Efficient and Robust Out-Of-Distribution Vector Similarity Search with Cross-Distribution Monotonic Graph

QIANG YUE*, Hangzhou Dianzi University, China
MENGZHAO WANG*, Zhejiang University, China
XIAOLIANG XU†, Hangzhou Dianzi University, China
CHENG LONG, Nanyang Technological University, Singapore
YUXIANG WANG, Hangzhou Dianzi University, China
JIAHUI WANG, Hangzhou Dianzi University, China

Vector similarity search is a key component in many AI applications. While graph-based methods represent the state-of-the-art for vector similarity search, existing graph indexes often suffer from poor navigability and clusterability in Out-Of-Distribution (OOD) scenarios (e.g., database and queries are sourced from different modalities). Recent studies have attempted to mitigate this issue by projecting and integrating query-derived auxiliary index structures, but these approaches remain largely heuristic and lack theoretical grounding. In this work, we propose Cross-Distribution Monotonic Graph (CDMG), a novel graph index designed to inherently support navigability and clusterability for OOD queries. CDMG introduces an integration strategy that bridges the distribution gap between database and query vectors, ensuring a key property—cross-distribution monotonicity—that guarantees monotonic search paths for OOD queries on graph indexes. Theoretical analysis demonstrates that CDMG achieves a lower search time complexity in OOD scenarios compared with existing graph indexes. Furthermore, we introduce CDMG+, a practical variant of CDMG that improves construction efficiency by introducing query synthesis, fusion-based distance computation, as well as optimized candidate acquisition and neighbor selection strategies. Extensive empirical evaluations demonstrate that our techniques outperform state-of-the-art methods, achieving up to a 3.6× speedup on real-world OOD datasets.

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

Additional Key Words and Phrases: Out-Of-Distribution Vector Similarity Search, Approximate Nearest Neighbor Search, High-Dimensional Vector, Graph Index

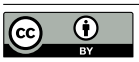
ACM Reference Format:
Qiang Yue, Mengzhao Wang, Xiaoliang Xu, Cheng Long, Yuxiang Wang, and Jiahui Wang. 2026. Efficient and Robust Out-Of-Distribution Vector Similarity Search with Cross-Distribution Monotonic Graph. *Proc. ACM Manag. Data* 4, 1, Article 29 (February 2026), 29 pages. https://doi.org/10.1145/3786643

1 INTRODUCTION

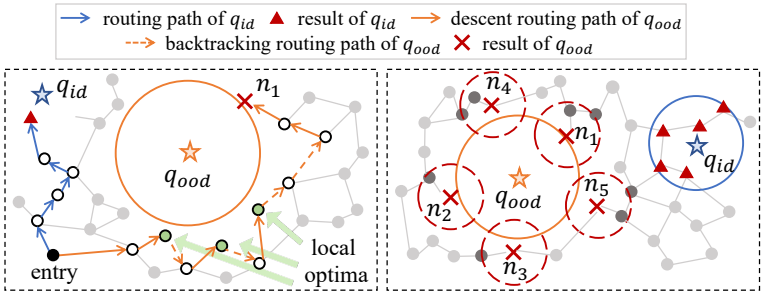
The growing prevalence of multimedia content, encompassing images, text, and audio, has intensified the need for effective multimedia data management solutions [26, 51, 78]. A widely adopted

*Both authors contributed equally to this research.
†Corresponding author.

Authors' Contact Information: Qiang Yue, Hangzhou Dianzi University, China, yq@hdu.edu.cn; Mengzhao Wang, Zhejiang University, China, wmzssy@zju.edu.cn; Xiaoliang Xu, Hangzhou Dianzi University, China, xxl@hdu.edu.cn; Cheng Long, Nanyang Technological University, Singapore, c.long@ntu.edu.sg; Yuxiang Wang, Hangzhou Dianzi University, China, lsswyx@hdu.edu.cn; Jiahui Wang, Hangzhou Dianzi University, China, jiahuiwang@hdu.edu.cn.



ACM 2836-6573/2026/2-ART29
https://doi.org/10.1145/3786643



(a) Navigability Issue (b) Clusterability Issue

Fig. 1. Navigability and clusterability issues of graph indexes under OOD queries.

approach involves embedding such multimedia content as *vectors* in high-dimensional space using neural networks [33, 35, 64], followed by employing *vector similarity search* [18, 19, 74, 89] to retrieve relevant content. Specifically, given a collection of database vectors and a query vector, vector similarity search first constructs an index over the database vectors, and then efficiently retrieves the k nearest neighbors of the query from the database vectors using the prebuilt index. Recent studies and industrial systems have demonstrated that *graph indexes* achieve state-of-the-art performance in balancing search efficiency and accuracy [2, 11, 37, 60, 69, 75]. In these graph indexes, each vertex represents a database vector, and each edge establishes a similarity relationship between corresponding vectors. The search process over a graph index employs a greedy routing strategy, guiding vertex traversal to rapidly converge to the nearest neighbors of the query. Owing to their *navigability* for search processes [15, 34, 45] and *clusterability* for search results [16, 31, 70], graph indexes have been extensively integrated into mainstream vector databases [26, 28, 49, 71, 94].

When designing graph indexes, it is typically assumed that both query and database data originate from single-modal sources [13, 14, 36, 42, 54, 67]. In such cases, query vectors follow the same distribution as database vectors in the high-dimensional space (i.e., the Mahalanobis distances between queries and database vectors are small, see §2.1 for details), referred to as In-Distribution (ID) queries. This assumption is a critical prerequisite for the effectiveness of current graph-based methods in addressing the vector similarity search problem [23, 45, 50]. With the rise of multi-modal learning models, particularly multi-modal large language models (LLMs), multi-modal retrieval has become a fundamental component of many emerging AI applications. One prominent example is retrieval-augmented generation (RAG) [1, 39, 43, 84], where a text-based query retrieves relevant information from an image database to generate a rich response. In this context, query and database data are derived from different modalities. Due to the inherent *modality gap* [38, 57, 91], query vectors often exhibit a different distribution from database vectors (i.e., the Mahalanobis distances are significantly larger than those of ID queries [11]). Such queries are termed *Out-Of-Distribution (OOD) queries* [31]. While current graph indexes perform well for ID queries, processing OOD queries encounters significant performance bottlenecks [11, 65]. For instance, on the Text-to-Image dataset [3], OOD queries using HNSW [45]—a state-of-the-art graph index—experience a 7× increase in search latency compared to ID queries to achieve the same search accuracy. Efficiently and effectively handling OOD queries on graph indexes remains a widely recognized challenge in the field [11, 31, 59, 69]. We identify that the performance bottleneck of OOD queries stems from the disruption of two key properties—navigability and clusterability—that are essential for ensuring high search efficiency and accuracy in graph indexes.

- **Navigability Issue.** Current graph indexes rely on the assumption that queries are located near database vectors, enabling the greedy routing to converge efficiently toward the nearest neighbors [22, 50]. However, OOD queries *deviate significantly* from database vectors, violating

this assumption. For instance, on the WebVid dataset [5], OOD queries are positioned up to 9× farther from database vectors compared to ID queries. As illustrated in Fig. 1(a), the vertices and edges between them constitute the graph index, where the solid black point denotes the entry vertex of the greedy routing. While the routing path for the ID query q_{id} is short and direct, the greedy routing for the OOD query q_{ood} oscillates unpredictably. This is because the greedy routing for q_{ood} often descends into and gets trapped in local optima, forcing it to perform extensive backtracking to locate the true result.

- **Clusterability Issue.** Clusterability in graph indexes assumes that query results are closely grouped, allowing greedy routing to accurately locate all nearest neighbors once one is identified [70]. However, the nearest neighbors of OOD queries tend to be *widely scattered* across the graph index. For example, on WebVid, distances between neighbors of OOD queries are more than 2× larger than those of ID queries. As illustrated in Fig. 1(b), the five results (n_1 to n_5) for the OOD query q_{ood} exhibit poor clusterability, with the neighbors of each result not overlapping with the retrieved results. Consequently, a single search process is forced to perform a disjointed traversal across multiple, distant regions of the graph index to retrieve the complete result set, drastically increasing overall search latency. In contrast, the nearest neighbors of the ID query q_{id} demonstrate strong clusterability.

To address the challenges posed by OOD queries on graph indexes, current solutions enhance existing graph indexes by incorporating the distribution information of OOD queries during index construction. Two state-of-the-art methods are particularly noteworthy: RobustVamana [31] and RoarGraph [11]. RobustVamana improves navigability and clusterability by adding edges among the nearest neighbors associated with OOD queries. Through this optimization, RobustVamana reduces query latency by 15% compared to its original counterpart while maintaining the same search accuracy on the Text-to-Image dataset. Similar to RobustVamana, RoarGraph refines the edge selection for the nearest neighbors linked to OOD queries, thereby further enhancing navigability and clusterability. Consequently, RoarGraph achieves a 33% reduction in query latency compared to RobustVamana under the same search accuracy on the Text-to-Image dataset.

Despite recent advancements, the search performance of OOD queries still lags significantly behind that of ID queries on graph indexes. For instance, even with the leading RoarGraph, the query latency for OOD queries remains 5.2× higher than for ID queries on the Text-to-Image dataset. We argue that these optimizations fail to fundamentally resolve the navigability and clusterability problems of OOD queries. Specifically, they refine the neighbor relationships for only a small subset of database vectors (i.e., vertices) using a sampled query set (typically 10% of the database [11]), leaving the *majority of vertices unoptimized* (referred to as unoptimized vertices). Consequently, OOD queries are forced to traverse numerous unnecessary vertices on the graph index, guided by these unoptimized vertices. Furthermore, if a nearest neighbor of an OOD query lies within these unoptimized vertices, the query cannot efficiently locate other nearest neighbors through it. Our evaluation reveals that even with RoarGraph, 58% of vertices remain unoptimized on the Text-to-Image dataset[1]. Therefore, existing optimizations only partially mitigate the navigability and clusterability issues in graph indexes. For example, 82% of OOD queries are still trapped in local optima, and only 26% of the results are well-clustered within the index. Additionally, current optimization methods are *heuristic in nature and lack theoretical guarantees for search efficiency and accuracy*, limiting their applicability in real-world systems.

Our Solution and Contributions. This paper presents a novel graph index that is specifically tailored for OOD queries. We first demonstrate that by appropriately bridging the database and

[1]The percentage of vertices retaining identical edges was calculated between two RoarGraph versions: one built with and the other without queries. Both follow the best parameters based on official guidelines [11].

query distributions, OOD greedy routing can be guided to follow the reliable and efficient ID routing behavior. We then introduce a key property called *cross-distribution monotonicity*, which ensures that the routing path from an entry vertex to the nearest neighbor of an OOD query remains monotonic on the graph index. Analogous to the established monotonicity property for ID queries [23, 50], cross-distribution monotonicity serves as a theoretical basis for achieving high search performance in OOD settings. Building upon this theoretical basis, we design the Cross-Distribution Monotonic Graph (CDMG), which integrates OOD query information into database vectors and connects neighbors under both distributions for each vertex via a new edge selection algorithm. Our theoretical analysis proves that CDMG achieves a lower OOD search time complexity compared to existing methods. To further enhance construction efficiency and OOD search effectiveness, we present CDMG+, a practical variant of CDMG that optimizes database-query pairing, distance computation, as well as candidate acquisition and neighbor selection strategies. The key contributions of this paper are highlighted as follows:

- We identify the fundamental theoretical limitation of existing graph-based methods when handling OOD queries. (**§2.2**)
- We formalize cross-distribution monotonicity, a property analogous to the established monotonicity for ID queries, which provides the theoretical foundation for ensuring the monotonic routing of OOD queries on graph indexes. (**§3.1**)
- We propose CDMG, an index characterized by cross-distribution monotonicity, and prove that greedy routing on CDMG achieves a lower complexity compared to existing methods. (**§3.2**)
- We propose CDMG+, which enhances construction efficiency and OOD search effectiveness by optimizing database-query pairing, distance computation, as well as candidate acquisition and neighbor selection strategies. (**§3.3**)
- Extensive evaluations conducted on real-world OOD datasets clearly demonstrate the superiority of our proposed method in terms of efficiency and effectiveness, index cost, navigability and clusterability, as well as scalability. (**§4**)

For the rest of the paper, **§5** reviews related work, **§6** discusses potential future research directions, and **§7** concludes the paper.

2 PRELIMINARIES AND PROBLEM

2.1 Preliminaries

Definition 1 (**Vector Similarity Search** [18, 19, 74, 89]). *Given a database $\mathcal{X}$ and a query vector q, let $\delta(\cdot,\cdot)$ denote a distance metric between two vectors. Vector similarity search aims to retrieve the k result vectors $\mathcal{R}(q) = \{x_{n_1}, x_{n_2}, \ldots, x_{n_k}\}$ for query vector q, where $n_i \in \{1, 2, \ldots, |\mathcal{X}|\}$. Each result vector $x_{n_i} \in \mathcal{R}(q)$ satisfies $\delta(x_{n_i}, q) \leq \delta(x_j, q)$ for all $x_j \in \mathcal{X} \setminus \mathcal{R}(q)$.*

Achieving exact vector similarity search in a large database is impractical due to computational and storage constraints [25, 32]. Therefore, most existing vector similarity search methods focus on finding approximate results to balance efficiency and accuracy while minimizing overhead [29, 68, 90, 92]. We adopt the standard evaluation metric recall rate *Recall@k* to measure the quality of the k results for q. *Recall@k* is formally defined as follows:

$$Recall@k = \frac{|\mathcal{R}(q) \cap \mathcal{G}(q)|}{k} \quad , \tag{1}$$

where $\mathcal{G}(q)$ represents the true k nearest neighbors of the query vector q. Nowadays, numerous vector similarity search methods have been developed, with graph-based methods demonstrating particular promise. We introduce the graph-based vector similarity search as follows:

Graph-Based Vector Similarity Search. Graph-based methods are widely adopted in modern vector similarity search [10]. This paradigm consists of two processes: construction and search. In the construction process, a graph index [40, 48, 73, 85] is built over the database $\mathcal{X}$. Specifically, given a database $\mathcal{X}$ and an edge selection criterion, the graph index is defined as $G = (V, E)$. The vertices V correspond to the vectors in $\mathcal{X}$, and an edge $(v, u) \in E$ is established between two close vertices v and u based on the edge selection criterion. In the search process, graph-based vector similarity search utilizes a greedy routing algorithm on the graph index G to identify the top-k results for a query q [2, 7, 41, 50, 61, 80]. Starting from a random or preselected entry vertex e, the greedy routing iteratively converges to vertices closer to the query q. At each iteration, the closest unvisited vertex from the candidate set is selected as the next hop, and its out-neighbors are added to the candidate pool. The process terminates when all candidates have been visited, and the top-k candidates are returned as the final results. Current state-of-the-art graph-based methods are built upon the Monotonic Relative Neighborhood Graph (MRNG) [23], and they further enhance navigability and clusterability to achieve more efficient search performance.

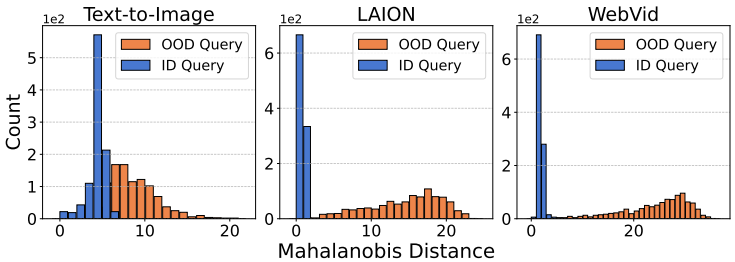


Fig. 2. Mahalanobis distances [11] of OOD and ID queries.

Out-Of-Distribution Query. Existing graph indexes are fundamentally designed under the assumption that the database $\mathcal{X}$ and the query vector q originate from the same underlying distribution [6, 11, 31]. However, with the rise of complex AI systems [28, 30, 76], recent developments have increasingly focused on generating vectors learned from multiple modalities to tackle cross-modal tasks. In such scenarios, queries often significantly deviate from the distribution of the database. Given a database $\mathcal{X}$, where database vectors are independently and identically distributed (i.i.d.) from a distribution P_{id}, a query q is considered out-of-distribution (OOD) if it is drawn i.i.d. from a distribution P_{ood} that differs from P_{id}, i.e., $P_{ood} \neq P_{id}$. Current vector similarity search methods lack rigorous quantification of OOD queries [31]. As a result, RoarGraph [11] adopts an empirical approach that uses the Mahalanobis distance $\delta_M(q, P_{id})$ to quantify the deviation of a query q from the database distribution P_{id}. Mahalanobis distance $\delta_M(q, P_{id})$ incorporates the covariance of P_{id}, effectively identifies OOD queries that exhibit distributional deviation from P_{id} despite appearing proximal under standard measures. As shown in Fig. 2, we present the Mahalanobis distances of OOD and in-distribution (ID) queries with respect to the database on three datasets: Text-to-Image [3], LAION [55], and WebVid [5]. The histograms demonstrate that the Mahalanobis distances for OOD queries exhibit significant deviations compared to those of ID queries.

2.2 Problem Analysis

Based on the preceding definitions, this paper investigates the problem of designing a graph-based method specifically optimized for OOD queries. We begin by explaining why ID-oriented graph-based methods are not suitable for OOD queries. Considering that most of the state-of-the-art ID-oriented methods are mainly derived from MRNG, we first present a detailed introduction to MRNG and its variants, followed by an analysis of their inherent limitations. Finally, we review recent OOD-oriented graph-based methods and highlight their limitations.

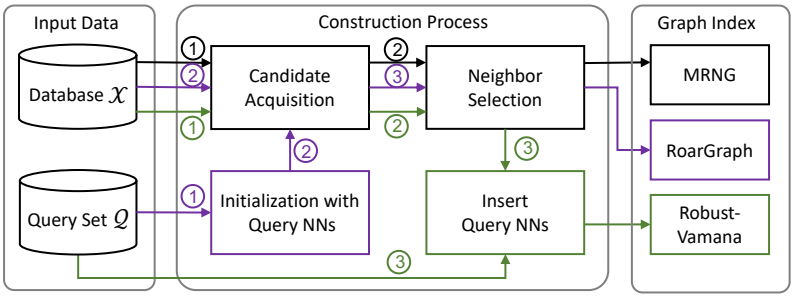


Fig. 3. Overview of construction pipeline for ID-oriented graph-based methods (MRNG) and OOD-oriented graph-based methods (RobustVamana and RoarGraph).

MRNG and Its Variants. As shown in Fig. 3, the construction of MRNG involves two phases [75]: *candidate acquisition* and *neighbor selection*, designed to enable clusterability and navigability.

Clusterability of MRNG. Clusterability refers to the likelihood that neighbors of a neighbor are also neighbors [16]. When the greedy routing identifies a result, the clusterability of graph indexes enables the rapid retrieval of all nearest neighbors [70, 79]. The clusterability of a graph index can be quantified as the ratio of mutual neighbors within the k nearest neighbors of a query q. Higher clusterability increases the probability that the nearest neighbors of q are also neighbors of each other. To achieve high clusterability, MRNG and its variants [23, 50] acquire candidates using an approximate K-Nearest Neighbor Graph (KNNG) (e.g., KGraph [16] and NSW [44]), which is constructed through an iterative or incremental process [75]. The iterative process refines neighbors by iteratively updating connections, while the incremental process refines neighbors through a greedy search. In both cases, each vertex is linked to vertices that are closer to it in the final KNNG.

Navigability of MRNG. The concept of navigability originates from early research on the small-world phenomenon [34, 46]. A graph index is considered navigable if greedy routing can consistently find a monotonic path between any two vertices [86]. While a complete graph is navigable, it is inefficient [15]. The goal of MRNG and its variants is to construct a sparser graph that maintains monotonic routing [23, 81]. However, they assume that the query is a vertex within the graph index [23, 95], which is often impractical in real-world scenarios. Recent advancements, including NSSG [22], Vamana [32], and τ-MNG [50], relax this limitation but still operate under the assumption that the query and database share an identical distribution [21, 22, 82, 89]. For instance, given an approximate KNNG, the state-of-the-art τ-MNG employs RobustPruning (described in Algorithm 1) to select the out-neighbors $N_{out}(v)$ of vertex v based on a query relaxation parameter τ in two steps: (1) v directly links to u if $\delta(x_v, x_u) \leq 3\tau$ (lines 7-9), where x_v and x_u are the vectors in $\mathcal{X}$ corresponding to v and u, respectively; (2) if $\delta(x_v, x_u) > 3\tau$, v links to u if and only if no existing neighbor $v^* \in N_{out}(v)$ occludes the edge from v to u (lines 10-12). As illustrated in Fig. 4(a), if $\delta(q_{id}, x_{n_1}) \leq \tau$, the greedy routing over τ-MNG moves at least τ closer to the query q_{id} in each iteration [50], ultimately guaranteeing the discovery of the nearest neighbors (vertices n_1 and n_2).

Limitations of MRNG. For an ID query, the query results tend to be mutual neighbors, and the query is close to its nearest neighbor. Thus, when constructing graph indexes for ID queries, MRNG can leverage KNNG to approximate the neighbor relationships among query results, and an appropriate query relaxation parameter τ can be readily determined to satisfy monotonicity. In contrast, for an OOD query: First, *the query results are widely dispersed*, and the misalignment between KNNG and the actual query results fundamentally violates the principle of MRNG. Second, *the query itself is often distant from its nearest neighbor*, further complicating the process. The underlying cause can be illustrated as follows.

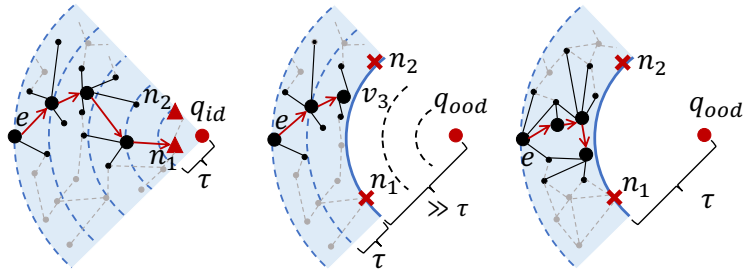


Fig. 4. (a) illustrates the greedy routing for an ID query q_{id} over MRNG. (b) and (c) compare the greedy routing for an OOD query q_{ood} under different τ settings.

If the query relaxation parameter τ remains small, the condition $\delta(q_{ood}, x_{n_1}) \leq \tau$ is rarely satisfied. This impedes the greedy routing from converging along a specific direction, trapping it in a local minimum [50]. Let $B(q, r)$ denote a ball of radius r centered at q, and $R(q, r_1, r_2)$ represent a ring within $B(q, r_2)$ but outside $B(q, r_1)$. As shown in Fig. 4(a), given an ID query q_{id} and an entry vertex $e \in R(q_{id}, 4\tau, 5\tau)$, the greedy routing on the graph index identifies the next hop in $R(q_{id}, 3\tau, 4\tau)$, progressing until it reaches the nearest neighbors (vertices n_1 and n_2). In contrast, as shown in Fig. 4(b), for an OOD query q_{ood}, all vertices in $R(q_{ood}, 3\tau, 4\tau)$ may be considered potential nearest neighbors due to the disparity between q_{ood} and the database vectors. As a result, the greedy routing may converge to a false result v_3, necessitating backtracking and traversal of numerous unnecessary vertices to locate the true nearest neighbors.

Table 1. Comparison of $\delta(q, x_{n_1})$ and Δ_i on OOD/ID datasets[2].

Dataset	$\delta(\mathbf{q}, \mathbf{x_{n_1}})$	Δ_1	Δ_{20}	Δ_{40}	Δ_{60}	Δ_{80}	Δ_{100}
WebVid	0.719	0.109	0.176	0.190	0.199	0.206	0.211
LAION	0.695	0.247	0.338	0.366	0.375	0.385	0.392
MNIST	0.261	0.253	0.419	0.471	0.505	0.532	0.554
DEEP	0.575	0.574	0.831	0.907	0.958	0.997	1.028

When the query relaxation parameter τ is set to $\delta(q_{ood}, x_{n_1})$, it exceeds the average distance between vertices and their respective nearest neighbors, meaning that the condition $\delta(x_v, x_u) \leq 3\tau$ is always met. As shown in Table 1, Δ_i denotes the average distance from each vector in $\mathcal{X}$ to its i-th nearest neighbor. Unlike ID datasets (MNIST and DEEP), where $\delta(q_{id}, x_{n_1})$ closely approximates Δ_1, OOD datasets (WebVid and LAION) exhibit $\delta(q_{ood}, x_{n_1})$ values that far exceed Δ_{100}. Note that 100 exceeds the typical out-degree setting for τ-MNG, which has an expected out-degree of $O(\ln |V|)$ (e.g., for a billion-scale database, the expected out-degree of τ-MNG is only 21). In this case, RobustPruning selects out-neighbors solely based on $\delta(x_v, x_u)$, where vertex u with smaller $\delta(x_v, x_u)$ is preferred. Thus, when τ is determined by $\delta(q_{ood}, x_{n_1})$, the τ-MNG degenerates into a KNNG, which lacks navigability and has proven inefficient in prior work [23, 75]. Fig. 4(c) illustrates that an excessively large τ still leads to prolonged search paths and local optima for OOD queries.

Limitations of OOD-Oriented Methods. As illustrated in Fig. 3, recent advancements extend the MRNG by incorporating query distribution information to optimize graph indexes for OOD queries. For instance, RobustVamana extends the ID-oriented MRNG variant Vamana by leveraging query-derived nearest neighbors (Query NNs) to fill available slots in the out-neighbor lists of

[2] $\delta(q, x_{n_1})$ denotes the distance between a query q (either an ID query q_{id} or an OOD query q_{ood}) and its nearest neighbor n_1. Cosine similarity measures $\delta(q, x_{n_1})$ and Δ_i for WebVid and LAION, Euclidean distance is used for MNIST and DEEP.

Algorithm 1 RobustPruning(v, Cand, $\mathcal{X}$, τ, o) [50]

Input: vertex v, candidate set Cand, database $\mathcal{X}$ (v and $u, v^* \in$ Cand correspond to vectors $x_v, x_u, x_{v^*} \in \mathcal{X}$, respectively), relaxation parameter τ, out-degree bound o
Output: out-neighbors $N_{out}(v)$
Heap $\leftarrow$ (Cand $\cup\, N_{out}(v)$) $\setminus \{v\}$; $N_{out}(v) \leftarrow \emptyset$;
while Heap $\neq \emptyset$ and out-degree $|N_{out}(v)| < o$ **do**
    $u \leftarrow$ closest vertex to v in Heap;
    Heap $\leftarrow$ Heap$\setminus\{u\}$;
    if $\delta(x_v, x_u) \leq 3\tau$ **then** ▷ Directly link if within relaxation radius
        $N_{out}(v) \leftarrow N_{out}(v) \cup \{u\}$; **continue**;
    end if
    if $\forall v^* \in N_{out}(v)$ s.t. $\delta(x_u, x_{v^*}) + 3\tau > \delta(x_v, x_u)$ **then** ▷ Ensure no occlusion
        $N_{out}(v) \leftarrow N_{out}(v) \cup \{u\}$;
    end if
end while
return $N_{out}(v)$

vertices. Similarly, RoarGraph directly initializes part of the graph index using Query NNs. However, these methods *continue to rely on ID-oriented candidate acquisition and neighbor selection strategies*: (1) In candidate acquisition, most candidates are selected based on distances between database vectors, which fail to capture the underlying cluster structure of the OOD query results. (2) In neighbor selection, the standard edge occlusion rule becomes invalid when $\delta(q, x_{n_1})$ significantly exceeds the average distance between database vectors and their nearest neighbors (i.e., Δ_1 in Table 1), leading to inefficient navigation. We quantitatively validate clusterability and navigability issues in OOD-oriented methods as follows.

To evaluate the clusterability produced by candidate acquisition, we introduce the k Query Nearest Neighbor Quality (*QNNQ@k*). Given a query q, let $\mathcal{G}(q) = \{x_{g_1}, x_{g_2}, \ldots, x_{g_{k+1}}\}$ denote its top-$(k+1)$ true nearest neighbors in the database, and let $N_{knng}(x_{g_1}) = \{x_{g_1}, x_{n_1}, \ldots, x_{n_k}\}$ denote $k+1$ neighbors of vertex g_1 in approximate KNNG built during candidate acquisition. We define:

$$QNNQ@k = \frac{|(\mathcal{G}(q) \setminus \{x_{g_1}\}) \cap (N_{knng}(x_{g_1}) \setminus \{x_{g_1}\})|}{k} \quad . \tag{2}$$

A higher *QNNQ@k* indicates stronger clusterability, as it reflects greater overlap between the query's true nearest neighbors and the neighborhood of its closest point g_1. If q is sampled from the database, $\mathcal{G}(q)$ will substantially overlap with $N_{knng}(x_{g_1})$, yielding $QNNQ@k \approx 100\%$. In contrast, an OOD query exhibits much smaller overlap. For example, we evaluate *QNNQ@100* for the same OOD query set on two index types: an ID-oriented index built without query information and an OOD-oriented index built with it. The OOD-oriented index (21%) shows only a marginal improvement over the ID-oriented one (20%) on the LAION dataset, showing that prevailing candidate acquisition falls short of enhancing clusterability. Refer to §4.4 for a detailed comparison of clusterability.

Regarding navigability induced by neighbor selection for the same OOD queries on two index types, 86.2% of queries are trapped in local optima under the ID-oriented index[3], while this ratio only slightly decreases to 82.1% under the OOD-oriented index, resulting in only a 4.1% improvement. Moreover, the average number of hops at the same recall level is reduced by only 11%. These results highlight the limited navigability improvement achieved by current neighbor selection.

[3]We fix the heap size to 1 during search (see Algorithm 3 for details) and measure the proportion of queries that fail to retrieve their true nearest neighbor under this setting, which we use as an indicator of being trapped in local optima.

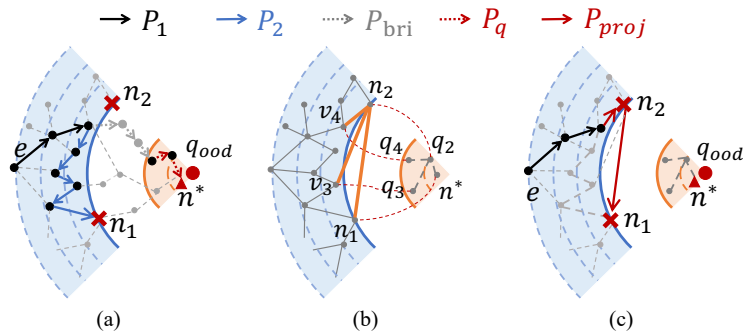


Fig. 5. (a) illustrates the difference in routing behavior between ID and OOD queries. (b) and (c) depict the construction and search process for cross-distribution monotonic path.

In summary, the limited performance of existing solutions stems from their reliance on the ID paradigm. This motivates us to (1) *generalize MRNG theory for OOD queries*, and (2) *optimize candidate acquisition and neighbor selection within the OOD context to enhance navigability and clusterability on graph indexes.*

3 THE PROPOSED ALGORITHM

3.1 Cross-Distribution Monotonicity

In this section, we investigate how OOD queries can achieve efficient monotonic search on graph indexes, analogous to that of ID queries. We first analyze why the monotonic paths of MRNG break down for OOD queries. We then introduce an integration strategy that bridges the distribution gap, ensuring cross-distribution monotonic search. Finally, we formally define the related concepts.

Failure Analysis of MRNG Monotonicity. In ID scenarios, the alignment between the distributions of the database and query vectors ensures a monotonic path on MRNG: even when initiated from a distant entry vertex, greedy routing can efficiently locate the query's nearest neighbor by following a monotonic path [23, 50]. In contrast, the routing paths induced by OOD queries on MRNG lack monotonicity due to the distribution gap. Specifically, given an OOD query q_{ood}, let r_{n_1} denote the distance between q_{ood} and its nearest neighbor n_1 in the database. The surface of the ball $B(q_{ood}, r_{n_1})$ defines the distribution boundary. The remaining nearest neighbors of q_{ood} are typically located near the boundary. The search process for q_{ood} on MRNG unfolds in two distinct phases: In the first phase, while the visited vertices remain distant from the distribution boundary, routing follows a monotonic path and progressively converges toward the OOD query. In the second phase, as the routing approaches the boundary, convergence toward the OOD query is disrupted due to the distribution gap, leading to a failure in maintaining a monotonic path.

Example 1. *As shown in Fig. 5(a), consider a database $\mathcal{X}$ (points in the blue region) and an OOD query set $\mathcal{Q}$ (points in the orange region). Let $\mathcal{X}^*$ denote points within convex hull that encloses both $\mathcal{X}$ and $\mathcal{Q}$. The bridging set is defined as $\mathcal{B} = \mathcal{X}^* \setminus (\mathcal{X} \cup \mathcal{Q})$, which characterizes the region separating $\mathcal{X}$ and $\mathcal{Q}$. Conceptually, an OOD query q_{ood} with respect to $\mathcal{X}$ can be interpreted as an ID query within the extended space $\mathcal{X}^*$. When performing search on the MRNG constructed over $\mathcal{X}^*$, q_{ood} follows a monotonic path $P^* = [P_1, P_{bri}, P_q]$ and finds nearest neighbor n^* as result, where P_1, P_{bri}, and P_q consist of points from $\mathcal{X}$, $\mathcal{B}$, and $\mathcal{Q}$, respectively. However, the bridging set $\mathcal{B}$ is not available in practical settings, leading to erratic oscillations in the second phase of the OOD query search, denoted as P_2.*

Motivation. To bridge the distribution gap, a natural approach is to establish connections between database and query vectors during graph construction, thereby enabling monotonic search over

the OOD query set $\mathcal{Q}$ in second-phase routing. However, this method is inefficient as it requires the graph to explicitly index $\mathcal{Q}$, resulting in elongated routing paths and the inclusion of redundant query vectors in retrieval results. To overcome this limitation, an intuitive strategy is to incorporate the topological structure induced by the MRNG built on $\mathcal{Q}$ into the MRNG constructed on $\mathcal{X}$. Consequently, the two routing phases on the resulting integrated graph index collectively enable *cross-distribution monotonic search*: In the first phase, greedy routing proceeds along a monotonic path guided by the database distribution toward the boundary. In the second phase, routing continues along a monotonic path shaped by the query distribution until reaching the target result.

Example 2. *As shown in Fig. 5(b), each query in $\mathcal{Q}$ is linked to its nearest vertex (red dashed lines). Additional projected edges (orange lines) can be introduced between vertices based on the neighbor relationships of their corresponding queries. For example, since q_2 is connected to n^* and q_4, n_2 is linked to n_1 and v_4. During the search process (as shown in Fig. 5(c)), P_1 and P_{proj} combine to form a new monotonic routing path $P = [P_1, P_{proj}]$ entirely within $\mathcal{X}$. The projected path P_{proj} traces the path P_q, returning n_1 and n_2 as results because their corresponding queries q_2 and n^* are closer to q_{ood}. The OOD routing path (i.e., $[P_1, P_2]$) in Fig. 5(a) requires seven hops to retrieve a single correct result n_1, whereas the monotonic routing path P (i.e., $[P_1, P_{proj}]$) retrieves two results n_1 and n_2 in just four hops.*

Definition and Notation. Based on the preceding analysis, our key insight is to construct a graph index over database vectors that ensures *cross-distribution monotonicity* for OOD queries. We formally define the cross-distribution monotonic path as follows:

Definition 2 (**Cross-Distribution Monotonic Path**). *For a database $\mathcal{X} \sim P_{id}$, an OOD query set $\mathcal{Q} \sim P_{ood}$ where $|\mathcal{Q}| \geq |\mathcal{X}|$, and an OOD query $q_{ood} \sim P_{ood}$, assume that each database vector $x_i \in \mathcal{X}$ is one-to-one paired with a unique OOD query $q_i \in \mathcal{Q}$, where x_i is the nearest neighbor of q_i in $\mathcal{X}$. Let $G(V, E)$ be a graph index over database $\mathcal{X}$, where each vertex $v_i \in V$ is associated with a database vector $x_i \in \mathcal{X}$ and its corresponding OOD query $q_i \in \mathcal{Q}$. Let the query relaxation parameter $\tau \geq 0$ be a constant. A routing path $P = [v_1, v_2, \ldots, v_b, v_{b+1}, \ldots, v_{\ell-1}, n_1]$ on G is a cross-distribution monotonic path for q_{ood} if it satisfies: $\forall i = 1, 2, \ldots, b-1$, $\delta(q_{ood}, x_i) - \tau > \delta(q_{ood}, x_{i+1})$, $\forall j = b, b+1, \ldots, \ell-2$, $\delta(q_{ood}, q_j) - \tau > \delta(q_{ood}, q_{j+1})$, and $\delta(q_{ood}, q_{\ell-1}) > \delta(q_{ood}, n^*)$, where n^* is the nearest query to q_{ood} in $\mathcal{Q}$ and n_1 is the nearest neighbor of n^* in $\mathcal{X}$.*

The *one-to-one pairing* and *distribution requirement of OOD query set* $\mathcal{Q}$ serve as two assumptions that establish the theoretical foundation: First, the one-to-one pairing allows us to formally reason about monotonic paths. Second, the distribution requirement ensures that the cross-distribution monotonic path converges to the nearest neighbor of q_{ood}. Building on this foundation, the cross-distribution monotonic path extends the τ-monotonic path [50] to OOD scenarios. When the database $\mathcal{X} \sim P_{id}$ and the query set $\mathcal{Q} \sim P_{id}$ follow the same distribution P_{id}, a monotonic path under the query distribution is equivalent to a monotonic path under the database distribution. In this case, given a ID query q_{id}, the cross-distribution monotonic path condition reduces to: $\forall i = 1, 2, \ldots, \ell-2$, $\delta(q_{id}, x_i) - \tau > \delta(q_{id}, x_{i+1})$ and $\delta(q_{id}, x_{\ell-1}) > \delta(q_{id}, x_{n_1})$, which corresponds to the τ-monotonic path in ID scenario. Building upon this extended concept, we introduce cross-distribution monotonicity as follows:

Definition 3 (**Cross-Distribution Monotonicity**). *Given a database $\mathcal{X} \sim P_{id}$ and an OOD query set $\mathcal{Q} \sim P_{ood}$ where $|\mathcal{Q}| \geq |\mathcal{X}|$, let the query relaxation parameter $\tau > 0$ be a constant. A graph index $G(V, E)$ over $\mathcal{X}$ has cross-distribution monotonicity if for any query $q_{ood} \sim P_{ood}$ satisfying $\delta(q_{ood}, n^*) \leq \tau$, there exists a cross-distribution monotonic path in G from any entry vertex to the result n_1. Here, n^* is the nearest query to q_{ood} in $\mathcal{Q}$, and n_1 is the nearest vertex to n^* in $\mathcal{X}$.*

3.2 CDMG Construction

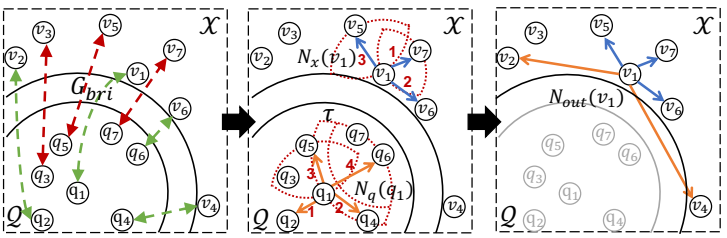
We introduce the Cross-Distribution Monotonic Graph (CDMG), which achieves cross-distribution monotonicity by applying a new edge selection algorithm guided by historical OOD query information. The use of historical queries is a well-established practice, as they are readily available and their distribution is typically stable in real applications [93]. Consequently, this has become a widely adopted strategy in prior work [11, 31, 65] and its use is standard practice in OOD benchmark datasets [59], where such queries are sourced from the training set. CDMG is defined as follows:

Definition 4 (**Cross-Distribution Monotonic Graph**). *Given a database $\mathcal{X} \sim P_{id}$ and an OOD query set $\mathcal{Q} \sim P_{ood}$ where $|\mathcal{Q}| \geq |\mathcal{X}|$, let the query relaxation parameter $\tau \geq 0$ be a constant. Assume that each database vector $x_i \in \mathcal{X}$ is one-to-one paired with a unique query $q_i \in \mathcal{Q}$. A CDMG is a directed graph $G(V, E)$ such that each vertex $v_i \in V$ is associated with a database vector $x_i \in \mathcal{X}$ and its corresponding OOD query $q_i \in \mathcal{Q}$, and the following conditions hold: (1) If $\delta(q_i, q_j) \leq 3\tau$, $(v_i, v_j) \in E$. (2) If $\delta(q_i, q_j) > 3\tau$ and $(v_i, v_j) \notin E$, there exists a vertex v^* such that $(v_i, v^*) \in E$, where the corresponding vectors x^* and q^* satisfy: $x^* \in (B(x_i, \delta(x_i, x_j)) \cap B(x_j, \delta(x_i, x_j) - 3\tau))$ and $q^* \in (B(q_i, \delta(q_i, q_j)) \cap B(q_j, \delta(q_i, q_j) - 3\tau))$.*

Lemma 1 establishes the monotonicity guarantee of CDMG:

Lemma 1. *CDMG guarantees cross-distribution monotonicity.*

Proof. If the edge $(v_1, n_1) \in E$, the path $P = [v_1, n_1]$ is a cross-distribution monotonic path. We first consider the case where $(v_1, n_1) \notin E$. Case 1: If $\delta(q_1, n^*) \leq 6\tau$, then G must contain a cross-distribution monotonic path $P = [v_1, v_2, n_1]$. This follows because v_1 must have an out-neighbor v_2 such that $q_2 \in (B(q_1, \delta(q_1, n^*)) \cap B(n^*, \delta(q_1, n^*) - 3\tau))$. Since $\delta(q_2, n^*) \leq 3\tau$, it follows that $(v_2, n_1) \in E$. When $\delta(q_{ood}, n^*) \leq \tau$, we have $\delta(q_{ood}, q_2) \leq \delta(q_{v_1}, n^*) - 2\tau$ and $\delta(q_{ood}, q_1) > \delta(q_1, n^*) - \tau$, which leads to $\delta(q_{ood}, q_2) < \delta(q_{ood}, q_1) - \tau$. Therefore, the path $P = [v_1, v_2, n_1]$ is a cross-distribution monotonic path. Case 2: If $\delta(q_1, n^*) > 6\tau$, then G must contain a cross-distribution monotonic path $P = [v_1, v_2, \ldots, v_b, v_{b+1}, \ldots, v_{\ell-1}, n_1]$. In the first phase, since G is a CDMG, it must satisfy $\delta(x_2, n^*) < \delta(x_1, n^*) - 3\tau$ and $\delta(x_2, q_{ood}) < \delta(x_1, q_{ood}) - \tau$, and at each step, the path moves closer to q_{ood} by at least τ until reaching a vertex v_b. In the second phase, similar to the case above, we have $\delta(q_{b+1}, n^*) < \delta(q_b, n^*) - 3\tau$ and $\delta(q_{ood}, q_{b+1}) < \delta(q_{ood}, q_b) - \tau$. Eventually, the search reaches a vertex $v_{\ell-2}$ such that $\delta(q_{\ell-2}, n^*) < 6\tau$, whose monotonicity has already been established. □



(a) Initialization (b) Candidate Acquisition & Neighbor Selection (c) Merging

Fig. 6. Illustration of the CDMG construction process.

Construction Process. As illustrated in Fig. 6, CDMG is constructed through three components, which are detailed as follows.

(1) Initialization. Following the requirement in Definition 2, given a database $\mathcal{X} \sim P_{id}$ and an OOD query set $\mathcal{Q} \sim P_{ood}$, the initialization aims to establish a one-to-one pairing between each vertex v_i

Algorithm 2 BuildCDMG($\mathcal{X}, \mathcal{Q}, \tau, o$)

1: **Input:** database $\mathcal{X}$, query set $\mathcal{Q}$ ($|\mathcal{Q}| \geq |\mathcal{X}|$), relaxation parameter $\tau > 0$, out-degree bound o
2: **Output:** CDMG G
3: $G \leftarrow (V \leftarrow \mathcal{X}, E \leftarrow \emptyset)$;
4: // Initialization
5: $C \leftarrow$ pair each query in $\mathcal{Q}$ with its nearest vertex in V;
6: $G_{bri} \leftarrow$ prioritize pairs in C with largest distance;
7: $G_{bri} \leftarrow G_{bri} \cup$ (pair unassigned vertices with nearest remaining queries);
8: **for** $\forall v_i \in V$ **do**
9: $q_i \leftarrow G_{bri}(v_i)$;
10: // Candidate acquisition & Neighbor selection
11: $N_x(v_i) \leftarrow$ RobustPruning(v_i, $V \setminus \{v_i\}$, $\mathcal{X}$, τ, o);
12: $N_q(q_i) \leftarrow$ RobustPruning(q_i, $\mathcal{Q} \setminus \{q_i\}$, $\mathcal{Q}$, τ, o);
13: // Merging
14: $N_{out}(v_i) \leftarrow N_x(v_i) \cup G_{bri}(N_q(q_i))$;
15: add edges (v_i, v_j) to E for all $v_j \in N_{out}(v_i)$;
16: **end for**
17: **return** G

in CDMG $G(V, E)$ constructed over $\mathcal{X}$ and a unique OOD query $q_i \in \mathcal{Q}$, where v_i is the nearest vertex in V to q_i. To achieve this, CDMG constructs a bipartite graph $G_{bri}(V, U, E_{bri})$ to bridge two different data distributions, where U corresponds to queries in $\mathcal{Q}$. The initialization is a prioritized process (lines 5-7 in Algorithm 2). First, a candidate matching set C is formed by identifying the nearest database neighbor for every query, from which CDMG iteratively selects and confirms the pair with the largest distance for matching. This prioritization ensures the most representative OOD queries—those furthest from any vertex and deep within the OOD region—are matched first, establishing reliable projected paths for the most challenging searches (green arrows in Fig. 6(a)). Subsequently, any vertices remaining unassigned after this process are matched to their nearest available query to satisfy the 1-regular property. *For feasible database-query pairing, the size of the OOD query set $|\mathcal{Q}|$ must be greater than or equal to the size of the database $|\mathcal{X}|$.*

(2) Candidate Acquisition & Neighbor Selection. After the initialization component, each vertex v_i is associated with a database vector $x_i \in \mathcal{X}$ and a unique OOD query $q_i \in \mathcal{Q}$. The core target of candidate acquisition and neighbor selection is to select out-neighbors for each vertex v_i from both the database $\mathcal{X}$ and query set $\mathcal{Q}$ by applying the same edge occlusion rule (detailed in Algorithm 1). Specifically, under the query distribution, each query q_i ranks all remaining query vectors $\mathcal{Q} \setminus \{q_i\}$ by distance to form a candidate list Heap. Each candidate $q_j \in$ Heap is then checked against the current out-neighbors: if any existing out-neighbor q^* occludes q_j, the edge (q_i, q_j) is discarded; otherwise, q_j is added to $N_q(q_i)$. This process iterates until all candidates in Heap are examined. The identical edge occlusion rule is applied to each vertex v_i to select its out-neighbors $N_x(v_i)$ from $V \setminus \{v_i\}$. *The naive implementation of candidate acquisition and neighbor selection performs exhaustive search in both query and database distributions, doubling the construction cost.*

(3) Merging. Since all elements in $N_q(q_i)$ are OOD queries, the target of the merging component is to replace each query with its corresponding vertex to eliminate the need for indexed queries. Specifically, all queries in $N_q(q_i)$ are replaced by their corresponding vertices $G_{bri}(N_q(q_i))$, and then merged with $N_x(v_i)$ to form the final out-neighbors $N_{out}(v_i)$ (line 14 in Algorithm 2). For example, as shown in Fig. 6(a), v_i is paired with q_i in the G_{bri}. After candidate acquisition and

Algorithm 3 GreedySearch(G, q, k, L)

```
1:  Input: graph index G, query q, results size k, candidates size L
2:  Output: top-k results of q
3:  e ← entry vertex in G;
4:  Heap ← {e};                                   ▷ Initialize candidates container
5:  while Heap has unvisited vertex do
6:      v ← closest unvisited vertex to q in Heap;          ▷ Find next hop
7:      mark vertex v as visited;
8:      N_out(v) ← out-neighbors of vertex v in G;
9:      Heap ← Heap ∪ N_out(v);                     ▷ Expand candidates
10:     if |Heap| > L then
11:         resize Heap to retain top-L vertices for q;
12:     end if
13: end while
14: return [closest k vertices from Heap]
```

neighbor selection (Fig. 6(b)), the out-neighbors $N_x(v_1)$ of v_1 include v_5, v_6, and v_7, while the out-neighbors $N_q(q_1)$ of q_1 include q_2, q_4, q_5, and q_6. In Fig. 6(c), the final out-neighbors $N_{out}(v_1)$ (v_2, v_4, v_5, v_6, v_7) are obtained by merging $N_x(v_1)$ and $G_{bri}(N_q(q_1))$ (v_2, v_4, v_5, v_6). However, edges in $N_x(v_i)$ and $N_q(q_i)$ are derived from different distributions, leading to clear differences in edge lengths. For example, OOD edges in LAION are 38% longer than ID edges, which causes *standard pruning to incorrectly discard vital long-range OOD edges, resulting in an uncontrollable final out-degree.*

Search Process. The search process of CDMG adopts the standard GreedySearch approach (Algorithm 3), which is commonly utilized in graph-based methods. GreedySearch uses a parameter L to regulate the capacity of the candidate container, referred to as Heap. The search initiates from an entry vertex e (line 4). In each iteration, GreedySearch selects the vertex v that is closest to the query from Heap, and computes the distances between query and the out-neighbors of v (lines 6–9). A new vertex is inserted into Heap if it is closer to query than the current candidates or if Heap has not yet reached its capacity L (lines 10–12). The search terminates when no closer vertex can be identified. Ultimately, the top-k vertices in Heap are returned as the retrieval results.

Complexity Analysis. We provide theoretical analysis of CDMG's graph index size (Lemma 2), construction complexity (Lemma 3), and search complexity (Lemma 4), as detailed below.

Lemma 2. *Given a database $\mathcal{X}$ and an OOD query set $\mathcal{Q}$. Let $\tau > 0$ be a constant, the CDMG $G(V, E)$ built on $\mathcal{X}$ and $\mathcal{Q}$ has expected out-degree $O(2 \cdot \ln |V|)$ and expected index size $O(2 \cdot |V| \ln |V|)$.*

Lemma 3. *Given a database $\mathcal{X}$ and an OOD query set $\mathcal{Q}$. Let relaxation parameter $\tau > 0$ be a constant, the construction complexity of CDMG $G(V, E)$ is $O(|V||\mathcal{Q}| + |\mathcal{Q}| \ln |\mathcal{Q}| + 2 \cdot |V|^2 \ln |V|)$.*

Lemma 4. *Given a database $\mathcal{X}$ and an OOD query set $\mathcal{Q}$ in the space $\mathbb{R}^D$, where D denotes the dimensionality, let τ be a constant. For an OOD query q_{ood} that follows the same distribution as $\mathcal{Q}$, and $\delta(q, n_1) < \tau$, the expected routing path length ℓ over the CDMG $G(V, E)$ is $O(|V|^{\frac{1}{D}} \ln |V|)$, and the search complexity is $O(|V|^{\frac{1}{D}} (\ln |V|)^2)$.*

Proof. The search complexity of graph-based methods can be expressed as $O(o\ell)$, where o denotes the expected out-degree and ℓ represents the routing path length. As established in Lemma 2, the expected out-degree of CDMG is $O(\ln |V|)$. We further show that the expected routing path length of CDMG ℓ is $O(|V|^{\frac{1}{D}} \ln |V|)$. The cross-distribution monotonic path $P = [P_1, P_{proj}]$ for

q_{ood} comprises two phases: (1) To estimate the path length of P_1, we consider an ID routing path $P^* = [P_1, P_{bri}, P_q]$ on the convex hull $\mathcal{X}^*$ that encloses both $\mathcal{X}$ and $\mathcal{Q}$. The points in $\mathcal{X}^*$ are uniformly distributed, with $|\mathcal{X}^*| = \lambda|V|$, where λ is a constant. Since q_{ood} follows the same distribution as $\mathcal{X}^*$, the length of P^* is $O(|\mathcal{X}^*|^{\frac{1}{D}} \ln |\mathcal{X}^*|)$. However, points within $B(q_{ood}, \epsilon)$ are not indexed, where ϵ denotes the distance between q_{ood} and its nearest database vector. Therefore, the length of P_1 is bounded by $O(|\mathcal{X}^*|^{\frac{1}{D}} \ln |\mathcal{X}^*| - \frac{\epsilon}{\tau})$. (2) The length of P_{proj} equals that of P_q. Since $|P_q| = \frac{\epsilon}{\tau} - |P_{bri}|$, the length of P_{proj} is upper bounded by $\frac{\epsilon}{\tau}$. Combining both phases, the total expected routing path length ℓ is $O(|V|^{\frac{1}{D}} \ln |V|)$, and the overall search complexity becomes $O(|V|^{\frac{1}{D}} (\ln |V|)^2)$. □

In summary, CDMG leverages cross-distribution monotonicity to address navigability and clusterability issues under OOD queries. However, it still faces three issues: (1) The initialization component requires a large query set, which significantly increases the cost. (2) The merging component is biased by the disparate edge lengths from the two distributions, leading to doubled construction time and an uncontrollable final out-degree. (3) Its candidate acquisition and neighbor selection rely on exhaustive search, and the introduced OOD query set further increases time complexity. *All factors increase construction overhead and affect search effectiveness, motivating us to design a practical variant of CDMG to overcome these limitations.*

3.3 CDMG+: A Practical and Efficient Approximation Algorithm of CDMG

Overview. We propose CDMG+, a practical and efficient construction algorithm designed to overcome the limitations of CDMG while achieving cross-distribution monotonicity. It consists of three main strategies: query synthesis, fusion-based distance computation, and optimized candidate acquisition and neighbor selection. We detail these strategies as follows.

Query Synthesis. The initialization component in CDMG relies on an impractically large OOD query set ($|\mathcal{Q}| \geq |\mathcal{X}|$) to ensure a one-to-one mapping between database vectors and OOD queries. To address this scalability issue, CDMG+ introduces a query synthesis strategy. Given a database $\mathcal{X}$ and a significantly smaller OOD query set $\mathcal{Q}$ ($|\mathcal{Q}| \ll |\mathcal{X}|$), the core objective of query synthesis is to generate a unique proxy query $\overline{q}_i \sim P_{ood}$ for each database vector x_i that statistically approximates the original OOD query distribution P_{ood}. This process effectively constructs the bipartite graph G_{bri} that records the one-to-one mapping between database vectors and their corresponding proxy queries at a significantly reduced cost.

Fusion-Based Distance Computation. In CDMG, separate distance calculations under two disparate data distributions lead to inconsistent edge scales between database-derived out-neighbors $N_x(v_i)$ and query-derived out-neighbors $N_q(q_i)$, thereby impeding unified edge selection. To address this, CDMG+ adopts a fusion-based distance computation strategy drawing inspiration from fusion techniques in hybrid search [9, 72, 73, 77, 83]. The core principle of fusion-based distance computation is to evaluate the connection between two vertices v_i and v_j by jointly considering their corresponding database vectors x_i and x_j and their proxy queries $\overline{q}_i$ and $\overline{q}_j$. The fusion-based distance is small only when both the database vectors and the proxy queries are close in their respective distributions. Conversely, if either pair is far apart, the fusion-based distance becomes large, thereby penalizing the connection. Thereby, CDMG+ enables a unified distance computation to select out-neighbors that simultaneously satisfy the edge selection criteria under both the database and query distributions, while maintaining a bounded out-degree of the final graph index.

Optimized Candidate Acquisition & Neighbor Selection. The candidate acquisition and neighbor selection in the CDMG are computationally expensive for two reasons: First, while the fusion-based distance unifies edge selection, calculating it on-the-fly necessitates jointly maintaining and processing both database vectors and their proxy queries, effectively doubling the computational load.

Algorithm 4 BuildCDMG+($\mathcal{X}, \mathcal{Q}, S, L, \tau, o$)

1: **Input:** database $\mathcal{X}$, OOD query set $\mathcal{Q}$, query synthesis parameter S, candidates size L, relaxation parameter τ, out-degree bound o
2: **Output:** CDMG+ G
3: $\mathcal{F} \leftarrow \emptyset$; ▷ Initialize the fusion vector set
4: **for** $\forall x_i \in \mathcal{X}$ **do**
5: $\mathsf{NN}(x_i) \leftarrow$ top-S queries to x_i in $\mathcal{Q}$;
6: $\overline{q}_i \leftarrow \frac{1}{S}\sum_{q \in \mathsf{NN}(x_i)} q$; ▷ Synthesize proxy query
7: $f_i \leftarrow x_i + \overline{q}_i$; ▷ Compute fusion vector
8: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_i\}$;
9: **end for**
10: $G \leftarrow (V \leftarrow \mathcal{F}, E \leftarrow \emptyset)$; ▷ Construct index over the fusion vector set
11: **for** $\forall v_i \in G$ **do**
12: $\mathsf{Cand} \leftarrow \mathsf{GreedySearch}(G, f_i, L, L)$; ▷ Approximate candidate acquisition
13: $N_{out}(v_i) \leftarrow \mathsf{RobustPruning}(v_i, \mathsf{Cand}, \mathcal{F}, \tau, o)$; ▷ Neighbor selection
14: add edges (v_i, v_j) to E for all $v_j \in N_{out}(v_i)$;
15: **end for**
16: **return** G

Second, standard candidate acquisition relies on exhaustive search to identify neighbor candidates. CDMG+ overcomes these limitations with a two-fold optimization: First, it eliminates redundant calculations by pre-fusing each database vector x_i and its proxy query $\overline{q}_i$ into a single fusion vector f_i, on which the entire graph index is constructed. Second, it replaces the exhaustive search with an efficient incremental process, using GreedySearch to retrieve a small set of approximate nearest neighbors for each inserted vertex as its candidate set Cand. This streamlined approach significantly reduces construction overhead.

We next detail the three construction strategies of CDMG+, which is illustrated in Algorithm 4:

Query Synthesis. For each database vector x_i, we retrieve its S nearest queries from $\mathcal{Q}$, denoted as $\mathsf{NN}(x_i)$ with $|\mathsf{NN}(x_i)| = S$, and compute the centroid of these queries as the proxy query: $\overline{q}_i = \frac{1}{S}\sum_{q \in \mathsf{NN}(x_i)} q$ (line 6). This is reasonable since the OOD query set $\mathcal{Q}$ serves merely as an auxiliary tool for index construction and is not part of the final graph index G. This approach is deliberately designed to satisfy the premises of the theoretical CDMG in a practical manner: First, by aggregating multiple real OOD queries that are proximate to x_i, the resulting proxy query $\overline{q}_i$ becomes statistically robust and more likely to be representative of the original OOD query distribution than a single, potentially noisy, query. Second, this process ensures that each database vector is associated with a unique proxy query, thereby establishing the requisite one-to-one mapping for the subsequent fusion-based distance computation.

Fusion-Based Distance Computation. Given database vectors in $\mathcal{X}$ and their corresponding proxy queries, we define the fusion-based distance $\xi(v_i, v_j)$ between vertices v_i and v_j as follows:

$$\xi(v_i, v_j) = \delta(x_i + \overline{q}_i, x_j + \overline{q}_j) \quad . \tag{3}$$

To assess its underlying mechanism, we decompose $\xi(v_i, v_j)$ under the assumption that δ represents the Euclidean distance, which facilitates insight into its underlying mechanisms. This leads to:

$$\begin{aligned} \xi(v_i, v_j) &= \|x_i + \overline{q}_i - (x_j + \overline{q}_j)\|^2 = \|(\overline{q}_i - x_j) - (\overline{q}_j - x_i)\|^2 \\ &= \delta(\overline{q}_i, x_j) + \delta(\overline{q}_j, x_i) - 2\|\overline{q}_i - x_j\|\|\overline{q}_j - x_i\|\cos\theta \quad , \end{aligned} \tag{4}$$

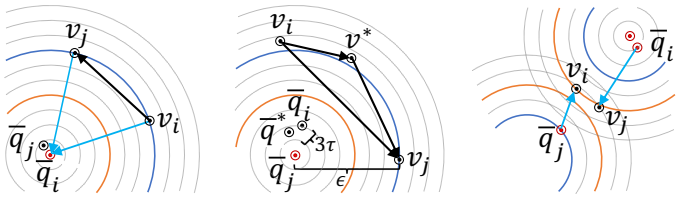


Fig. 7. Illustration of three different cases.

where θ denotes the angle between $\overline{q}_i - x_j$ and $\overline{q}_j - x_i$. We justify the efficacy of the fusion-based distance by deconstructing it into its individually *suboptimal* terms. This analysis demonstrates how the synthesis of these terms in the final formulation provides a more effective solution.

(1) Asymmetric Distance. For an OOD query q_{ood}, the search process determines whether a vertex v qualifies as a result based on the distance $\delta(q_{ood}, x)$. Consequently, a natural intuition is to use the asymmetric distance $\xi_A(v_i, v_j) = \delta(\overline{q}_i, x_j)$ to determine whether v_j should be considered a neighbor of v_i. This choice is consistent with the distance metric employed during the search process. As shown in Fig. 7(a), both v_i and v_j lie on the boundary of the database distribution. Although v_i and v_j are spatially distant, their asymmetric distances $\delta(\overline{q}_i, x_i)$ and $\delta(\overline{q}_i, x_j)$ to the proxy query $\overline{q}_i$ are identical (blue arrowed lines), permitting v_j to be selected as an out-neighbor of v_i (black arrowed lines). However, directly employing asymmetric distance $\delta(\overline{q}_i, x_j)$ as a measure overlooks the intrinsic relationships within the database and query distributions (i.e., $\delta(\overline{q}_i, \overline{q}_j)$ and $\delta(x_i, x_j)$). This leads to two critical issues: First, it violates the proposed edge occlusion rule. Second, it fails to account for database vectors that deviate significantly from the query distribution.

(2) Symmetric Distance. The symmetric distance form, $\xi_S(v_i, v_j) = \delta(\overline{q}_i, x_j) + \delta(\overline{q}_j, x_i)$, better captures the mutual relationships between queries and database vectors, and facilitates the preservation of cross-distribution monotonicity during construction. We validate its effectiveness by demonstrating that the edge occlusion rules of CDMG+ can be reformulated as follows: (1) if $\xi_S(v_i, v_j) \leq 2\epsilon + 3\tau$, a direct link from v_i to v_j is established, where ϵ denotes the distance between an OOD query and its nearest database vector; (2) otherwise, v_i links to v_j only if the occlusion condition $\xi_S(v_j, v^*) + 2\epsilon + 3\tau > \xi_S(v_i, v_j)$ holds. This formulation is grounded in the observation that when $\delta(\overline{q}_i, \overline{q}_j) \leq 3\tau$, vertices v_i and v_j are located near the boundary of the database distribution, implying both $\delta(\overline{q}_i, x_j)$ and $\delta(\overline{q}_i, x_i)$ approximate the constant ϵ (Fig. 7(a)). Therefore, employing the symmetric distance $\xi_S(v_i, v_j)$ yields:

$$\xi_S(v_i, v_j) = \delta(\overline{q}_i, x_j) + \delta(\overline{q}_j, x_i) \leq \epsilon + \epsilon + \delta(\overline{q}_i, \overline{q}_j) \leq 2\epsilon + 3\tau \quad . \tag{5}$$

Furthermore, for cases where $\delta(\overline{q}_i, \overline{q}_j) \geq 3\tau$, and $\delta(x^*, x_i) < 2\epsilon$. Given $\delta(\overline{q}_i, \overline{q}_j) - \delta(\overline{q}_j, \overline{q}^*) \leq 3\tau$, we derive that:

$$\begin{aligned} \xi_S(v_i, v_j) - \xi_S(v_j, v^*) &= \delta(\overline{q}_i, x_j) + \delta(\overline{q}_j, x_i) - \delta(\overline{q}_j, x^*) - \delta(\overline{q}^*, x_j) \\ &\leq (\epsilon + \delta(\overline{q}_i, \overline{q}_j)) + (\delta(\overline{q}_j, x^*) + 2\epsilon) - \delta(\overline{q}_j, x^*) - (\epsilon + \delta(\overline{q}^*, \overline{q}_j)) \\ &\leq 2\epsilon + \delta(\overline{q}_i, \overline{q}_j) - \delta(\overline{q}^*, \overline{q}_j) \leq 2\epsilon + 3\tau \quad . \end{aligned} \tag{6}$$

However, although symmetric distance preserves cross-distribution monotonicity in the second routing phase, it still has two main limitations: First, it is effective only when the corresponding queries are approximately co-directional. As illustrated in Fig. 7(c), when the queries for v_i and v_j are counter-directional, v_j may still be mistakenly selected as a neighbor. Second, it fails to adequately handle database vectors located far from the distribution boundary.

(3) Fusion-Based Distance. The limitations of the symmetric distance $\xi_S(v_i, v_j)$, particularly in counter-directional cases, are mitigated by the angular correction term: $-2\|\overline{q}_i - x_j\|\|\overline{q}_j - x_i\| \cos\theta$. In counter-directional scenarios, $\cos\theta$ is negative, causing this term to become positive. This effectively increases the total distance, thereby penalizing counter-directional query pairs while preserving the desired behavior for co-directional ones. Furthermore, for the vertices that are close to each other yet deviate from the distribution boundary, their corresponding proxy queries are highly similar or even identical. As a result, their fusion-based distance can be approximated as $\xi(v_i, v_j) \approx \delta(x_i, x_j)$. It is trivial to prove that it guarantees both clusterability and navigability.

Optimized Candidate Acquisition & Neighbor Selection. Each database vector x_i is pre-fused with its corresponding proxy query $\overline{q}_i$ to form the fusion set $\mathcal{F}$, where each fusion vector in $\mathcal{F}$ is defined as $f_i = x_i + \overline{q}_i$ (line 7 in Algorithm 4). Based on the fusion set $\mathcal{F}$, CDMG+ incrementally constructs the graph index. Specifically, for each vertex v_i, it is treated as a query, and GreedySearch is used to retrieve the top-L approximate nearest neighbors as the candidate set Cand of v_i (line 12), avoiding exhaustive comparisons. The search starts from an entry vertex e and moves toward the nearest neighbors of v_i by comparing distances $\delta(f_i, f^*)$, where f^* denotes the fusion vector of a candidate in Heap. After candidate acquisition, a single round of RobustPruning is applied to prune the candidate set Cand to satisfy the out-degree of $N_{out}(v_i)$ to o (line 13). Once all vertices are processed, BuildCDMG+ returns G as the final index.

Remarks. We clarify the metric assumptions underpinning our theoretical framework, and the distinction between the vectors used in the construction and search processes, as follows: (1) Our theoretical derivations are established in Euclidean space, as its metric properties facilitate the geometric guarantees for navigability and monotonicity. This aligns with standard practices in graph-based indexes. For instance, HNSW is theoretically derived under Euclidean distance yet widely deployed for inner product and cosine similarity. Furthermore, it is a well-established fact that Maximum Inner Product Search (MIPS) can be formally reduced to an equivalent Euclidean Nearest Neighbor Search (NNS) problem [12, 24]. Since these transformations occur during standard pre-processing, they do not impact the graph traversal or comparative performance. (2) Crucially, although CDMG+ is constructed on the fusion vector set $\mathcal{F}$, GreedySearch still relies on the distances $\delta(q_{ood}, x_i)$ between the *original* query q_{ood} and the *original* database vectors $x_i \in \mathcal{X}$.

Complexity Analysis. The construction of CDMG+ is a multi-stage process where the overall complexity is determined by the sum of its steps. Initially, query synthesis takes $O(|V||Q|)$ time to find nearest queries for each database vector, followed by a linear $O(|V|)$ pre-fusion step. Subsequently, the incremental graph construction iterates through each of the $|V|$ vertices, where for each one, candidate acquisition is performed via a greedy search and costs $O(|V|^{\frac{1}{D}} \ln|V|)$, neighbor selection is then applied with a complexity of $O(L \cdot (\ln L + o))$. Since parameters like $|\boldsymbol{Q}|$, L, and o are typically small, the overall construction time is $O(|V|^{\frac{(1+D)}{D}} \ln|V|)$ which is comparable to mainstream MRNG variants. For search complexity, CDMG+ maintains the strong theoretical efficiency of CDMG while supporting flexible out-degree adjustment to improve practical performance. This flexible out-degree bound enables CDMG+ to achieve a better balance between search speed and accuracy in practical high-dimensional data retrieval scenarios.

4 EXPERIMENTAL STUDY

In this section, we evaluate our proposed CDMG+ with a comprehensive experimental study to answer the following questions: (1) How do CDMG+ and other graph-based methods perform for OOD queries (**§4.2**)? (2) How does CDMG+ impact the index size and construction time (**§4.3**)? (3) To what extent does CDMG+ optimize the navigability and clusterability of the graph index (**§4.4**)? (4) How do different strategies contribute to the search performance of CDMG+ (**§4.5**)? (5) How

Table 2. Dataset statistics.

Dataset	Dimensions	Distance metric	# Database	# Query	Modalities
Text-to-Image [3]	200	IP	1M∼10M	1,000	Image, Text
LAION [52]	512	COS	1M∼10M	1,000	Image, Text
WIT [63]	512	IP	1M	1,000	Image, Text
WebVid [5]	512	COS	1M∼2.5M	1,000	Video, Text
CC3M [56]	256	IP	1M	1,000	Image, Text

do parameters and query set size affect CDMG+ (**§4.6**)? (6) How does CDMG+ scale with dataset size (**§4.7**)? Our source code and datasets are available at: https://github.com/Lsyhprum/CDMG.

4.1 Experimental Setup

Datasets. We evaluate our proposed method on five modern large-scale OOD datasets [11, 65], as summarized in Table 2. These datasets vary in dimensions, metrics, modality, and OOD properties. As described in §2.1, we employ Mahalanobis distance histograms to characterize the OOD properties of queries w.r.t. the database. Following the setup in [11, 31], we use the provided training set to construct the graph index and evaluate performance on the provided query set. Although both sets are sampled from the same query distribution, they are mutually exclusive.

Baselines. We compare CDMG+ with several state-of-the-art graph-based methods known for their efficiency in vector similarity search. All algorithms are obtained from the official source codes, implemented in C++, and compiled using GCC 9.4.0.

- **HNSW** [45]: A well-known hierarchical graph-based method.
- **τ-MNG** [50]: A MRNG variant that incorporates query relaxation to enhance performance.
- **RobustVamana** [31]: A specialized variant of Vamana [32] tailored for handling OOD queries. Unless otherwise specified, the size of the queries is 10% of the database.
- **RoarGraph** [11]: RoarGraph is a recently proposed efficient graph index designed for OOD queries. The size of the queries is the same as in RobustVamana, representing 10% of the database.

The standard benchmark datasets for OOD vector similarity search primarily use inner product (IP) and cosine similarity (COS) as their evaluation metrics. Although mainstream baselines, such as HNSW, are theoretically grounded in Euclidean distance, they are typically validated in experiments directly on these standard IP and COS datasets. We follow this same established methodology. While our graph method is also theoretically grounded in Euclidean space, we use the same IP and COS metrics in our experiments as the baselines [11, 31, 88]. This ensures that our comparisons are fair and consistent with the standard practice in the field.

Evaluation Metrics. We evaluate the search accuracy using *Recall@k*, which is a widely used metric for vector similarity search [17, 22]. We evaluate search efficiency with the queries per second (*QPS*) and the number of distance computations (*NDC*) [50, 75], where *QPS* measures the number of handled queries per second, and *NDC* quantifies computational cost. Additionally, we introduce the number of routing hops (*HOP*) [11] as a supplementary metric to evaluate graph index navigability, and use *QNNQ@k* to measure graph index clusterability.

Implementation Setup. All experiments are conducted on an Ubuntu 20.04 server equipped with Intel(R) Xeon(R) Gold 5218 CPU (2.30GHz) and 125GB of memory. The Eigen library [20] enhances matrix operations, while OpenMP enables parallel index construction execution using 20 threads. All search performance is evaluated in a single-threaded environment.

4.2 Efficiency and Effectiveness

Fig. 8 presents the performance comparison of the proposed CDMG+ against existing graph-based methods. We evaluate accuracy using *Recall@100* (rows 1 and 2) and *Recall@10* (rows 3 and 4) and

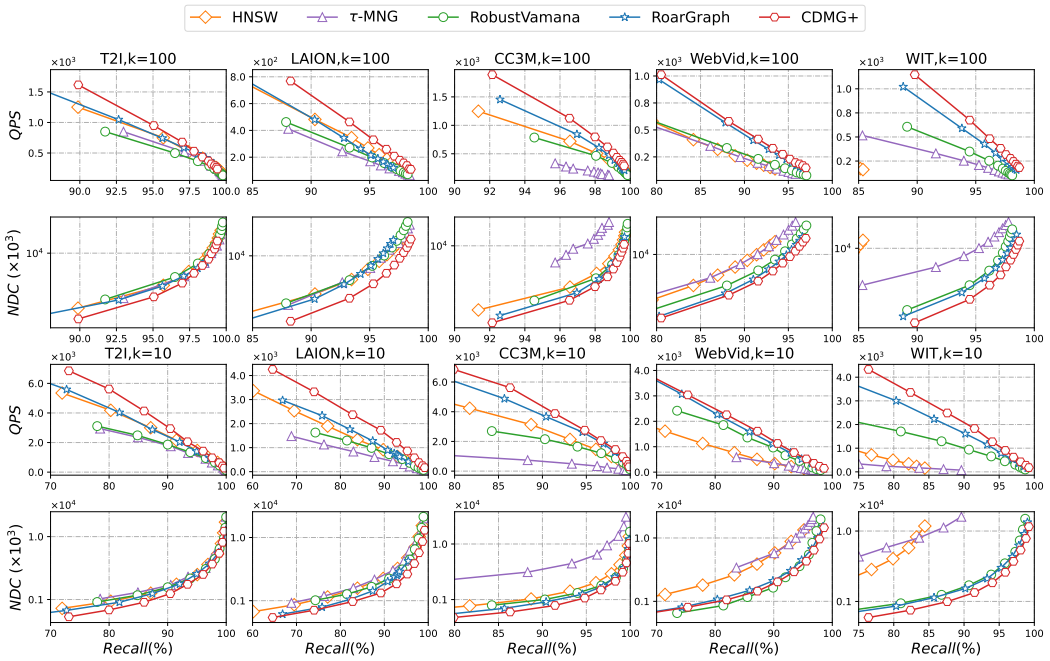


Fig. 8. Efficiency (*QPS* and *NDC*) and effectiveness (*Recall@k*) of all algorithms across five OOD datasets: For the first and third rows, the top-right is better; for the second and fourth rows, the bottom-right is better.

measure search efficiency with *QPS* and *NDC*. In the figures illustrating *Recall@k* vs. *QPS* (rows 1 and 3), a curve positioned towards the top-right indicates better performance. Conversely, in those depicting *Recall@k* vs. *NDC* (rows 2 and 4), a position toward the bottom-right is preferable. The results demonstrate that our proposed algorithm consistently outperforms existing methods across all datasets. A detailed quantitative comparison at *Recall@100*=95% and *Recall@10*=95% is presented in Table 3 and Table 4, respectively. CDMG+ achieves up to a 3.6× gain in *QPS* and up to a 68% reduction in *NDC* compared to HNSW. Notably, HNSW fails to reach the 95% recall target on the WIT dataset. Among the baseline methods, ID-oriented graph-based methods perform worse than OOD-oriented methods. Although RobustVamana and RoarGraph are specifically designed for OOD queries, their improvements remain limited, particularly on Text-to-Image, LAION, and CC3M. This is primarily attributed to their reliance on heuristic optimization, often leading to unstable and suboptimal performance. In contrast, CDMG+ exhibits adaptability across all OOD datasets. It is also worth noting that while HNSW is designed for ID queries, it achieves competitive results on certain datasets, likely due to its hierarchical structure that enables more effective entry selection from a larger distance scale, which aligns with recent research progress (see §5 for details).

Table 3. *QPS* (↑) and *NDC* (↓) at *Recall@100*=95%.

Algorithm	Text-to-Image		LAION		CC3M		WebVid		WIT	
	QPS	NDC	QPS	NDC	QPS	NDC	QPS	NDC	QPS	NDC
HNSW	750	4675	269	7363	719	3766	103	18107	-	-
τ-MNG	654	4815	169	7593	331	6736	110	17943	208	10035
RobustVamana	498	5684	197	7960	654	3104	196	11218	348	4498
RoarGraph	742	4588	215	7987	842	3294	195	11217	425	5211
CDMG+	**948**	**3555**	**329**	**5297**	**1124**	**2757**	**229**	**9496**	**674**	**3313**
Gain	1.3×	24%	1.2×	28%	1.6×	27%	2.2×	48%	-	-

Table 4. *QPS* (↑) and *NDC* (↓) at *Recall@10*=95%.

Algorithm	Text-to-Image		LAION		CC3M		WebVid		WIT	
	QPS	NDC	QPS	NDC	QPS	NDC	QPS	NDC	QPS	NDC
HNSW	1253	2992	416	4751	1467	2010	143	13378	-	-
τ-MNG	902	3163	216	6184	340	6426	140	14107	-	-
RobustVamana	948	3062	365	4465	1206	1789	287	5469	448	3478
RoarGraph	945	3739	420	4246	1845	1755	340	6675	789	3004
CDMG+	**1420**	**2503**	**862**	**2638**	**2754**	**1065**	**518**	**4305**	**904**	**2595**
Gain	1.1×	16%	2.1×	45%	1.9×	47%	3.6×	68%	-	-

4.3 Index Cost

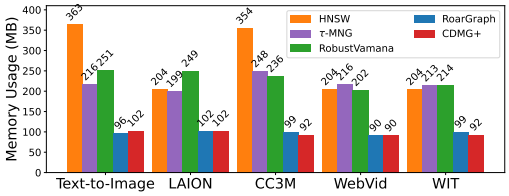


Fig. 9. Index size.

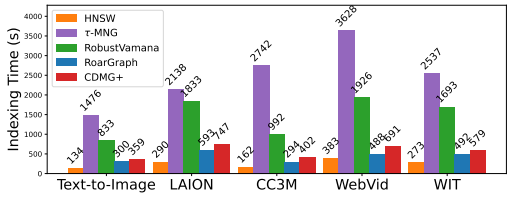


Fig. 10. Indexing time.

Regarding index cost, our proposed CDMG+ demonstrates remarkable efficiency in both memory usage and construction time. As illustrated in Fig. 9 and Fig. 10, CDMG+ achieves a highly compact index size, making it exceptionally suitable for resource-constrained environments, especially on the Text-to-Image, LAION, and WIT datasets. In contrast, other methods incur greater memory usage. For example, HNSW's hierarchical structure and τ-MNG's query relaxation result in larger graphs. The index for RobustVamana is particularly large, ranging from 2.24× to 2.57× larger than that of CDMG+, primarily because it retains numerous edges from the database distribution and further augments them with additional query-derived edges, which inflates the average out-degree from 22 to 57. Beyond its memory efficiency, CDMG+ is also highly competitive in terms of indexing time, building the index 56.9% to 65.8% faster than RobustVamana. While the conceptual foundation of its theoretical counterpart, CDMG, involves a higher overhead by design, the practical CDMG+ implementation effectively mitigates this. Through the introduction of a fusion-based distance, CDMG+ delivers significantly superior index quality with an index cost comparable to the most optimized methods such as RoarGraph.

4.4 Navigability and Clusterability

We conduct a comparison of the navigability and clusterability of CDMG+ against baseline methods.

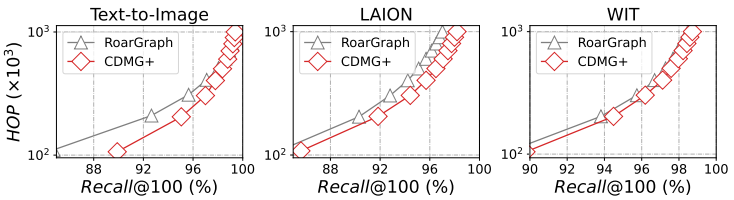


Fig. 11. Hop statistics of RoarGraph and CDMG+.

Navigability. We compare the number of hops *HOP* required by different algorithms to achieve the target *Recall@100* on the Text-to-Image, LAION, and WIT datasets. To ensure a fair comparison, we control the out-degree of the compared algorithms to be approximately the same. Specifically, we select RoarGraph as the baseline algorithm, and the average out-degree of CDMG+ and RoarGraph

are 21 compared with 23, 25 compared with 25, and 23 compared with 24 on the three datasets, respectively. As shown in Fig. 11, CDMG+ reaches the target *Recall* with fewer hops than RoarGraph. For example, on the LAION dataset, the *HOP* of RoarGraph is 1.25× that of CDMG+ at 95% *Recall@100*. This result demonstrates that CDMG+ effectively optimizes the graph structure, enabling faster convergence to query-relevant regions.

Table 5. Clusterability statistics (*QNNQ@100*).

Method	Text-to-Image	LAION	CC3M	WEBVID	WIT
$KNNG_{id}$	26.34%	20.06%	27.44%	15.45%	21.06%
$KNNG_{projection}$	25.95%	21.31%	32.95%	16.69%	**26.97%**
$KNNG_{fusion}$ (ours)	**29.38%**	**25.92%**	**32.97%**	**18.93%**	25.90%

Clusterability. To evaluate clusterability, we isolate the effect of the candidate acquisition phase from neighbor selection. We constructed three types of KNNG, all built without pruning: (1) a standard $KNNG_{id}$ built without any query information; (2) a $KNNG_{projection}$ optimized using the query projection preprocessing from RoarGraph; and (3) a $KNNG_{fusion}$ using our method. We then employ *QNNQ@100* to measure the clustering quality of OOD query results on these graphs. As illustrated in Table 5, $KNNG_{id}$ yields consistently low *QNNQ@100* scores, as it fails to capture the neighbor structure of OOD queries. $KNNG_{projection}$ offers inconsistent performance; while it shows benefits on some datasets (like CC3M and WIT), it provides only marginal gains elsewhere and even underperforms the simple baseline on the Text-to-Image dataset. In contrast, our $KNNG_{fusion}$ directly refines the representation of every database vector, demonstrating substantial and more reliable improvements. It achieves the highest clusterability on 4 out of 5 datasets, delivering significant *QNNQ@100* quality increases over the baseline.

4.5 Ablation Study

As shown in Fig. 12, we present the performance comparison between CDMG and CDMG+, as well as the differences among $CDMG_A$, $CDMG_S$, and CDMG+, which are constructed using asymmetric, symmetric, and fusion-based distances, respectively.

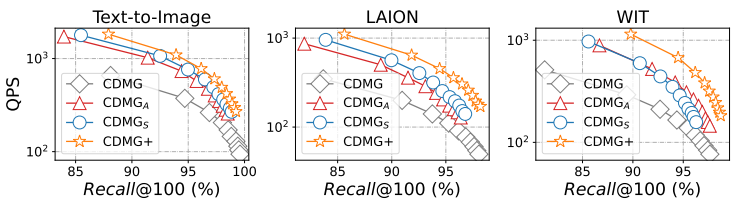


Fig. 12. Ablation comparisons.

CDMG simply merges the out-neighbors obtained from both distributions without an effective comparison process, resulting in a significantly higher out-degree. For example, on the Text-to-Image dataset, CDMG exhibits a 1.3× increase in out-degree compared to CDMG+. In contrast, CDMG+ effectively constrains the out-degree, thereby reducing the distance computations during the search process. $CDMG_A$ performs the worst on the Text-to-Image and LAION datasets, as it captures only partial clusterability and fails to maintain navigability. $CDMG_S$ improves performance by better preserving navigability. However, on the WIT dataset, it underperforms the asymmetric variant—likely due to its vulnerability in cases where database vectors are spatially close but their corresponding queries are counter-directional. CDMG+ introduces a corrective term and ensures both clusterability and navigability, achieving the best search performance across all datasets.

4.6 Parameter Sensitivity

In this section, we investigate the sensitivity of CDMG+ to key parameters. We specifically examine the impact of the query set size $|Q|$, the query relaxation parameter τ, and the query synthesis parameter S on search performance.

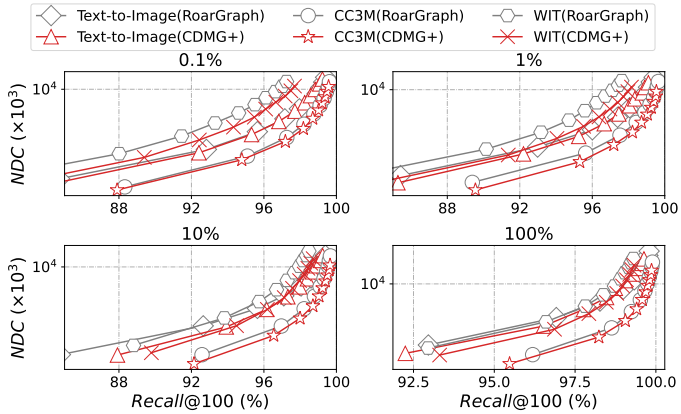


Fig. 13. Performance comparison across different query set sizes for index construction.

Query Set Size $|Q|$. We evaluate search performance under varying query set sizes—ranging from 0.1%, 1%, 10%, to 100% of the database—on the Text-to-Image, CC3M, and WIT datasets. As illustrated in Fig. 13, we analyze the trade-off between *Recall@100* and *NDC*. The results demonstrate that CDMG+ consistently enhances search performance across all query set sizes, maintaining a clear advantage over the baseline, RoarGraph. Notably, a query set size of 1% strikes a favorable balance between search efficiency and accuracy.

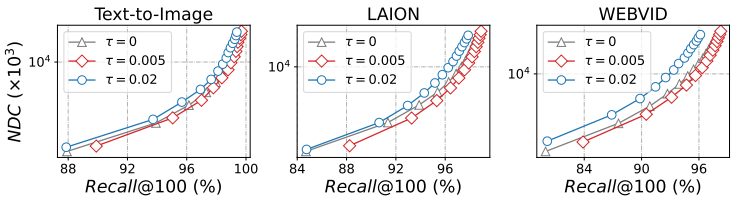


Fig. 14. Impacts from different parameter τ.

Query Relaxed Parameter τ. We analyze the impact of the query relaxation parameter τ on search performance. As illustrated in Fig. 14, A smaller τ enforces a larger angular gap between connected edges, resulting in a sparser graph structure. While this promotes directional diversity, aggressive pruning increases the risk of bypassing the optimal next-hop, potentially leading to an increased number of routing hops. Conversely, a larger τ permits more spatially similar neighbors, thereby increasing the likelihood of retaining the optimal next-hop. However, this reduces navigation capability in diverse directions. Consistent with observations in methods like τ-MNG, both excessively large and small τ values can degrade performance. In practice, the optimal τ is dataset-dependent and should be determined via grid search or similarparameter tuning methods.

Query Synthesis Parameter S. We analyze the performance impact of the parameter S during query synthesis. As shown in Fig. 15, we compare different configurations with $S \in \{1, 5, 10, 15, 30\}$. Search performance generally improves as S increases, as averaging multiple vectors helps mitigate the sparsity of smaller training sets and yields more robust proxy queries. Empirically, $S = 15$ proves to be a robust choice across all datasets. However, as S increases, the improvement in search

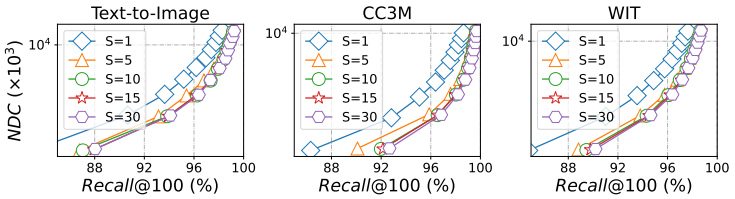


Fig. 15. Impacts from different parameter S.

performance gradually diminishes. Compared to $S = 15$, larger S values provide only marginal gains at low recall levels, while significantly increasing the computational cost of construction.

4.7 Scalability

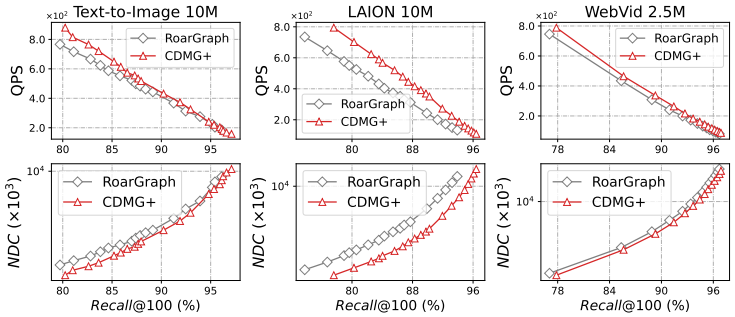


Fig. 16. Scalability evaluation.

Table 6. Indexing time (s) and size (MB) comparison.

Algorithm	Text-to-Image 10M		LAION 10M		WebVid 2.5M	
	Time	Size	Time	Size	Time	Size
RoarGraph	2744	959	5603	876	1226	214
CDMG+	**2218**	**860**	**4678**	**842**	**1062**	**205**

We further evaluate the scalability of CDMG+ on large-scale datasets, including Text-to-Image 10M, LAION 10M, and WebVid 2.5M. Consistent with prior studies [22, 23], the KNNG for preprocessing were constructed using a single NVIDIA GeForce RTX 3090 GPU. As illustrated in Fig. 16, CDMG+ consistently outperforms the state-of-the-art RoarGraph, exhibiting a superior trade-off between search accuracy (*Recall@100*) and efficiency (*QPS* and *NDC*) across all datasets. Beyond search performance, CDMG+ demonstrates significant advantages in construction costs as detailed in Table 6. Notably, on the Text-to-Image 10M dataset, CDMG+ reduces indexing time by 19.2% and memory usage by 10.3% compared to RoarGraph, thereby validating its robustness and resource efficiency in large-scale OOD scenarios.

5 RELATED WORK

OOD Vector Similarity Search. Numerous algorithms have been proposed to address the vector similarity search problem, with the dominant index structures being tree [8, 58], inverted index (IVF) [4, 62], and graph [23, 50]. However, all of them experience significant performance degradation under OOD scenarios. (1) Tree-based methods recursively partition the space and traverse the most promising branches during search. They are effective in low-dimensional settings but suffer from the curse of dimensionality [75]. Moreover, the poor clusterability of OOD results forces

costly backtracking across distant branches. (2) IVF-based methods cluster the database vectors and assign each vector to its nearest centroid. At query time, IVF selects the closest centroids and scans the corresponding clusters. Similar to tree methods, the poor clusterability forces the algorithm to examine more clusters to achieve high recall, thereby increases latency [31]. (3) Graph-based methods achieve a strong balance between accuracy and latency in both ID and OOD scenarios. Their flexible connectivity permits low-cost backtracking through localized candidate exploration [29], rather than the expensive jumps across partitions required by tree or IVF.

OOD-Oriented Graph-Based Methods. To address the performance degradation in OOD vector similarity search, RobustVamana [31] integrates a subset of the query set during construction and establishes connections between database vectors that co-occur as out-neighbors of an indexed query. RoarGraph [11] constructs its index guided by query distribution, first building a query-base graph, then projecting it onto base vectors, and finally adding neighbors to enhance connectivity. However, these methods rely on local, heuristic optimizations over a small subset of vertices, while our method explicitly considers the navigability and clusterability of each vertex under the query distribution. Recent studies have also explored optimization methods orthogonal to index. B-MSNET [47] shows that optimizing entry selection can improve OOD search efficiency. GATE [53] learns a set of topology-aware and query-aware entry points to shorten search paths. Another optimization direction is to optimize graph vectors via query-aware quantization and dimensionality reduction. For example, APQ [31] learns codebooks by directly minimizing the distance distortion between queries and database vectors. LeanVec-OOD [65] learns and jointly optimizes two linear projection matrices—one for the database and one for queries—by minimizing the similarity error.

6 DISCUSSION

Query Distribution Adaptation. CDMG+ requires online and construction query distributions to be consistent to ensure effective monotonic search. This results in performance degradation when the two distributions differ. For example, according to our experiments, both CDMG+ and other OOD-oriented graph indexes exhibit slight performance drops under ID queries. Therefore, a key direction is to enhance CDMG+ to efficiently support ID queries. Moreover, handling distribution drift remains an open challenge. In many real-world applications, query distributions evolve over time [6, 87], leading to OOD effects between the pre-deployed index and newly arriving queries.

Hybrid Optimization. A promising direction for further work is to jointly optimize entry point selection [47], dimensionality reduction [65, 66], quantization [31], and hardware-specific acceleration [27] under OOD scenarios, in conjunction with CDMG+.

7 CONCLUSION

This work presents CDMG+, a theoretically grounded (with CDMG as its theoretical counterpart) and computationally efficient graph index designed for OOD queries. By enforcing cross-distribution monotonicity, CDMG guarantees reliable and efficient search paths even under significant distributional shifts between database and query vectors. Theoretical analysis establishes that CDMG achieves a state-of-the-art search time complexity for OOD scenarios, while the practical refinements in CDMG+ ensure scalable and robust performance. Extensive experiments demonstrate the superiority of our approach, achieving substantial gains in speed and accuracy over existing methods across diverse benchmarks.

Acknowledgments

This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2024C01020) and National Natural Science Foundation of China (No. 62572162).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Martin Aumüller and Matteo Ceccarello. 2023. Recent Approaches and Trends in Approximate Nearest Neighbor Search, with Remarks on Benchmarking. *IEEE Data Eng. Bull.* 46, 3 (2023), 89–105.

[3] Artem Babenko and Dmitry Baranchuk. 2021. Text-to-Image Dataset for Billion-Scale Similarity Search. Retrieved August 23, 2023, from https://research.yandex.com/datasets/text-to-image-dataset-for-billion-scale-similarity-search.

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

[5] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1728–1738.

[6] Dmitry Baranchuk, Matthijs Douze, Yash Upadhyay, and I Zeki Yalniz. 2023. Dedrift: Robust similarity search under content drift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11026–11035.

[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] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. 322–331.

[9] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems* 42, 1 (2023), 1–35.

[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] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X Sean Wang. 2024. RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2735–2749.

[12] Tingyang Chen, Cong Fu, Kun Wang, Xiangyu Ke, Yunjun Gao, Wenchao Zhou, Yabo Ni, and Anxiang Zeng. 2025. Maximum Inner Product is Query-Scaled Nearest Neighbor. *Proc. VLDB Endow.* 18, 6 (Aug. 2025), 1770–1783.

[13] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[14] Yijie Dang, Nan Jiang, Hao Hu, Zhuoxiao Ji, and Wenyin Zhang. 2018. Image classification based on quantum K-Nearest-Neighbor algorithm. *Quantum Information Processing* 17 (2018), 1–18.

[15] Haya Diwan, Jinrui Gou, Cameron Musco, Christopher Musco, and Torsten Suel. 2024. Navigable graphs for high-dimensional nearest neighbor search: Constructions and limits. *Advances in Neural Information Processing Systems* 37 (2024), 59513–59531.

[16] Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*. 577–586.

[17] Matthijs Douze, Alexandre Sablayrolles, and Hervé Jégou. 2018. Link and code: Fast indexing with graphs and compact regression codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3646–3654.

[18] Karima Echihabi. 2020. High-dimensional vector similarity search: from time series to deep network embeddings. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2829–2832.

[19] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. New trends in high-d vector similarity search: al-driven, progressive, and distributed. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3198–3201.

[20] Eigen Library. 2009. Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. https://eigen.tuxfamily.org/index.php?title=Main_Page. [Online; accessed 01-June-2024].

[21] Cole Foster, Berk Sevilmis, and Benjamin Kimia. 2025. Generalized relative neighborhood graph (GRNG) for similarity search. *Pattern Recognition Letters* 188 (2025), 103–110.

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

[23] 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.

[24] 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.

[25] Jianyang Gao and Cheng Long. 2023. High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.

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

[27] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*. PMLR, 3887–3896.

[28] Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703* (2023).

[29] Ben Harwood and Tom Drummond. 2016. Fanng: Fast approximate nearest neighbour graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5713–5722.

[30] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[31] Shikhar Jaiswal, Ravishankar Krishnaswamy, Ankit Garg, Harsha Vardhan Simhadri, and Sheshansh Agrawal. 2022. Ood-diskann: Efficient and scalable graph anns for out-of-distribution queries. *arXiv preprint arXiv:2211.12850* (2022).

[32] 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).

[33] Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*. 36–45.

[34] Jon M Kleinberg. 2000. Navigation in a small world. *Nature* 406, 6798 (2000), 845–845.

[35] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. 2020. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 2880–2894.

[36] Mr Ramesh Krishnamaneni and AN Murthy. 2021. Advancing Drug Dealing Detection Using Neural Embedding and Nearest Neighbour Searching Techniques. *International Journal on Recent and Innovation Trends in Computing and Communication* 9, 7 (2021), 19–23.

[37] 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.

[38] Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. 2022. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in Neural Information Processing Systems* 35 (2022), 17612–17625.

[39] Weizhe Lin, Jinghong Chen, Jingbiao Mei, Alexandru Coca, and Bill Byrne. 2023. Fine-grained late-interaction multi-modal retrieval for retrieval augmented visual question answering. *Advances in Neural Information Processing Systems* 36 (2023), 22820–22840.

[40] Jie Liu, Xiao Yan, Xinyan Dai, Zhirong Li, James Cheng, and Ming-Chang Yang. 2020. Understanding and improving proximity graph based maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 139–146.

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

[42] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. 2015. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling* 55, 2 (2015), 263–274.

[43] Zi-Ao Ma, Tian Lan, Rong-Cheng Tu, Yong Hu, Heyan Huang, and Xian-Ling Mao. 2024. Multi-modal Retrieval Augmented Multi-modal Generation: A Benchmark, Evaluate Metrics and Strong Baselines. *arXiv preprint arXiv:2411.16365* (2024).

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

[45] 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.

[46] Stanley Milgram. 1967. The small world problem. *Psychology today* 2, 1 (1967), 60–67.

[47] Yutaro Oguri and Yusuke Matsui. 2024. Theoretical and Empirical Analysis of Adaptive Entry Point Selection for Graph-based Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2402.04713* (2024).

[48] Hiroyuki Ootomo, Akira Naruse, Corey Nolet, Ray Wang, Tamas Feher, and Yong Wang. 2024. Cagra: Highly parallel graph construction and approximate nearest neighbor search for gpus. In *2024 IEEE 40th International Conference on*

Data Engineering (ICDE). IEEE, 4236–4247.

[49] James Jie Pan, Jianguo Wang, and Guoliang Li. 2024. Survey of vector database management systems. *The VLDB Journal* 33, 5 (2024), 1591–1615.

[50] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient approximate nearest neighbor search in multi-dimensional databases. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.

[51] Samira Pouyanfar, Yimin Yang, Shu-Ching Chen, Mei-Ling Shyu, and SS Iyengar. 2018. Multimedia big data analytics: A survey. *ACM computing surveys (CSUR)* 51, 1 (2018), 1–34.

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

[53] Jiancheng Ruan, Tingyang Chen, Renchi Yang, Xiangyu Ke, and Yunjun Gao. 2025. Empowering Graph-based Approximate Nearest Neighbor Search with Adaptive Awareness Capabilities. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2* (Toronto ON, Canada) *(KDD '25)*. Association for Computing Machinery, New York, NY, USA, 2444–2454.

[54] Andrey V Savchenko. 2017. Maximum-likelihood approximate nearest neighbor method in real-time image recognition. *Pattern Recognition* 61 (2017), 459–469.

[55] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114* (2021).

[56] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning. In *Proceedings of ACL*.

[57] Peiyang Shi, Michael C Welle, Mårten Björkman, and Danica Kragic. 2023. Towards understanding the modality gap in CLIP. In *ICLR 2023 Workshop on Multimodal Representation Learning: Perks and Pitfalls*.

[58] Chanop Silpa-Anan and Richard Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 1–8.

[59] Harsha Vardhan Simhadri, Martin Aumüller, Amir Ingber, Matthijs Douze, George Williams, Magdalen Dobson Manohar, Dmitry Baranchuk, Edo Liberty, Frank Liu, Ben Landrum, et al. 2024. Results of the Big ANN: NeurIPS'23 competition. *arXiv preprint arXiv:2409.17424* (2024).

[60] Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamny, Gopal Srinivasa, et al. 2022. Results of the NeurIPS'21 challenge on billion-scale approximate nearest neighbor search. In *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, 177–189.

[61] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. Freshdiskann: A fast and accurate graph-based ann index for streaming similarity search. *arXiv preprint arXiv:2105.09613* (2021).

[62] Sivic and Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *Proceedings ninth IEEE international conference on computer vision*. IEEE, 1470–1477.

[63] Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. 2021. Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 2443–2449.

[64] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1165–1174.

[65] Mariano Tepper, Ishwar Singh Bhati, Cecilia Aguerrebere, Mark Hildebrand, and Theodore L. Willke. 2024. LeanVec: Searching vectors faster by making them fit. *Transactions on Machine Learning Research* (2024). Featured Certification.

[66] Mariano Tepper, Ishwar Singh Bhati, Cecilia Aguerrebere, and Ted Willke. 2024. GleanVec: Accelerating vector search with minimalist nonlinear dimensionality reduction. *arXiv preprint arXiv:2410.22347* (2024).

[67] Sergios Theodoridis and Konstantinos Koutroumbas. 2006. *Pattern recognition*. Elsevier.

[68] Bing Tian, Haikun Liu, Yuhang Tang, Shihai Xiao, Zhuohui Duan, Xiaofei Liao, Hai Jin, Xuecang Zhang, Junhua Zhu, and Yu Zhang. 2025. Towards High-throughput and Low-latency Billion-scale Vector Search via {CPU/GPU} Collaborative Filtering and Re-ranking. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*. 171–185.

[69] 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 Eng. Bull.* 46, 3 (2023), 39–54.

[70] Hongya Wang, Zhizheng Wang, Wei Wang, Yingyuan Xiao, Zeng Zhao, and Kaixiang Yang. 2020. A note on graph-based nearest neighbor search. *arXiv preprint arXiv:2012.11083* (2020).

[71] 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.

[72] Mengzhao Wang, Xiangyu Ke, Xiaoliang Xu, Lu Chen, Yunjun Gao, Pinpin Huang, and Runkai Zhu. 2024. Must: An effective and scalable framework for multimodal search of target modality. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 4747–4759.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] Yi Wu, Edward Y Chang, Kevin Chen-Chuan Chang, and John R Smith. 2004. Optimal multimodal fusion for multimedia data analysis. In *Proceedings of the 12th annual ACM international conference on Multimedia*. 572–579.

[79] Xiaoliang Xu, Mengzhao Wang, Yuxiang Wang, and Dingcheng Ma. 2021. Two-stage routing with optimized guided search and greedy algorithm on proximity graph. *Knowledge-Based Systems* 229 (2021), 107305.

[80] 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.

[81] Ming Yang, Yuzheng Cai, and Weiguo Zheng. 2024. CSPG: Crossing Sparse Proximity Graphs for Approximate Nearest Neighbor Search. *Advances in Neural Information Processing Systems* 37 (2024), 103076–103100.

[82] Shuo Yang, Jiadong Xie, Yingfan Liu, Jeffrey Xu Yu, Xiyue Gao, Qianru Wang, Yanguo Peng, and Jiangtao Cui. 2025. Revisiting the Index Construction of Proximity Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 18, 6 (Aug. 2025), 1825–1838.

[83] Ziqi Yin, Jianyang Gao, Pasquale Balsebre, Gao Cong, and Cheng Long. 2025. DEG: Efficient Hybrid Vector Search Using the Dynamic Edge Navigation Graph. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–28.

[84] Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. 2024. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. *arXiv preprint arXiv:2410.10594* (2024).

[85] Yuanhang Yu, Dong Wen, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2022. GPU-accelerated proximity graph approximate nearest Neighbor search and construction. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 552–564.

[86] Qiang Yue, Xiaoliang Xu, Yuxiang Wang, Yikun Tao, and Xuliyuan Luo. 2024. Routing-Guided Learned Product Quantization for Graph-Based Approximate Nearest Neighbor Search. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 4870–4883.

[87] Huayi Zhang, Lei Cao, Yizhou Yan, Samuel Madden, and Elke A Rundensteiner. 2020. Continuously adaptive similarity search. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2601–2616.

[88] Haoyu Zhang, Jun Liu, Zhenhua Zhu, Shulin Zeng, Maojia Sheng, Tao Yang, Guohao Dai, and Yu Wang. 2024. Efficient and effective retrieval of dense-sparse hybrid vectors using graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2410.20381* (2024).

[89] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, et al. 2023. {VBASE}: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. 377–395.

[90] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite quantization for approximate nearest neighbor search. In *International Conference on Machine Learning*. PMLR, 838–846.

[91] Weitai Zhang, Simran Naagar, Zhongyi Ye, Peiwang Tang, Xinyuan Zhou, Junhua Liu, and Lirong Dai. 2025. Bridging Modality Gap with Large Speech and Language Models for End-to-End Speech-to-Text Translation. In *ICASSP 2025 -*

2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 1–5.

[92] 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.

[93] Xiaoyao Zhong, Jiabao Jin, Peng Cheng, Mingyu Yang, Lei Chen, Haoyang Li, Zhitao Shen, Xuemin Lin, Heng Tao Shen, and Jingkuan Song. 2025. EnhanceGraph: A Continuously Enhanced Graph-based Index for High-dimensional Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2506.13144* (2025).

[94] Xiaoyao Zhong, Haotian Li, Jiabao Jin, Mingyu Yang, Deming Chu, Xiangyu Wang, Zhitao Shen, Wei Jia, George Gu, Yi Xie, Xuemin Lin, Heng Tao Shen, Jingkuan Song, and Peng Cheng. 2025. VSAG: An Optimized Search Framework for Graph-Based Approximate Nearest Neighbor Search. 18, 12 (Sept. 2025), 5017–5030.

[95] Dantong Zhu and Minjia Zhang. 2021. Understanding and Generalizing Monotonic Proximity Graphs for Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2107.13052* (2021).

Received July 2025; revised October 2025; accepted November 2025