QPS when $Recall@10 = 90\%$, demonstrating the effectiveness of using exclusion distance. Furthermore, adopting $D$ significantly outperforms the solution with $D_{max}$, which is consistent with the intuition presented in Figure 5.3.1. Note that we observe that the advantage of employing $D$ is less pronounced `GIST1M`, because of its higher dimensionality and a relatively smaller $\Delta d$ among neighboring points. Consequently, $D$ has a less significant impact on the distance distribution, which caused the limited QPS improvement.

*6.4.2 Termination Condition.* As analyzed in Section 5.4, without optimizing the filtering condition, reliance solely on distance comparisons for search termination may lead to premature termination of the retrieval process. In such cases, recall under the same search parameter *ef* falls below the expected level, while QPS appears higher than anticipated. To validate the selection of our termination condition parameter, we examine the trade-off between QPS and recall in the *Equality_bool* scenario across three datasets of varying scales: `SIFT1M`, `Words`, and `LAION25M`, using different values of the optimized termination threshold $\overline{p}$, where $\overline{p} = 0$ corresponds to the case without termination condition optimization. As shown in Figure 11, the optimal balance occurs at $\overline{p} = 0.5$, where the system achieves the highest QPS while still maintaining high recall rates ($Recall@10 > 95\%$). Furthermore, by exploiting early termination, a smaller $\overline{p}$ can be further reduced to enable fast, lower-precision retrieval when speed is prioritized over accuracy.

## 6.5 Verification Test

*6.5.1 Search path.* To validate the key idea presented in Section 3.1.1, we conduct two sets of experiments on three datasets, including `SIFT1M`, `GIST1M`, and `DEEP1M`. First, we collect the proportion of TD points along the search paths generated by FAVOR and Result Set Filtering. Figure 12 illustrates the relationship between QPS and the TD point proportion across different datasets and selectivity levels under the *Equality* scenario at $Recall@10 = 95\%$. It is clear that there is a consistent positive

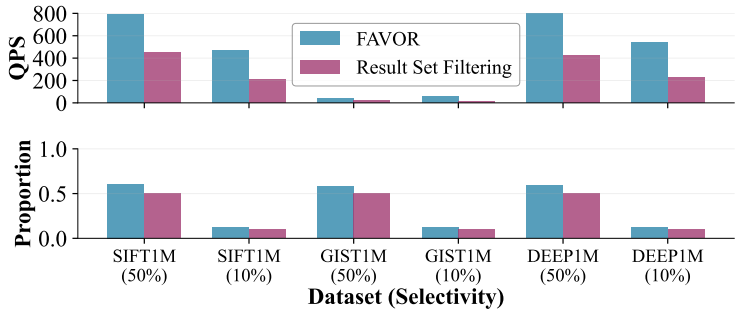



Fig. 12. Performance vs. TD proportion on search paths.

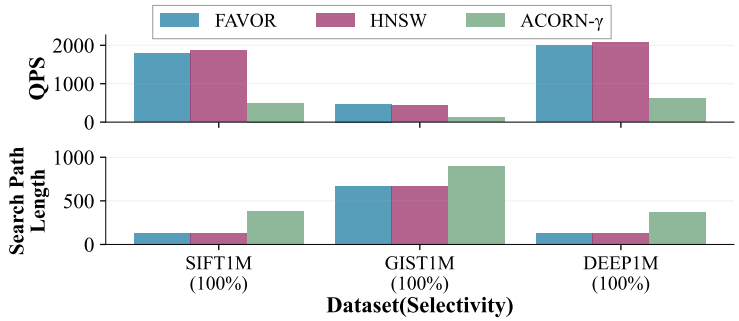



Fig. 13. Comparison of search path length and QPS.

correlation between the TD proportion and QPS for both methods. Given that QPS is typically negatively correlated with search path length, this phenomenon implies that FAVOR achieves shorter search paths under high-recall conditions, thereby indirectly validating the idea.

Second, under a non-filtering setting ($p = 100\%$) with $Recall@10 = 95\%$, we compare FAVOR against vanilla HNSW and ACORN-$\gamma$ in terms of QPS and search path length. As shown in Figure 13, FAVOR achieves QPS and path lengths comparable to those of vanilla HNSW, and exhibits shorter paths and higher QPS compared to ACORN-$\gamma$. This phenomenon demonstrates that the exclusion distance mechanism maintains the graph connectivity, avoiding inefficient traversals.

*6.5.2 The Linear Model.* We conduct experiments to validate the consistency between the proposed linear model and Equation (5). Specifically, we examine whether the distance from a point in the space to its $m$-th nearest neighbor approximately follows a linear relationship with $m$. For each vector in the dataset $D$, we treat it as an anchor point and compute its distances to the $m$-th nearest neighbors, where $m = 1, 2, \ldots, 1000$. We then perform linear regression with $m$ as the independent variable and the $d_m$ as the dependent variable. The goodness of fit is assessed by the coefficient of determination, *i.e.*, $R^2$[35]. It is used to quantify the proportion of the variation in the dependent variable that is predictable from the independent variable. Its range is from 0 to 1, with higher values indicating a stronger linear relationship between the variables. As summarized in Table 6, $R^2$ exceeds 0.8 across all datasets, with a consistently small standard deviation. Given high $R^2$ and low variance observed, for an arbitrary point in the space, the distance to its $m$-th nearest neighbor increases approximately linearly with $m$. These results constitute strong empirical evidence supporting the validity of the proposed linear model.

## 7 Related Work

**ANNS.** ANNS techniques can be broadly classified into hash-based and graph-based methods. Hash-based approaches [36] perform coarse-grained partitioning of the search space, struggling

Table 6. Linear Model.

| Metric \ Dataset | SIFT1M | GIST1M | DEEP1M | Msong | Paper | Words | LAION25M |
|---|---|---|---|---|---|---|---|
| **Mean $R^2$** | 0.848 | 0.835 | 0.856 | 0.864 | 0.867 | 0.817 | 0.821 |
| **$R^2$ Std.** | 0.035 | 0.046 | 0.044 | 0.037 | 0.048 | 0.061 | 0.083 |

to achieve high recall efficiently for high-dimensional datasets. Graph-based methods [37–41] explicitly encode accurate neighbor relationships into the indexing structure and have recently emerged as the dominant paradigm. FAVOR builds on the success of the widely used HNSW graph and significantly improves its applicability in filtered ANNS. Quantization [12, 42, 43] can be added to both hash-based and graph-based methods to reduce storage and speed up vector comparison by approximation with lossy representations, which can be added on top of FAVOR to boost efficiency.

**Filtered ANNS.** Pre-filtering and post-filtering policies are common options in vector databases [5, 34, 44] due to their simplicity and versatility; however, the former is only efficient for low selectivity, and the latter often requires iterative search to return enough target points. Recent studies integrate filtering support into vector indexes to improve efficiency [30, 45]. SeRF [10], UNG [8], RangePQ [46], DSG [47] are tailored methods for specific filtering conditions to achieve efficient search by additional index structures, which limit their flexibility and incur construction and maintenance overheads. FAVOR only uses an exclusion distance during serving and does not modify the graph index structure, adding minimal overhead and offering competitive performance. SIEVE [9] supports arbitrary filters but groups vectors and maintains collections of subindexes, which means it still requires prior knowledge about queries. FAVOR keeps a unified index and does not assume queries. ACORN [11] searches only along TD points but suffers from connectivity issues, resulting in difficulty in achieving high recall under low selectivity. FAVOR strikes a balance between prioritizing search on TD points and exploration with near NTDs, which improves the selectivity range suitable for graph-based search.

## 8 Conclusion

In this paper, we presented FAVOR, a filter-agnostic vector ANNS method that delivers high efficiency across varying selectivity levels. It employs a selectivity-driven search selector to dynamically choose the more suitable approach between FAVOR search algorithm and pre-filtering brute-force search to achieve higher QPS. FAVOR search introduces a novel selectivity-aware exclusion distance mechanism for efficient and seamless integration with HNSW. FAVOR achieves 1.3× to 5× higher QPS at $Recall@10 = 95\%$ than state-of-the-art filter-agnostic methods, while remaining competitive with tailored solutions in their target filtering scenarios. FAVOR bridges the gap between generality and performance, making it suitable for a wide range of real-world applications.

## Acknowledgments

This work was achieved in Key Laboratory of Information Storage System and Ministry of Education of China. This work was supported by the National Key Research and Development Program Grant No.2023YFB4502701, the National Natural Science Foundation of China Grant No.62232007. This work was supported by the China Scholarship Council (No. 202406160132).

## References

[1] Quoc V. Le and Tomás Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, Vol. 32. JMLR.org, 1188–1196. http://proceedings.mlr.press/v32/le14.html

[2] Martin Grohe. 2020. word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In *PODS*. ACM, 1–16. doi:10.1145/3375395.3387641

[3] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving Language Models by Retrieving from Trillions of Tokens. In *ICML*, Vol. 162. 2206–2240. https://proceedings.mlr.press/v162/borgeaud22a.html

[5] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD*. ACM, 2614–2627. doi:10.1145/3448016.3457550

[6] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *PVLDB* 14, 11 (2021), 1964–1978. doi:10.14778/3476249.3476255

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

[8] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *SIGMOD* 2, 6, Article 246 (2024), 27 pages. doi:10.1145/3698822

[9] Zhaoheng Li, Silu Huang, Wei Ding, Yongjoo Park, and Jianjun Chen. 2025. SIEVE: Effective Filtered Vector Search with Collection of Indexes. *PVLDB* 18, 11 (2025), 4723–4736.

[10] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search. *SIGMOD* 2, 1 (2024), 69:1–69:26. doi:10.1145/3639324

[11] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *SIGMOD* 2, 3 (2024), 120. doi:10.1145/3654923

[12] Guoyu Hu, Shaofeng Cai, Tien Tuan Anh Dinh, Zhongle Xie, Cong Yue, Gang Chen, and Beng Chin Ooi. 2025. HAKES: Scalable Vector Database for Embedding Search Service. *PVLDB* 18, 9 (2025), 3049–3062. doi:10.14778/3746405.3746427

[13] Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51, 1 (2008), 117–122. doi:10.1145/1327452.1327494

[14] Alexandr Andoni and Ilya P. Razenshteyn. 2015. Optimal Data-Dependent Hashing for Approximate Near Neighbors. In *STOC*. 793–801. doi:10.1145/2746539.2746553

[15] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. 2015. Practical and Optimal LSH for Angular Distance. In *NeurIPS*. 1225–1233. https://proceedings.neurips.cc/paper/2015/hash/2823f4797102ce1a1aec05359cc16dd9-Abstract.html

[16] Yury Elkin and Vitaliy Kurlin. 2022. Paired compressed cover trees guarantee a near linear parametrized complexity for all k-nearest neighbors search in an arbitrary metric space. *CoRR* abs/2201.06553 (2022). arXiv:2201.06553 https://arxiv.org/abs/2201.06553

[17] Michael E. Houle and Michael Nett. 2015. Rank-Based Similarity Search: Reducing the Dimensional Dependence. *TPAMI* 37, 1 (2015), 136–150. doi:10.1109/TPAMI.2014.2343223

[18] Kejing Lu, Hongya Wang, Wei Wang, and Mineichi Kudo. 2020. VHP: Approximate Nearest Neighbor Search via Virtual Hypersphere Partitioning. *PVLDB* 13, 9 (2020), 1443–1455. doi:10.14778/3397230.3397240

[19] Marius Muja and David G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *TPAMI* 36, 11 (2014), 2227–2240. doi:10.1109/TPAMI.2014.2321376

[20] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68. doi:10.1016/J.IS.2013.10.006

[21] 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 *CIKM*. ACM, 4580–4584. doi:10.1145/3511808.3557610

[22] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An Efficient and Robust Framework for Approximate Nearest Neighbor Search with Attribute Constraint. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2023/hash/32e41d6b0a51a63a9a90697da19d235d-Abstract-Conference.html

[23] Gaurav Gupta, Jonah Yi, Benjamin Coleman, Chen Luo, Vihan Lakshman, and Anshumali Shrivastava. 2023. CAPS: A Practical Partition Index for Filtered Similarity Search. *CoRR* abs/2308.15014 (2023). doi:10.48550/ARXIV.2308.15014 arXiv:2308.15014

[24] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *WWW*. ACM, 3406–3416. doi:10.1145/3543507.3583552

[25] Joshua Engels, Benjamin Landrum, Shangdi Yu, Laxman Dhulipala, and Julian Shun. 2024. Approximate Nearest Neighbor Search with Window Filters. In *ICML*. OpenReview.net. https://openreview.net/forum?id=8t8zBaGFar

[26] Yuexuan Xu, Jianyang Gao, Yutong Gou, Cheng Long, and Christian S. Jensen. 2024. iRangeGraph: Improvising Range-dedicated Graphs for Range-filtering Nearest Neighbor Search. *SIGMOD* 2, 6 (2024), 239:1–239:26. doi:10.1145/3698814

[27] Anqi Liang, Pengcheng Zhang, Bin Yao, Zhongpu Chen, Yitong Song, and Guangxu Cheng. 2024. UNIFY: Unified Index for Range Filtered Approximate Nearest Neighbors Search. *PVLDB* 18, 4 (2024), 1118–1130. https://www.vldb.org/pvldb/vol18/p1118-yao.pdf

[28] Chaoji Zuo and Dong Deng. 2023. ARKGraph: All-Range Approximate K-Nearest-Neighbor Graph. *PVLDB* 16, 10 (2023), 2645–2658. doi:10.14778/3603581.3603601

[29] Weijie Zhao, Shulong Tan, and Ping Li. 2022. Constrained Approximate Similarity Search on Proximity Graph. *CoRR* abs/2210.14958 (2022). doi:10.48550/ARXIV.2210.14958 arXiv:2210.14958

[30] Gaurav Sehgal and Semih Salihoglu. 2025. NaviX: A Native Vector Index Design for Graph DBMSs With Robust Predicate-Agnostic Search Performance. *PVLDB* 18, 11 (2025), 4438–4450. https://www.vldb.org/pvldb/vol18/p4438-sehgal.pdf

[31] Jerzy W. Jaromczyk and Godfried T. Toussaint. 1992. Relative neighborhood graphs and their relatives. *Proc. IEEE* 80, 9 (1992), 1502–1517. doi:10.1109/5.163414

[32] William Feller. 1991. *An introduction to probability theory and its applications, Volume 2*. Vol. 2. John Wiley & Sons.

[33] JFC Kingman. 1993. *Poisson Processes*. Oxford University Press, United Kingdom.

[34] Jie Li, Haifeng Liu, Chuanghua Gui, Jianyu Chen, Zhenyun Ni, and Ning Wang. 2019. The Design and Implementation of a Real Time Visual Search System on JD E-commerce Platform. arXiv:1908.07389 [cs.IR]

[35] Norman R. Draper and Harry Smith. 1998. *Applied Regression Analysis*.

[36] 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

[37] Sairaj Voruganti and M. Tamer Özsu. 2025. MIRAGE-ANNS: Mixed Approach Graph-based Indexing for Approximate Nearest Neighbor Search. *SIGMOD* 3, 3, Article 188 (2025), 27 pages. doi:10.1145/3725325

[38] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *SIGMOD* 1, 1, Article 54 (2023), 27 pages. doi:10.1145/3588908

[39] Yun Peng, Byron Choi, Tsz Nam Chan, and Jianliang Xu. 2022. LAN: Learning-based Approximate k-Nearest Neighbor Search in Graph Databases. In *ICDE*. 2508–2521. doi:10.1109/ICDE53745.2022.00233

[40] Mengxu Jiang, Zhi Yang, Fangyuan Zhang, Guanhao Hou, Jieming Shi, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. DIGRA: A Dynamic Graph Indexing for Approximate Nearest Neighbor Search with Range Filter. *SIGMOD* 3, 3, Article 148 (2025), 26 pages. doi:10.1145/3725399

[41] Mohsen Dehghankar and Abolfazl Asudeh. 2025. HENN: A Hierarchical Epsilon Net Navigation Graph for Approximate Nearest Neighbor Search. arXiv:2505.17368 [cs.DS] https://arxiv.org/abs/2505.17368

[42] 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. *SIGMOD* 3, 3, Article 202 (2025), 26 pages. doi:10.1145/3725413

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

[44] 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. *PVLDB* 13, 12 (2020), 3152–3165. doi:10.14778/3415478.3415541

[45] Yannis Chronis, Helena Caminal, Yannis Papakonstantinou, Fatma Özcan, and Anastasia Ailamaki. 2025. Filtered Vector Search: State-of-the-Art and Research Opportunities. *PVLDB* 18, 12 (2025), 5488–5492. doi:10.14778/3750601.3750700

[46] Fangyuan Zhang, Mengxu Jiang, Guanhao Hou, Jieming Shi, Hua Fan, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. Efficient Dynamic Indexing for Range Filtered Approximate Nearest Neighbor Search. *SIGMOD* 3, 3, Article 152 (2025), 26 pages. doi:10.1145/3725401

[47] Zhencan Peng, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2025. Dynamic Range-Filtering Approximate Nearest Neighbor Search. *PVLDB* 18, 10 (2025), 3256–3268. doi:10.14778/3748191.3748193

Received October 2025; revised January 2026; accepted February 2026