BETTER NEAREST NEIGHBOR GRAPH INDICES VIA (EFFICIENT) LLM-GUIDED PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based approximate nearest neighbor search (ANNS) is widely used for large-scale semantic search. Its indices are constructed primarily based on *geometric relationships* among embeddings of an input dataset (e.g., documents or images), rather than explicitly optimizing for *semantic relevance*. However, when using these indices for downstream query retrieval, performance is evaluated based on the semantic relevance of the retrieved results to the query. This creates a fundamental "geometry–semantic" mismatch between how the indices are *constructed* and how their retrieval results are *evaluated*. While existing LLM-based reranking methods can partially mitigate this mismatch at query time, they leave this underlying structural problem in the graph unaddressed. We therefore propose ***LLM-Guided Graph Pruning*** (LGP), a general framework that addresses this mismatch directly by leveraging LLM reasoning to refine an existing ANN graph index itself. LGP identifies "low-value" neighbors of nodes and replaces them with LLM-selected alternatives that provide useful semantic information while retaining desired geometric structures of the original graph, including sparsity and efficient navigability. Experiments on representative semantic retrieval benchmarks show that LGP consistently improves end-to-end retrieval performance over both vanilla greedy graph search and LLM-based reranking across widely used graph-based ANN indices such as DiskANN and HNSW.

1 INTRODUCTION

Graph-based Approximate Nearest Neighbor Search (ANNS) is a popular and foundational component of modern retrieval systems (Douze et al., 2026), including retrieval-augmented search (Arslan et al., 2024), and is an active area of research in the machine learning, algorithms, and database communities.[1] At a high level, this family of algorithms constructs a sparse, navigable graph (i.e., one that enables greedy search to efficiently reach points close to a query) over an input dataset (e.g., documents or images) during preprocessing, based on the geometric proximity of their embeddings Malkov & Yashunin (2020); Jayaram Subramanya et al. (2019). When a query arrives, the prevailing paradigm is to perform a (highly efficient) greedy search over this graph based on embedding similarity, followed by a reranking step over the retrieved candidates. This second step often utilizes a Large Language Model (LLM) or a more accurate similarity computation and is much more computationally expensive than the greedy search (Sun et al., 2023; Pradeep et al., 2023; Qin et al., 2024; Weller et al., 2025; Shao et al., 2025). However, this reranking step is considered crucial because embeddings alone often fail to fully capture the complex semantic relationships between queries and the input dataset, and thus, a more "powerful" secondary semantic evaluator is necessary for accurate retrieval (Chan et al., 2024; Yao et al., 2025).

From this discussion, it is evident that there is a fundamental mismatch between *how graph indices are built and how they are ultimately deployed*. They are built using embedding-space geometry due to efficiency constraints, but actual query search requires refining the greedy search using LLMs or expensive similarity measures for the sake of accuracy. This "geometry–semantic" mismatch is especially problematic under a limited search budget, where graph search can visit only a small fraction of the input dataset (Yoon et al., 2025). Within this budget, the graph search may favor geometrically close but semantically less relevant documents while missing relevant ones. Even

[1] See, e.g., the recent VecDB Workshop at VLDB26 (`https://vecdb-ws.github.io/vldb2026/`).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

though "post-hoc" methods such as LLM-based reranking can partially mitigate this problem, their effectiveness remains constrained by the recall of relevant documents in the initial candidate set. If a relevant document is never reached initially, even a powerful LLM reranker cannot recover it afterward. Although increasing the search budget can reduce this "candidate bottleneck", doing so incurs prohibitive query-time computation.

Another possibility is to construct the graph directly according to semantic relationships among documents. However, evaluating these relationships across a large dataset using expensive measures, such as LLM-based reasoning, would again incur substantial computational costs. More importantly, a graph created based on purely semantic relationships may no longer be compatible with the efficient geometric computations used by greedy search at query time. Then navigating such a graph using greedy search may therefore reduce retrieval accuracy, whereas using stronger semantic measures to guide navigation would substantially increase query-time latency and cost. **Our paper focuses on addressing this mismatch between geometry-driven graph construction and downstream semantic retrieval under a constrained search budget**.

Our Results. To understand how pronounced this mismatch is in practice and whether geometry and semantics can be better balanced in a single graph, we first conduct a controlled and idealized empirical analysis. Our analysis shows that graph construction based on embeddings alone leaves substantial semantic search potential untapped: changing only a single outgoing edge to a local alternative (from a two-hop neighborhood) can substantially improve ground-truth recall under the same search budget, while preserving useful geometric properties. These findings motivate performing targeted, "local" edge updates on top of an existing graph index to better reconfigure its edge connections with semantic relationships, while still preserving efficient navigation.

Based on these insights, we propose ***L**LM-**G**uided Graph **P**runing* (LGP), a general framework that moves beyond only using reranking at query time and uses LLM-guided graph refinement *during the index construction itself*. LGP operates on top of existing embedding-based graph indices and makes local edge changes. At a high level, LGP constructs a bounded candidate pool from local two-hop neighborhoods, identifies structurally redundant neighbors that provide little additional reachability capabilities, and judiciously uses LLM reasoning to select candidates that add distinct semantic value beyond the current neighborhood. These new candidates replace the identified "low-value" neighbors. Crucially, LGP is designed to preserve the geometric structure that supports efficient graph search (i.e., navigability) and leaves the query-time greedy search procedure unchanged, while making local neighborhoods more semantically informative without sacrificing sparsity or introducing additional LLM inference at query time. Furthermore, since it modifies only the graph index, it can be seamlessly integrated into existing retrieve-then-rerank pipelines.

To show its versatility, we instantiate LGP on two widely used graph-based ANN indices, DiskANN (Jayaram Subramanya et al., 2019) and HNSW (Malkov & Yashunin, 2020), although the framework can be applied more broadly to other graph-based ANN methods. Extensive experiments demonstrate that LGP addresses limitations of prior embedding-based graph constructions and improves end-to-end retrieval performance. Specifically, relative to the vanilla baseline, LGP improves average greedy-search NDCG@10 by **24.4%** on the textual BRIGHT and **11.7%** on the multimodal M-BEIR, with the improvements reaching **26.0%** and **13.2%** after reranking, respectively. Furthermore, LGP substantially improves search efficiency, achieving comparable retrieval quality with up to **50% fewer distance computations**. To summarize, our main contributions are:

1. We identify and formulate a fundamental *geometry–semantic mismatch* in graph-based ANN retrieval, where graph indices are constructed based on embedding-space geometry, while downstream retrieval is ultimately evaluated by semantic relevance.
2. Through controlled one-edge replacements, we empirically show that geometrically comparable edge choices give substantially different ground-truth recall during search, revealing the potential for better trade-offs between geometric navigation and semantic retrieval in a single graph.
3. Guided by these findings, we propose LGP, an offline graph-refinement framework that selectively replaces structurally low-value neighbors with LLM-selected alternatives that provide more useful semantic connections, while leaving the query-time search procedure unchanged.
4. We instantiate LGP on two widely used graph-based ANN indices, DiskANN and HNSW, and demonstrate consistent retrieval improvements across semantic retrieval benchmarks while preserving graph sparsity and achieving comparable retrieval quality with up to 50% fewer distance computations.

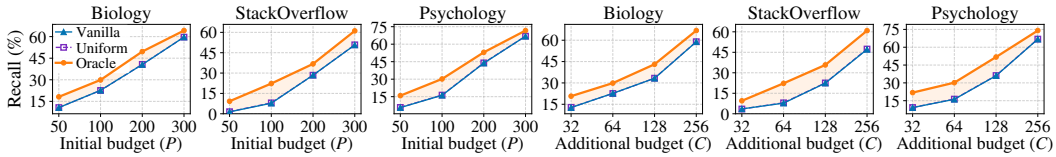

Figure 1: Alternate one-edge replacements reveal a substantial geometry–semantic mismatch and potential for improvement. Across both additional budgets C and initial budgets P, *Oracle* consistently achieves higher ground-truth recall than *Vanilla*, while *Uniform* is close to *Vanilla*.

2 PRELIMINARIES

Graph-based approximate nearest neighbor search (ANNS) methods build sparse navigable graphs over dense vector representations to support efficient retrieval (Jayaram Subramanya et al., 2019; Malkov & Yashunin, 2020). Here, navigability refers to the ability of greedy graph search to efficiently reach nodes that are close to a query in embedding space. Given a corpus $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$, each document $d_i \in \mathcal{D}$ is encoded by a bi-encoder into an embedding vector $\boldsymbol{e}(d_i)$. A graph-based ANN index is represented as a directed graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a document d_i, and $(v_i, v_j) \in E$ denotes a directed edge from v_i to v_j. We use $N(i) = \{v_j : (v_i, v_j) \in E\}$ to denote the outgoing neighborhood of v_i with degree bound R. We use $D(\cdot, \cdot)$ to denote the embedding-space distance (e.g., cosine distance) between two nodes or between a query and a node.

Although graph-based ANN indices differ in construction details, most follow a common pattern. For each node v_i, the index first retrieves a candidate set U_i through graph traversal or local search, then applies an index-specific neighbor-selection rule to form the outgoing neighborhood $N(i)$ under degree limit R. For example, DiskANN uses geometric pruning on a flat graph, while HNSW combines hierarchical search with heuristic neighbor selection.[2] In both cases, construction is based on embedding-space geometry rather than explicit semantic relationships among documents.

At query time, a query q is encoded as $\boldsymbol{e}(q)$ and greedy search is performed over G. Starting from an entry node, the search repeatedly expands promising frontier nodes according to their embedding-space distance to q and evaluates their outgoing neighbors. We count each evaluation of the embedding-space distance between q and a previously unseen node as one query–node distance computation. Under a finite computation budget, the search explores only part of the graph, producing candidates that are either returned directly or passed to a downstream reranker.

Other Related Work on LLM-Augmented and Query-Time Retrieval Algorithms. A separate line of work improves retrieval quality by adding stronger semantic reasoning at inference time, including query rewriting, reasoning-aware retrievers, and reranking methods (Ma et al., 2023; Chan et al., 2024; Shao et al., 2025; Zhang et al., 2025; Liu et al., 2025). Other recent methods modify graph search itself: bi-metric search uses a cheap proxy metric for indexing while adapting query-time search toward a more expensive target metric (Xu et al., 2026), and RGS alters graph traversal to decide which candidates should be evaluated by a stronger reranker under a limited budget (Xu & Chen, 2026). These approaches are complementary to ours: they primarily improve how a fixed graph is queried online, whereas our method improves the graph structure itself offline. Thus, our refined graph remains compatible with standard greedy search and a broad class of query-time retrieval algorithms that incorporate stronger semantic reasoning. See extended related work in Section F.

3 MOTIVATION: THE GEOMETRY–SEMANTIC MISMATCH

Embedding-based graph indices connect each document to a small set of geometrically close neighbors, which may not align with its underlying semantic relationships. This creates a mismatch because geometrically preferred edges are not necessarily the most useful for semantic retrieval. It is therefore important to empirically investigate and quantify how pronounced this mismatch is in practice. Furthermore, sparse ANN indices generally allow multiple edge choices that are similar under the geometric constraints used to build the index, so we naturally ask: can some of these alternative edge choices be a better fit with semantic relationships, and if so, to what extent? To this

[2]Detailed algorithms for DiskANN and HNSW are provided in Section D.

end, we design a controlled setup where we modify only a single edge of the vanilla index using alternatives from a bounded candidate set under the same geometric constraints. This tests whether vanilla geometric construction leaves room for improving semantic retrieval without violating existing geometric constraints, and whether geometry and semantics can be balanced within a single graph.

One-Edge Replacement. Specifically, for each query q and initial search budget P, we run greedy search on the Vanilla DiskANN graph until at least P unique embedding-space distance computations have been performed, and save the search immediately before the next frontier node v_i is expanded. Let $N(i)$ denote the outgoing neighbors of v_i, and let $N_2(i)$ denote its two-hop non-neighbors: $N_2(i) = \left(\bigcup_{v_j \in N(i)} N(j)\right) \setminus (N(i) \cup \{v_i\})$. Starting from this same point, for every $z \in N(i)$ and $c \in N_2(i)$, we construct a one-edge variant of the graph by replacing (v_i, z) with (v_i, c) and continue the greedy search for an additional budget of C query–document distance computations. By varying both P and C, we examine whether a single local edge change can improve relevant-document discovery across different stages of the search and different search budgets.

Measuring the Geometry–Semantic Gap. For each query q and each pair of initial and additional search budgets (P, C), we evaluate the vanilla graph and every one-edge variant by the fraction of ground-truth relevant documents, $\mathrm{GT}(q)$, reached after the additional C distance computations. Specifically, for query q and graph G , let $\mathcal{D}_G^{\mathrm{seen}}(q, P, C)$ denote the documents evaluated during the initial P distance computations together with those evaluated during the additional C computations. We measure ground-truth recall as

$$\frac{|\mathcal{D}_G^{\mathrm{seen}}(q, P, C) \cap \mathrm{GT}(q)|}{|\mathrm{GT}(q)|}. \tag{1}$$

To preserve the geometric constraints of the original graph, we keep only one-edge variants whose recall of the exact top-10 embedding-space neighbors of q is no worse than that of the vanilla graph under the same (P, C) budgets. For each query and budget setting, we define the variant with the highest ground-truth recall as *Oracle* and compare it with *Vanilla* to quantify the potential improvement under the same geometric constraints. As a control, we report the average recall across all such one-edge variants as the *Uniform* baseline.

Geometry Leaves Semantic Retrieval Potential Untapped. We conduct this analysis on three BRIGHT datasets, Biology, StackOverflow, and Psychology, using DiskANN as the underlying graph index (SU et al., 2025). As shown in Figure 1, *Oracle* consistently outperforms *Vanilla* in ground-truth recall across varying C at fixed $P = 100$ and varying P at fixed $C = 64$. This consistent gap confirms that geometrically comparable edge choices can differ substantially in their semantic retrieval performance, directly demonstrating a substantial geometry–semantic mismatch in practice. More importantly, these results indicate that graph construction based on embeddings alone leaves substantial potential for improving semantic relevance while satisfying the same geometric constraints. Furthermore, the near-zero improvement of *Uniform* shows that arbitrary one-edge replacements do not perform well, and that realizing this potential requires selecting semantically useful alternatives. These findings suggest that geometry and semantics can be combined and "harmonized" within a single graph and motivate our method, LGP, which selectively prunes local edges on top of the existing geometric index to incorporate more semantically useful connections.

4 Our Framework: LLM-Guided Graph Pruning

To address the geometry–semantic mismatch while preserving efficient graph navigation, we introduce ***L**LM-Guided **G**raph **P**runing* (LGP), a general framework for refining an existing ANN index by making its local neighborhoods better aligned with semantic relationships (Algorithm 1). Rather than replacing the original indexing algorithm, LGP operates on top of the existing graph index (e.g., DiskANN and HNSW), identifies neighbors that provide little additional reachability beyond the rest of the local neighborhood, and replaces them with LLM-selected candidates that can help the search reach more relevant documents without violating the geometric constraints of the original index. For each node in the index, our algorithm proceeds in three main steps. First, it constructs a small set of local replacement candidates using directed two-hop non-neighbors. It then uses efficient to compute "structural" signals, including additional two-hop reachability and neighborhood redundancy, to shortlist promising candidates. From this shortlist, LLM reasoning is used to select semantically useful alternatives. Finally, the selected candidates are added to the neighborhood

Algorithm 1 LLM-Guided Graph Pruning (LGP)

Input ANN graph G, current node v_i, neighborhood $N(i)$, and index-specific candidate set U_i
Parameters Degree cap R, shortlist budget M, and edit budget B
▷ /* Form a local replacement set */
$U_i \leftarrow U_i$ together with directed two-hop non-neighbors of v_i
▷ /* Shortlist local replacement candidates */
$U_i \leftarrow$ shortlist up to M candidates based on additional two-hop reachability and redundancy with $N(i)$
▷ /* Select semantically useful candidates with an LLM */
$\mathcal{A}_i \leftarrow \text{LLMSELECT}(v_i, N(i), U_i, B)$
▷ /* Local edge pruning and replacement */
for each $v_c \in \mathcal{A}_i$ **do**
    Identify an existing neighbor v_w with little additional reachability and high redundancy
    Update $N(i)$ by adding v_c and, when necessary, removing v_w to satisfy the degree constraint

by replacing existing neighbors with little additional reachability and high redundancy. A detailed version of the algorithm is provided in Algorithm 8 in the appendix.

Efficient Local Candidate Set Construction. Given an ANN index G, for each node v_i, LGP considers only a small set of local replacement candidates rather than all nodes in the graph. Since most graph-based ANN methods already produce a candidate set of geometrically close neighbors when selecting or updating the outgoing neighborhood $N(i)$ of v_i, LGP reuses this set, denoted by U_i, and augments it with directed two-hop non-neighbors, i.e., nodes reachable through a current neighbor but not already directly connected to v_i. This provides a broader set of local alternatives from which the LLM can identify semantically useful replacements, while keeping candidate construction computationally inexpensive. To further control the cost and context length of LLM selection, LGP prunes the augmented set to at most M candidates using only graph connectivity and embedding distances, without invoking the LLM. The pruning favors candidates that provide greater additional two-hop reachability and are less redundant with the current neighborhood (lines 7–15 in Algorithm 8).

LLM-Guided Semantic Selection. From the shortlisted set U_i, LGP uses an LLM to identify replacement candidates that add useful semantic information beyond what is already represented in the current neighborhood. The LLM is given the document represented by v_i, a small set of its current neighbors as context, and the documents in the shortlisted set U_i. It selects up to B candidates that extend rather than repeat the information already available through the current neighborhood, for example, by providing missing supporting context or connecting v_i to a related concept that is not well represented by its existing neighbors. Here, B is a small per-node edit budget that limits the number of edge changes (e.g., $B = 2$). Prompts for LLMSELECT are provided in Section C.

Local Edge Replacement. After LLM-guided selection, LGP applies a lightweight structural check to keep candidates that provide sufficient additional reachability. Such a candidate is added directly when the neighborhood is below the degree limit. Otherwise, LGP replaces an existing neighbor that provides little additional two-hop reachability beyond the other neighbors and is highly redundant with the current neighborhood. This allows LGP to introduce more semantically useful connections while preserving the sparsity and local geometric structure of the original index.

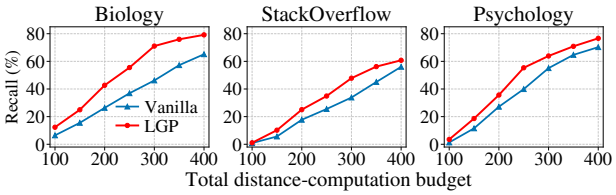

Figure 2: LGP-refined DiskANN consistently improves recall over Vanilla across distance-computation budgets.

Instantiation for Representative ANN Indices. LGP is general and can be instantiated on different graph-based ANN indices, including DiskANN and HNSW. In DiskANN, LGP is applied immediately after RobustPrune during the second construction pass, taking the nodes visited by GreedySearch as U_i. In HNSW, LGP refines only the base layer of the constructed index, taking the base-layer nodes visited by hierarchical search as U_i. Both instantiations follow the shared pruning procedure in Algorithm 1, with detailed algorithms provided in Section E.1.

LGP Realizes the Semantic Improvement Potential. To examine whether the improvement potential identified by the controlled one-edge analysis in Section 3 can be realized by LGP, we

Table 1: BRIGHT main results on DiskANN at search width 20 using Diver embeddings. Each entry reports Vanilla / LGP. Green subscripts denote relative improvements over Vanilla.

	Greedy search		Rerank	
Dataset	NDCG@10	Recall@10	NDCG@10	Recall@10
Biology	24.69 / **37.06**$_{+50.1\%}$	28.66 / **44.63**$_{+55.7\%}$	28.30 / **41.57**$_{+46.9\%}$	30.71 / **46.19**$_{+50.4\%}$
Earth Science	20.60 / **27.92**$_{+35.5\%}$	22.70 / **30.59**$_{+34.8\%}$	23.29 / **30.06**$_{+29.1\%}$	25.51 / **33.03**$_{+29.5\%}$
Economics	19.56 / **22.35**$_{+14.3\%}$	20.77 / **25.61**$_{+23.3\%}$	19.98 / **25.42**$_{+27.2\%}$	21.92 / **28.84**$_{+31.6\%}$
Psychology	28.31 / **34.76**$_{+22.8\%}$	34.51 / **41.61**$_{+20.6\%}$	32.78 / **41.46**$_{+26.5\%}$	34.40 / **44.10**$_{+28.2\%}$
Robotics	19.48 / **21.68**$_{+11.3\%}$	24.83 / **27.12**$_{+9.2\%}$	25.67 / **28.63**$_{+11.5\%}$	28.61 / **31.16**$_{+8.9\%}$
StackOverflow	10.82 / **17.36**$_{+60.4\%}$	13.87 / **21.37**$_{+54.1\%}$	11.96 / **19.61**$_{+64.0\%}$	14.66 / **23.73**$_{+61.9\%}$
Sustainable Living	18.09 / **24.33**$_{+34.5\%}$	22.12 / **29.76**$_{+34.5\%}$	20.20 / **27.43**$_{+35.8\%}$	22.04 / **30.82**$_{+39.8\%}$
Pony	17.08 / **17.80**$_{+4.2\%}$	9.06 / **9.40**$_{+3.8\%}$	19.54 / **20.07**$_{+2.7\%}$	**9.73** / 9.00
LeetCode	3.20 / **6.20**$_{+93.8\%}$	4.54 / **9.51**$_{+109.5\%}$	2.97 / **6.37**$_{+114.5\%}$	4.27 / **10.07**$_{+135.8\%}$
AoPS	3.13 / **3.36**$_{+7.3\%}$	4.58 / **5.03**$_{+9.8\%}$	3.12 / **3.23**$_{+3.5\%}$	4.69 / **4.79**$_{+2.1\%}$
TheoremQA-Theorems	34.70 / **36.35**$_{+4.8\%}$	46.26 / **47.57**$_{+2.8\%}$	33.57 / **36.44**$_{+8.5\%}$	45.60 / **48.89**$_{+7.2\%}$
TheoremQA-Questions	5.73 / **6.45**$_{+12.6\%}$	8.24 / **8.36**$_{+1.5\%}$	5.63 / **5.68**$_{+0.9\%}$	9.24 / **9.49**$_{+2.7\%}$
Average	17.12 / **21.30**$_{+24.4\%}$	20.01 / **25.05**$_{+25.2\%}$	18.92 / **23.83**$_{+26.0\%}$	20.95 / **26.68**$_{+27.4\%}$

compare the fixed LGP-refined DiskANN index directly against the original vanilla index across total distance-computation budgets. As shown in Figure 2, LGP consistently achieves higher ground-truth recall than *Vanilla* across all tested budgets and datasets. These results show that the potential revealed by the one-edge analysis translates into consistent retrieval gains with LGP, demonstrating that semantic relevance and geometric constraints can be better balanced within a single graph.

5 Empirical Results

We comprehensively evaluate LGP on textual and multimodal benchmarks using DiskANN and HNSW. Across settings, LGP consistently improves greedy-search retrieval, and these gains persist after downstream reranking. The improvements are most pronounced at narrow search widths.

Benchmarks and Models. We use BRIGHT (SU et al., 2025) for reasoning-intensive text retrieval and M-BEIR (Wei et al., 2024) for multimodal retrieval. BRIGHT contains 12 datasets spanning diverse domains, while M-BEIR contains 16 tasks covering text, image, and image–text retrieval. We evaluate all BRIGHT queries and, following Xu & Chen (2026), uniformly sample 100 queries per M-BEIR task using the same samples across all experiments. For BRIGHT, we use Diver-Retriever-4B-1020 (Sun et al., 2026) and Qwen3-Embedding-4B as embedding models. For M-BEIR, following Xu & Chen (2026), we use CLIP (ViT-B/32) (Radford et al., 2021) to encode all 16 task subsets. For LLM-guided semantic selection and downstream reranking, we use Qwen3-32B (Team, 2025) on BRIGHT and Qwen3-VL-30B-A3B-Instruct (Bai et al., 2025) on M-BEIR, both in non-reasoning mode. Detailed benchmark and model configurations are provided in Tables 8 to 10.

Index Configurations. Our primary ANN index is DiskANN, with a maximum out-degree of 16, a construction search width of 125, and a pruning parameter of 1.2. We additionally evaluate HNSW with connectivity 16 and construction search width 200. We use an edit budget of $B = 2$ for both benchmarks, with a shortlist size of $M = 24$ on BRIGHT and $M = 12$ on M-BEIR.

Retrieval Configurations and Metrics. At query time, we evaluate greedy graph search and downstream reranking. We use *search width*, the standard query-time parameter for controlling search effort in graph-based ANN indices, corresponding to L_{search} in DiskANN and ef_{search} in HNSW. With graph degree bounded by R, the query-time search cost grows roughly with search width $\times R$ in terms of distance computations. For DiskANN, we vary $L_{\text{search}} \in \{10, 20, 50, 100, 200\}$; for HNSW, we vary $ef_{\text{search}} \in \{10, 12, 14, 16, 18, 20, 50\}$, using 20 by default. Greedy search returns candidate documents ordered by embedding distance. We either evaluate the top k directly or rerank the same candidate set before evaluating the resulting top k. We report NDCG@k and document-level Recall@k for $k \in \{5, 10\}$ on a 0–100 scale. Recall@k is the fraction of all ground-truth documents that appear in the top k, averaged over queries. When a query has more than k ground-truth documents, the maximum possible Recall@k is below 100. Full details are provided in Section A.

Main Results. As shown in Table 1, LGP improves both greedy-search and reranked NDCG@10 across all 12 BRIGHT datasets at the search width of 20. Under greedy search, LGP increases average NDCG@10 from 17.12 to 21.30 (**+24.4%**) and Recall@10 from 20.01 to 25.05 (**+25.2%**). Biology

Table 2: M-BEIR results on DiskANN at search width 20 using CLIP. Each entry reports Vanilla/LGP.

	Greedy search		Rerank	
Dataset (Task)	NDCG@10	Recall@10	NDCG@10	Recall@10
VisualNews (0)	**16.41** / 15.45	**21.00** / 20.00	15.23 / **15.97**$_{+4.9\%}$	21.00 / 21.00
MSCOCO (0)	32.45 / **32.88**$_{+1.3\%}$	51.00 / **52.00**$_{+2.0\%}$	45.09 / **46.09**$_{+2.2\%}$	56.00 / **57.00**$_{+1.8\%}$
Fashion200K (0)	1.80 / **1.82**$_{+1.1\%}$	2.63 / 2.63	**4.53** / 3.56	**4.97** / 3.97
WebQA (1)	5.99 / **6.61**$_{+10.4\%}$	7.50 / **8.00**$_{+6.7\%}$	8.29 / **8.93**$_{+7.7\%}$	8.00 / **9.00**$_{+12.5\%}$
EDIS (2)	6.76 / **11.29**$_{+67.0\%}$	9.07 / **14.74**$_{+62.5\%}$	11.02 / **15.67**$_{+42.2\%}$	12.80 / **18.71**$_{+46.2\%}$
WebQA (2)	15.15 / **22.97**$_{+51.6\%}$	20.50 / **29.50**$_{+43.9\%}$	20.25 / **29.18**$_{+44.1\%}$	23.50 / **33.50**$_{+42.6\%}$
VisualNews (3)	3.87 / **7.04**$_{+81.9\%}$	9.00 / **14.00**$_{+55.6\%}$	5.25 / **10.29**$_{+96.0\%}$	11.00 / **19.00**$_{+72.7\%}$
MSCOCO (3)	42.42 / **43.00**$_{+1.4\%}$	42.60 / **42.80**$_{+0.5\%}$	52.63 / **53.92**$_{+2.5\%}$	49.20 / **49.80**$_{+1.2\%}$
Fashion200K (3)	**2.93** / 2.86	6.00 / 6.00	3.97 / **4.00**$_{+0.8\%}$	6.00 / 6.00
NIGHTS (4)	28.73 / 28.73	51.00 / 51.00	41.99 / **42.51**$_{+1.2\%}$	61.00 / 61.00
OVEN (6)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
InfoSeek (6)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
CIRR (7)	6.47 / **7.83**$_{+21.0\%}$	13.00 / **16.00**$_{+23.1\%}$	10.99 / **12.34**$_{+12.3\%}$	13.00 / **15.00**$_{+15.4\%}$
FashionIQ (7)	0.67 / **1.12**$_{+67.2\%}$	2.00 / **3.00**$_{+50.0\%}$	1.61 / **3.66**$_{+127.3\%}$	4.00 / **6.00**$_{+50.0\%}$
OVEN (8)	0.50 / **1.35**$_{+170.0\%}$	0.75 / **2.08**$_{+177.3\%}$	1.16 / **3.09**$_{+166.4\%}$	0.75 / **2.08**$_{+177.3\%}$
InfoSeek (8)	0.48 / **0.93**$_{+93.8\%}$	1.00 / **1.58**$_{+58.0\%}$	0.45 / **2.53**$_{+462.2\%}$	1.00 / **2.31**$_{+131.0\%}$
Average	10.29 / **11.49**$_{+11.7\%}$	14.82 / **16.46**$_{+11.1\%}$	13.90 / **15.73**$_{+13.2\%}$	17.01 / **19.02**$_{+11.8\%}$

Table 3: Search-width sweeps on BRIGHT and M-BEIR with Diver and CLIP embeddings, respectively. Green subscripts denote relative improvements over the Vanilla baseline. Gray values denote the average exact-kNN NDCG@10 / Recall@10 for each benchmark.

Benchmark (Exact kNN ref.)	Search width	Greedy search				Rerank			
		NDCG@10		Recall@10		NDCG@10		Recall@10	
		Vanilla	LGP	Vanilla	LGP	Vanilla	LGP	Vanilla	LGP
BRIGHT (28.50 / 34.18)	10	11.90	**15.74**$_{+32.3\%}$	13.53	**18.27**$_{+35.0\%}$	13.01	**17.28**$_{+32.8\%}$	13.53	**18.27**$_{+35.0\%}$
	20	17.12	**21.30**$_{+24.4\%}$	20.01	**25.05**$_{+25.2\%}$	18.92	**23.83**$_{+26.0\%}$	20.95	**26.68**$_{+27.4\%}$
	50	23.09	**25.03**$_{+8.4\%}$	27.43	**29.51**$_{+7.6\%}$	26.31	**28.47**$_{+8.2\%}$	29.95	**32.56**$_{+8.7\%}$
	100	26.10	**26.87**$_{+3.0\%}$	31.05	**31.73**$_{+2.2\%}$	30.24	**31.28**$_{+3.4\%}$	34.94	**35.61**$_{+1.9\%}$
	200	26.94	**27.42**$_{+1.8\%}$	32.10	**32.47**$_{+1.2\%}$	31.64	**32.17**$_{+1.7\%}$	37.13	**37.66**$_{+1.4\%}$
M-BEIR (16.16 / 22.87)	10	9.08	**9.47**$_{+4.3\%}$	13.07	**13.69**$_{+4.7\%}$	11.00	**11.49**$_{+4.5\%}$	13.07	**13.69**$_{+4.7\%}$
	20	10.29	**11.49**$_{+11.7\%}$	14.82	**16.46**$_{+11.1\%}$	13.90	**15.73**$_{+13.2\%}$	17.01	**19.02**$_{+11.8\%}$
	50	12.99	**13.68**$_{+5.3\%}$	18.28	**19.19**$_{+5.0\%}$	19.00	**20.17**$_{+6.2\%}$	22.81	**24.01**$_{+5.3\%}$
	100	14.00	**14.66**$_{+4.7\%}$	19.60	**20.47**$_{+4.4\%}$	20.93	**22.00**$_{+5.1\%}$	25.66	**27.08**$_{+5.5\%}$
	200	14.82	**15.23**$_{+2.8\%}$	20.62	**21.25**$_{+3.1\%}$	22.20	**22.68**$_{+2.2\%}$	27.30	**27.77**$_{+1.7\%}$

shows the largest absolute NDCG@10 gain, increasing from 24.69 to 37.06 (**+50.1%**). These gains persist after LLM reranking, where average NDCG@10 increases from 18.92 to 23.83 (**+26.0%**) and Recall@10 from 20.95 to 26.68 (**+27.4%**). As shown in Table 2, across 16 M-BEIR tasks, LGP improves average NDCG@10 by **11.7%** under greedy search and **13.2%** after VLM reranking, with reranked Recall@10 improving by **11.8%**. WebQA Task 2 shows the largest NDCG@10 gains, improving by **51.6%** under greedy search and **44.1%** after reranking. These results show that LGP refines the index with connections that better reflect semantic relationships, improving graph-search retrieval and downstream reranking across both text and multimodal settings.

Search Width. To assess how the benefits of LGP vary with query-time search width, we evaluate performance across different widths. Table 3 shows that LGP achieves its largest gains on BRIGHT at narrower widths: at width 10, average NDCG@10 improves by **32.3%** under greedy search and **32.8%** after LLM reranking, while remaining consistently better than Vanilla as the search width increases. The per-dataset curves in Figure 3 exhibit the same pattern. On M-BEIR, LGP improves average NDCG@10 under both greedy search and reranking at every tested width. The task-level results in Figure 8 (Appendix) further show consistent gains across individual M-BEIR tasks. Furthermore, compared with Vanilla, LGP approaches the exact kNN faster as the search width increases and remains consistently closer to exact retrieval on both benchmarks. These results show that LGP provides its strongest gains at narrower search widths while remaining consistently beneficial as the search width increases. Dataset-wise sweeps are provided in Section B.1.

Explicit Distance-Computation Budgets. Search width is the standard control for query-time search effort, while explicit distance-computation budgets provide a complementary matched-compute view. We therefore compare Vanilla and LGP under the same limit on the number of query–document distance computations. As shown in Table 4, LGP consistently outperforms Vanilla across all tested

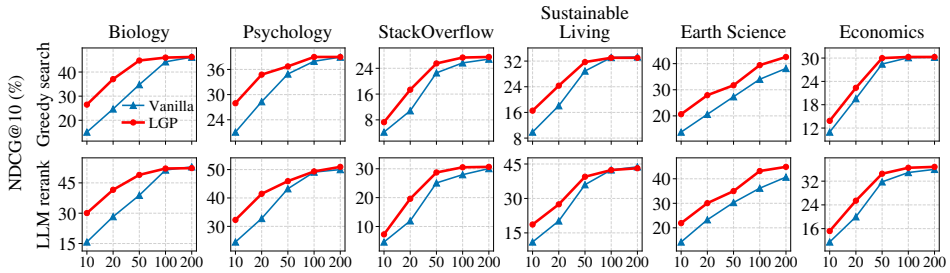

Figure 3: Search-width sweeps on 6 BRIGHT datasets using Diver embeddings. LGP consistently outperforms Vanilla under both greedy search and LLM reranking across search widths.

Table 4: Distance-computation-budget sweeps on BRIGHT using Diver embeddings.

Benchmark (Exact kNN ref.)	Computation budget	Greedy search				Rerank			
		NDCG@10		Recall@10		NDCG@10		Recall@10	
		Vanilla	LGP	Vanilla	LGP	Vanilla	LGP	Vanilla	LGP
BRIGHT (28.50 / 34.18)	100	3.96	**5.02**$_{+26.9\%}$	2.96	**4.15**$_{+40.0\%}$	4.28	**5.40**$_{+26.1\%}$	3.07	**4.12**$_{+34.3\%}$
	200	10.50	**13.98**$_{+33.2\%}$	11.62	**15.60**$_{+34.3\%}$	11.67	**15.41**$_{+32.0\%}$	12.32	**16.40**$_{+33.1\%}$
	300	16.69	**19.80**$_{+18.6\%}$	19.18	**23.19**$_{+20.9\%}$	18.70	**22.46**$_{+20.1\%}$	20.35	**24.61**$_{+21.0\%}$
	500	22.92	**24.57**$_{+7.2\%}$	27.14	**28.83**$_{+6.2\%}$	26.04	**27.79**$_{+6.7\%}$	28.70	**30.82**$_{+7.4\%}$
	800	25.89	**26.61**$_{+2.8\%}$	30.66	**31.37**$_{+2.3\%}$	29.00	**29.92**$_{+3.2\%}$	32.59	**33.54**$_{+2.9\%}$
	1600	27.11	**27.27**$_{+0.6\%}$	32.27	**32.32**$_{+0.2\%}$	30.83	**30.95**$_{+0.4\%}$	34.63	**34.71**$_{+0.2\%}$
	3200	27.65	**27.74**$_{+0.3\%}$	32.90	**32.96**$_{+0.2\%}$	31.26	**31.28**$_{+0.1\%}$	35.15	**35.18**$_{+0.1\%}$

Table 5: Cross-index evaluation of LGP on HNSW over BRIGHT datasets with Diver embeddings at search width 20. Each entry reports Vanilla / LGP.

Dataset	Greedy search				Rerank			
	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10
TheoremQA-Theorems	38.64 / **39.47**	42.39 / **43.34**	45.05 / **45.71**	54.81 / **56.12**	38.26 / **40.76**	41.85 / **43.52**	44.86 / **48.15**	55.47 / **56.12**
Biology	**42.71** / 42.19	**48.06** / 47.39	**42.13** / 41.40	**57.25** / 56.09	50.85 / **52.83**	54.67 / **55.74**	48.63 / **49.20**	**59.90** / 58.90
Pony	14.84 / **17.11**	12.83 / **14.48**	3.83 / **4.36**	6.71 / **8.04**	17.82 / **19.92**	14.80 / **16.28**	4.72 / **5.62**	7.37 / **8.78**

budgets, with the largest gains at smaller and intermediate budgets. At a budget of 200, LGP improves greedy NDCG@10 and Recall@10 by **33.2%** and **34.3%**, respectively. Moreover, on Economics from BRIGHT, LGP reaches the same greedy NDCG@10 as Vanilla using roughly 50% fewer distance computations (800 vs. 1,600). These results demonstrate that LGP improves search efficiency by achieving higher retrieval quality for the same computation and comparable quality with *substantially fewer distance evaluations*. Additional results are provided in Section B.6.

Graph-Based ANN Algorithms. To evaluate whether LGP generalizes beyond DiskANN, we additionally instantiate it on HNSW across three representative BRIGHT retrieval settings: Biology (domain-specific QA), Pony (code retrieval), and TheoremQA-Theorems (theorem retrieval). At search width 20, Table 5 shows consistent improvements after reranking across datasets. Under greedy search, LGP improves TheoremQA-Theorems and Pony, while remaining comparable to baseline on Biology. The search-width sweeps in Figure 6 (Appendix) show that these gains generally persist across HNSW search widths, with larger improvements at narrower widths. These results demonstrate that LGP generalizes across different graph-based ANN indices.

Embeddings. To evaluate robustness to the embedding model, we replace Diver with Qwen3-Embedding. As shown in Table 6, LGP improves every aggregate retrieval metric under both greedy search and LLM reranking. Average NDCG@10 improves by **14.7%** under greedy search and **14.4%** after reranking. These results show that the gains of LGP remain robust across different embedding models. Additional per-dataset results are provided in Section B.2.

Model Size and Graph Degree. We further study the effects of model size and graph degree. As shown in Table 7, using Qwen3-32B for semantic selection consistently yields stronger results than Qwen3-8B. Across graph degrees, LGP outperforms Vanilla in most settings, with consistent gains on Pony and substantial improvements on Psychology in several configurations. Notably, LGP

Table 6: Embedding robustness on BRIGHT using Qwen3-Embedding at search width 20. Results are averaged equally over all 12 datasets.

	NDCG@5		NDCG@10		Recall@5		Recall@10	
Retrieval	Vanilla	LGP	Vanilla	LGP	Vanilla	LGP	Vanilla	LGP
Greedy search	10.89	**12.33**$_{+13.2\%}$	12.04	**13.81**$_{+14.7\%}$	11.52	**13.06**$_{+13.4\%}$	15.34	**17.60**$_{+14.7\%}$
LLM rerank	15.91	**18.19**$_{+14.3\%}$	16.34	**18.70**$_{+14.4\%}$	15.31	**17.37**$_{+13.5\%}$	18.19	**20.66**$_{+13.6\%}$

Table 7: Ablations of model size (left) and graph degree (right) on Pony and Psychology using Diver embeddings at search width 20. Entries report NDCG@10.

		Greedy search		LLM reranking	
Dataset	Model	Vanilla	LGP	Vanilla	LGP
Pony	Qwen3-8B	**17.08**	17.00	19.54	**19.83**
	Qwen3-32B	17.08	**17.80**	19.54	**20.07**
Psychology	Qwen3-8B	28.31	**29.82**	32.55	**34.75**
	Qwen3-32B	28.31	**34.76**	32.69	**41.46**

		Greedy search		LLM reranking	
Dataset	R	Vanilla	LGP	Vanilla	LGP
Pony	8	5.08	**9.80**	5.08	**10.08**
	16	17.08	**17.80**	19.54	**20.07**
	32	15.87	**16.28**	18.97	**19.52**
Psychology	8	6.74	**18.52**	7.71	**22.68**
	16	28.31	**34.76**	32.69	**41.46**
	32	**37.66**	36.22	**45.17**	43.78

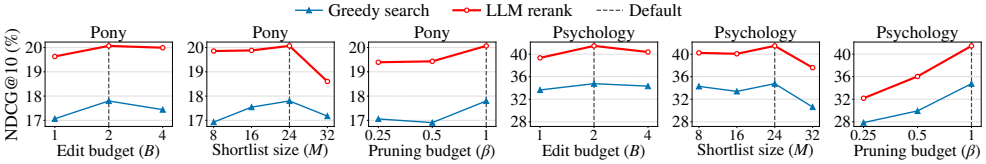

Figure 4: Hyperparameter ablation of LGP on Pony and Psychology at search width 20.

substantially improves both datasets at $R = 8$. These results show that stronger models can improve graph construction and that the benefits of LGP extend across different graph-degree settings.

Per-Node Edit Budget, Shortlist Size, and Global Pruning Budget. We further evaluate LGP under different per-node edit budgets B, shortlist sizes M, and global pruning budgets β. Here, β denotes the maximum fraction of graph nodes that undergo LLM-guided pruning, with $\beta \in \{0.25, 0.5, 1\}$ corresponding to at most 25%, 50%, and 100% of the graph, respectively. As shown in Figure 4, the moderate local settings $B = 2$ and $M = 24$ achieve the strongest greedy-search and reranked NDCG@10 among the tested choices. In contrast, increasing the global pruning budget improves performance, with $\beta = 1$ achieving the strongest results on both Pony and Psychology. These results suggest a clear trade-off: local updates should remain selective, since too few edits or candidates may limit semantic coverage, while overly aggressive editing or excessively large shortlists may hurt geometric structure or LLM selection quality. At the graph level, however, applying LLM-guided pruning to more neighborhoods yields stronger retrieval performance.

Additional Results. Additional search-width and distance-computation-budget sweeps, embedding-model, design, and hyperparameter ablations, and pruning-cost analysis are provided in Section B.

6 Conclusion and Limitations

We identify a fundamental *geometry–semantic mismatch* in graph-based ANNS: indices are built using embedding-space geometry but evaluated by semantic relevance. Our analysis shows that geometrically comparable edge choices can yield substantially different retrieval performance, motivating LGP, an offline framework that uses LLM reasoning to replace structurally low-value neighbors with semantically more useful alternatives. Across textual and multimodal benchmarks, LGP consistently improves both greedy graph search and downstream reranking on DiskANN and HNSW while preserving graph sparsity and leaving the query-time search procedure unchanged.

Our current framework is limited to graph-based ANN indices. Its effectiveness also depends on the quality of LLM/VLM semantic judgments. Extending this index-level refinement paradigm to other ANN algorithms is a promising direction for future work. More broadly, we view LGP as a first step towards identifying and addressing the geometry–semantic mismatch rather than the end of the story.

AI USE STATEMENT

We used generative AI tools only for language polishing, including improving the grammar, clarity, and readability of the manuscript. We did not use generative AI tools to develop research ideas or hypotheses, design the methodology or experiments, implement the proposed method, analyze or interpret experimental results, or formulate scientific claims or conclusions. All AI-assisted edits were reviewed by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies graph-based approximate nearest-neighbor retrieval and does not involve human-subject experiments or the collection of private user data. Our experiments use existing retrieval benchmarks and pretrained embedding and language models. Because LGP uses pretrained language or vision-language models to make semantic judgments during offline graph refinement, its behavior may inherit biases or errors present in these models and in the underlying datasets. The method is intended to improve retrieval efficiency and effectiveness, and we do not identify additional ethical risks beyond those associated with the underlying retrieval data and pretrained models.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our experiments throughout the paper and appendix. The appendix includes detailed algorithms for LGP and its DiskANN and HNSW instantiations, benchmark and preprocessing details, index-construction and query-time configurations, model choices, hyperparameters, evaluation metrics, and additional ablations. We also provide code in the supplementary material. These details are intended to make both the proposed method and the reported experimental comparisons reproducible.

REFERENCES

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. A survey on rag with llms. *Procedia Computer Science*, 246:3781–3790, 2024. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2024.09.178. URL https://www.sciencedirect.com/science/article/pii/S1877050924021860. 28th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES 2024).

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. RQ-RAG: Learning to refine queries for retrieval augmented generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tzE7VqsaJ4.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 19970–19997, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1011. URL https://aclanthology.org/2025.emnlp-main.1011/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 12(2):346–361, 2026. doi: 10.1109/TBDATA.2025.3618474.

Cong Fu and Deng Cai. Efanna : An extremely fast approximate nearest neighbor search algorithm based on knn graph, 2016. URL https://arxiv.org/abs/1609.07228.

Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proc. VLDB Endow.*, 12(5):461–474, January 2019. ISSN 2150-8097. doi: 10.14778/3303753.3303754. URL https://doi.org/10.14778/3303753.3303754.

Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters. In *Proceedings of the ACM Web Conference 2023*, WWW '23, pp. 3406–3416, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394161. doi: 10.1145/3543507.3583552. URL https://doi.org/10.1145/3543507.3583552.

Ben Harwood and Tom Drummond. Fanng: Fast approximate nearest neighbour graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5713–5722, 2016. doi: 10.1109/CVPR.2016.616.

Piotr Indyk and Haike Xu. Worst-case performance of popular approximate nearest neighbor search implementations: Guarantees and limitations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 66239–66256. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d0ac28b79816b51124fcc804b2496a36-Paper-Conference.pdf.

Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Paper.pdf.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, January 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.57. URL https://doi.org/10.1109/TPAMI.2010.57.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. Reasonrank: Empowering passage ranking with strong reasoning ability, 2025. URL https://arxiv.org/abs/2508.07050.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.322. URL https://aclanthology.org/2023.emnlp-main.322/.

Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42 (4):824–836, April 2020. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfr-embedding-2: Advanced text embedding with multi-stage training, 2024. URL https://huggingface.co/Salesforce/SFR-Embedding-2_R.

OpenAI. text-embedding-3-large, 2024. URL https://platform.openai.com/docs/models/text-embedding-3-large.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models, 2023. URL https://arxiv.org/abs/2309.15088.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.97. URL https://aclanthology.org/2024.findings-naacl.97/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=kkBCNLMbGj.

Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. Freshdiskann: A fast and accurate graph-based ann index for streaming similarity search, 2021. URL https://arxiv.org/abs/2105.09613.

Hongjin SU, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ykuc5q381b.

Duolin Sun, Meixiu Long, Dan Yang, Junjie Wang, Yecheng Luo, Yue Shen, Jian Wang, Hualei Zhou, Chunxiao Guo, Peng Wei, Jiahai Wang, and Jinjie Gu. Diver: A multi-stage approach for reasoning-intensive information retrieval, 2026. URL https://arxiv.org/abs/2508.07995.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

Zeinab Sadat Taghavi, Ali Modarressi, Hinrich Schutze, and Andreas Marfurt. With argus eyes: Assessing retrieval gaps via uncertainty scoring to detect and remedy retrieval blind spots, 2026. URL https://arxiv.org/abs/2602.09616.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. Uniir: Training and benchmarking universal multimodal information retrievers. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXVII*, pp. 387–404, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73020-7. doi: 10.1007/978-3-031-73021-4_23. URL https://doi.org/10.1007/978-3-031-73021-4_23.

Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn Lawrie, and Benjamin Van Durme. Rank1: Test-time compute for reranking in information retrieval. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Pg0PAvbhGv.

Haike Xu and Tong Chen. Beyond sequential reranking: Reranker-guided search improves reasoning intensive retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Fjw3OqdjTj.

Haike Xu, Sandeep Silwal, and Piotr Indyk. A bi-metric framework for efficient nearest neighbor search. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kBzzertldC.

Yichen Yao, Jiahe Wan, Yuxin Hong, Mengna Zhang, Junhan Yang, Zhouyu Jiang, Qing Xu, Kuan Lu, Yinghui Xu, Wei Chu, Emma Wang, and Yuan Qi. Inf-x-retriever: A pragmatic framework for reasoning-intensive dense retrieval, 2025. URL https://github.com/yaoyichen/INF-X-Retriever.

Soyoung Yoon, Gyuwan Kim, Gyu-Hwung Cho, and Seung won Hwang. Acurank: Uncertainty-aware adaptive computation for listwise reranking, 2025. URL https://arxiv.org/abs/2505.18512.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. REARANK: Reasoning re-ranking agent via reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2458–2471, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.125. URL https://aclanthology.org/2025.emnlp-main.125/.

Table 8: Dataset statistics for the BRIGHT benchmark.

Dataset	Queries	Documents
Biology	103	57,359
Earth Science	116	121,249
Economics	103	50,220
Psychology	101	52,835
Robotics	101	61,961
StackOverflow	117	107,081
Sustainable Living	108	60,792
Pony	112	7,894
LeetCode	142	413,932
AoPS	111	188,002
TheoremQA-Theorems	76	23,839
TheoremQA-Questions	194	188,002

Table 9: Task subsets, retrieval modalities, and evaluation set sizes for the M-BEIR benchmark. I and T denote image and text, respectively.

Dataset	Task	Query $\rightarrow$ target	Queries	Candidates
VisualNews	0	$\mathrm{T} \rightarrow \mathrm{I}$	19,995	542,246
MSCOCO	0	$\mathrm{T} \rightarrow \mathrm{I}$	24,809	5,000
Fashion200K	0	$\mathrm{T} \rightarrow \mathrm{I}$	1,719	201,824
WebQA	1	$\mathrm{T} \rightarrow \mathrm{T}$	2,455	544,457
EDIS	2	$\mathrm{T} \rightarrow \mathrm{I} + \mathrm{T}$	3,241	1,047,067
WebQA	2	$\mathrm{T} \rightarrow \mathrm{I} + \mathrm{T}$	2,511	403,196
VisualNews	3	$\mathrm{I} \rightarrow \mathrm{T}$	20,000	537,568
MSCOCO	3	$\mathrm{I} \rightarrow \mathrm{T}$	5,000	24,809
Fashion200K	3	$\mathrm{I} \rightarrow \mathrm{T}$	4,889	61,707
NIGHTS	4	$\mathrm{I} \rightarrow \mathrm{I}$	2,120	40,038
OVEN	6	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{T}$	50,004	676,667
InfoSeek	6	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{T}$	11,323	611,651
CIRR	7	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{I}$	4,170	21,551
FashionIQ	7	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{I}$	6,003	74,381
OVEN	8	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{I} + \mathrm{T}$	14,741	335,135
InfoSeek	8	$\mathrm{I} + \mathrm{T} \rightarrow \mathrm{I} + \mathrm{T}$	17,593	481,782

A Experimental Setup

Benchmarks. We use BRIGHT (SU et al., 2025) for reasoning-intensive text retrieval and M-BEIR (Wei et al., 2024) for multimodal retrieval. BRIGHT contains 12 datasets spanning diverse domains, while M-BEIR contains 16 tasks covering text, image, and image–text retrieval. For BRIGHT, we evaluate all queries in each dataset. For M-BEIR, following the setup of Xu & Chen (2026), we uniformly sample 100 queries from each task and use the same sampled queries across all experiments. Following the task-specific M-BEIR preprocessing (Wei et al., 2024; Xu & Chen, 2026), we use the released text corpora for OVEN and InfoSeek Task 6, restrict retrieval to their corresponding candidate pools, and use the associated relevance labels for evaluation. Detailed benchmark statistics are provided in Tables 8 and 9.

Models. For BRIGHT, we use Diver-Retriever-4B-1020 (Sun et al., 2026) and Qwen3-Embedding-4B as embedding models. For M-BEIR, following Xu & Chen (2026), we use CLIP (ViT-B/32) (Radford et al., 2021) to encode all 16 task subsets. We follow the benchmark-specific query formatting by prepending each text-bearing query with its corresponding task instruction. For single-modality inputs, we ℓ_2-normalize the corresponding CLIP feature. For image–text inputs, we first ℓ_2-normalize the image and text features separately, sum the two normalized vectors, and then normalize the resulting vector again. Retrieval is performed using cosine similarity. For LLM-guided semantic selection in LGP and downstream reranking, we use Qwen3-32B (Team, 2025) for BRIGHT and

Table 10: Models used in the experiments.

Benchmark	Role	Model
BRIGHT	Embedding	Diver-Retriever-4B-1020
BRIGHT	Embedding	Qwen3-Embedding-4B
BRIGHT	Graph refinement / reranking	Qwen3-32B
M-BEIR	Embedding	CLIP (ViT-B/32)
M-BEIR	Graph refinement / reranking	Qwen3-VL-30B-A3B-Instruct

Qwen3-VL-30B-A3B-Instruct (Bai et al., 2025) for M-BEIR. Both models are used in their non-reasoning configuration. For the model-size ablation, we vary only the model used for semantic selection, using Qwen3-8B or Qwen3-32B, while keeping Qwen3-32B fixed as the downstream reranker. Table 10 summarizes the specific model configurations.

Index Configurations. Our primary ANN index is DiskANN (Jayaram Subramanya et al., 2019), configured with a maximum out-degree of 16, a construction search-list size of 125, and a pruning parameter of 1.2. We additionally evaluate HNSW (Malkov & Yashunin, 2020), using a connectivity parameter of 16 and a construction search width of 200. For both indices, LGP performs local graph pruning on top of the corresponding Vanilla index. For the textual retrieval setting, LGP uses edit budget $B = 2$, shortlist size $M = 24$, neighborhood-context size $|K_i| = 4$, and threshold $g_{\min} = 12$. For the multimodal setting, we use $B = 2$ and $M = 12$, with the same context size and threshold. For the graph-degree ablation, we set $g_{\min} = 6$ at $R = 8$ to match the default $g_{\min}/R$ ratio at $R = 16$, while using $g_{\min} = 12$ for all other degree settings.

Retrieval Configurations and Metrics. At query time, we evaluate two settings: (i) greedy search, and (ii) downstream LLM/VLM reranking of the candidate set returned by greedy search. We use *search width*, the standard query-time search-effort parameter in graph-based ANN indices, corresponding to L_{search} in DiskANN and ef_{search} in HNSW. Search width controls the breadth of candidates explored during graph traversal, with larger values generally requiring more node expansions and distance computations. With graph degree bounded by R, the distance-computation cost grows roughly with search width $\times$ R, making search width a practical control for query-time search effort. We therefore use search width as the primary query-time search-effort parameter throughout our main evaluation. For DiskANN, we vary $L_{\text{search}} \in \{10, 20, 50, 100, 200\}$; for HNSW, we vary $ef_{\text{search}} \in \{10, 12, 14, 16, 18, 20, 50\}$. By default, we set the search width to 20 for both indices to target the budget-constrained regime, where graph connectivity most strongly dictates which relevant documents are reached. For reranking, we use the candidates returned by graph search, capped at 200 for text retrieval and 64 for multimodal retrieval. At the default search width of 20, both rerankers therefore receive 20 candidates. Both rerankers use listwise sliding windows of 10 candidates with a stride of 5. For greedy search, the top k documents are selected from the returned candidates according to embedding distance. For reranking, the same candidate set is reordered by the model before evaluating the resulting top k.

Our primary retrieval metrics are NDCG@k and document-level Recall@k for $k \in \{5, 10\}$, reported on a 0–100 scale. NDCG@k measures ranking quality within the top k, normalized by the ideal ranking at the same cutoff. Recall@k measures the fraction of all ground-truth documents that appear in the top k, averaged over queries. Because reranking can change which documents appear in the top k, it can also change Recall@k even though the candidate set itself is unchanged. When a query has more than k relevant documents, its maximum Recall@k is below 100, so Recall@k and NDCG@k are not directly comparable in magnitude.

B Additional Results

B.1 Search-Width Results

Figures 5 and 7 show the per-dataset search-width sweeps using Diver embeddings on BRIGHT. The gains are broadly consistent across datasets, showing that the aggregate improvements are not driven by a small subset of tasks. This further supports that LGP refines graph connectivity in ways that better align with downstream semantic relevance.

Table 11: BRIGHT main results on DiskANN at search width 20 using Diver embeddings. Each entry reports Vanilla / LGP.

	Greedy search				Rerank			
Dataset	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10
Biology	22.22 / **32.66**	24.69 / **37.06**	21.42 / **32.10**	28.66 / **44.63**	26.95 / **39.15**	28.30 / **41.57**	26.22 / **38.47**	30.71 / **46.19**
Earth Science	19.80 / **26.32**	20.60 / **27.92**	19.13 / **25.80**	22.70 / **30.59**	21.91 / **28.03**	23.29 / **30.06**	20.62 / **27.30**	25.51 / **33.03**
Economics	18.68 / **21.00**	19.56 / **22.35**	15.62 / **18.81**	20.77 / **25.61**	18.84 / **24.01**	19.98 / **25.42**	16.09 / **22.46**	21.92 / **28.84**
Psychology	26.62 / **32.37**	28.31 / **34.76**	26.62 / **31.62**	34.51 / **41.61**	32.88 / **40.20**	32.78 / **41.46**	29.47 / **35.11**	34.40 / **44.10**
Robotics	19.05 / **21.19**	19.48 / **21.68**	20.59 / **22.35**	24.83 / **27.12**	25.68 / **28.47**	25.67 / **28.63**	24.27 / **25.98**	28.61 / **31.16**
StackOverflow	8.75 / **14.83**	10.82 / **17.36**	8.71 / **14.76**	13.87 / **21.37**	10.22 / **16.86**	11.96 / **19.61**	9.72 / **15.94**	14.66 / **23.73**
Sustainable Living	16.35 / **22.30**	18.09 / **24.33**	15.09 / **22.12**	22.12 / **29.76**	19.32 / **26.05**	20.20 / **27.43**	18.21 / **25.06**	22.04 / **30.82**
Pony	20.34 / **20.99**	17.08 / **17.80**	5.29 / **5.36**	9.06 / **9.40**	24.81 / **26.04**	19.54 / **20.07**	6.56 / **6.73**	**9.73** / 9.00
LeetCode	2.66 / **4.89**	3.20 / **6.20**	3.35 / **6.27**	4.54 / **9.51**	2.34 / **5.27**	2.97 / **6.37**	2.66 / **7.05**	4.27 / **10.07**
AoPS	2.67 / 2.67	3.13 / **3.36**	3.03 / 3.03	4.58 / **5.03**	2.39 / **2.46**	3.12 / **3.23**	2.52 / 2.52	4.69 / **4.79**
TheoremQA-Theorems	31.23 / **32.63**	34.70 / **36.35**	37.16 / **37.48**	46.26 / **47.57**	30.36 / **33.18**	33.57 / **36.44**	36.31 / **39.27**	45.60 / **48.89**
TheoremQA-Questions	5.39 / **6.20**	5.73 / **6.45**	7.56 / **7.86**	8.24 / **8.36**	**4.92** / 4.90	5.63 / **5.68**	7.52 / **7.56**	9.24 / **9.49**
Average	16.15 / **19.84**	17.12 / **21.30**	15.30 / **18.96**	20.01 / **25.05**	18.38 / **22.88**	18.92 / **23.83**	16.68 / **21.12**	20.95 / **26.68**

Table 12: M-BEIR main results on DiskANN at search width 20 using CLIP (ViT-B/32). Parentheses indicate the M-BEIR task ID. Each entry reports Vanilla / LGP.

	Greedy search				Rerank			
Dataset (Task)	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10
VisualNews (0)	**15.80** / 14.54	**16.41** / 15.45	**19.00** / 17.00	**21.00** / 20.00	14.26 / **15.02**	15.23 / **15.97**	18.00 / **18.00**	21.00 / **21.00**
MSCOCO (0)	28.26 / **28.33**	32.45 / **32.88**	38.00 / **38.00**	51.00 / **52.00**	42.72 / **43.72**	45.09 / **46.09**	49.00 / **50.00**	56.00 / **57.00**
Fashion200K (0)	1.51 / **1.51**	1.80 / **1.82**	1.79 / **1.79**	2.63 / **2.63**	**4.41** / 3.41	**4.53** / 3.56	**3.97** / 2.97	**4.97** / 3.97
WebQA (1)	5.38 / **6.00**	5.99 / **6.61**	6.00 / **6.50**	7.50 / **8.00**	8.29 / **8.76**	8.29 / **8.93**	8.00 / **8.50**	8.00 / **9.00**
EDIS (2)	5.87 / **9.89**	6.76 / **11.29**	6.49 / **10.57**	9.07 / **14.74**	10.33 / **14.22**	11.02 / **15.67**	10.63 / **14.05**	12.80 / **18.71**
WebQA (2)	12.99 / **21.73**	15.15 / **22.97**	14.50 / **26.00**	20.50 / **29.50**	19.50 / **28.56**	20.25 / **29.18**	21.50 / **32.00**	23.50 / **33.50**
VisualNews (3)	2.65 / **5.77**	3.87 / **7.04**	5.00 / **10.00**	9.00 / **14.00**	3.97 / **7.77**	5.25 / **10.29**	7.00 / **11.00**	11.00 / **19.00**
MSCOCO (3)	37.05 / **37.51**	42.42 / **43.00**	32.80 / **32.80**	42.60 / **42.80**	49.83 / **51.11**	52.63 / **53.92**	44.20 / **44.80**	49.20 / **49.80**
Fashion200K (3)	**1.93** / 1.86	**2.93** / 2.86	3.00 / **3.00**	6.00 / **6.00**	3.26 / **3.65**	3.97 / **4.00**	4.00 / **5.00**	6.00 / **6.00**
NIGHTS (4)	23.93 / **23.93**	28.73 / **28.73**	36.00 / **36.00**	51.00 / **51.00**	38.01 / **39.22**	41.99 / **42.51**	49.00 / **51.00**	61.00 / **61.00**
OVEN (6)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
InfoSeek (6)	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00	0.00 / 0.00
CIRR (7)	6.16 / **6.86**	6.47 / **7.83**	12.00 / **13.00**	13.00 / **16.00**	10.99 / **12.34**	10.99 / **12.34**	13.00 / **15.00**	13.00 / **15.00**
FashionIQ (7)	0.00 / **0.43**	0.67 / **1.12**	0.00 / **1.00**	2.00 / **3.00**	0.63 / **2.63**	1.61 / **3.66**	1.00 / **3.00**	4.00 / **6.00**
OVEN (8)	0.25 / **0.59**	0.50 / **1.35**	0.25 / **0.27**	0.75 / **2.08**	1.20 / **3.39**	1.16 / **3.09**	0.75 / **2.08**	0.75 / **2.08**
InfoSeek (8)	0.26 / **0.41**	0.48 / **0.93**	0.50 / **0.58**	1.00 / **1.58**	0.26 / **2.82**	0.45 / **2.53**	0.50 / **1.62**	1.00 / **2.31**
Average	8.88 / **9.96**	10.29 / **11.49**	10.96 / **12.28**	14.82 / **16.46**	12.98 / **14.79**	13.90 / **15.73**	14.41 / **16.19**	17.01 / **19.02**

Table 13: Ablation of candidate shortlisting on Pony, Psychology, and Biology from BRIGHT at search width 20 using Diver embeddings. Entries report NDCG@10.

	Pony		Psychology		Biology	
Variant	Greedy	Rerank	Greedy	Rerank	Greedy	Rerank
Proximity only	16.89	18.94	33.71	39.52	33.07	37.69
Coverage only	**17.84**	**20.22**	33.18	39.30	37.06	41.17
Separation only	17.19	20.02	33.45	39.68	**39.22**	**45.71**
Balanced	17.80	20.07	**34.76**	**41.46**	37.06	41.57

B.2 Embedding-Model Ablations

Per-dataset BRIGHT results using Qwen3-Embedding are shown in Figures 9 and 10. Across datasets and search widths, LGP consistently matches or outperforms Vanilla, further demonstrating that its gains are robust to the choice of embedding model.

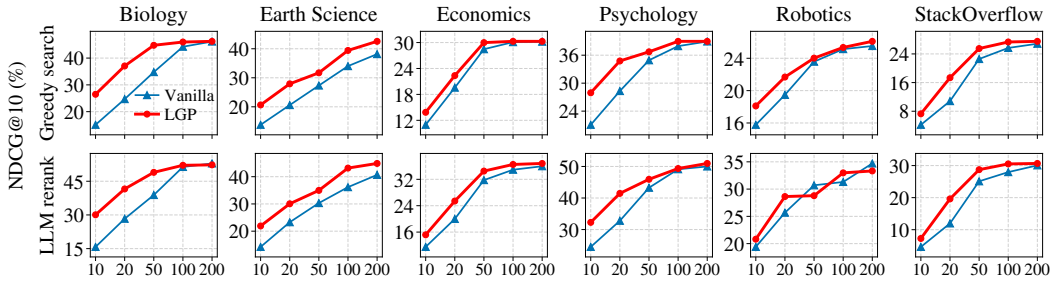

Figure 5: Search-width sweeps on six BRIGHT datasets using Diver embeddings (part 1 of 2).

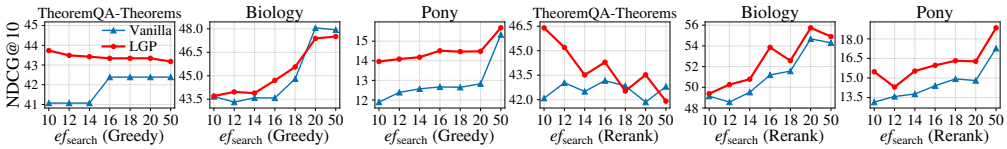

Figure 6: HNSW search-width sweeps on 3 BRIGHT datasets. The left three panels show greedy-search NDCG@10, while the right three show NDCG@10 after LLM reranking.

Table 14: Ablation of low-value neighbor identification on Pony, Psychology, and Biology from BRIGHT at search width 20 using Diver embeddings. Entries report NDCG@10.

	Pony		Psychology		Biology	
Variant	Greedy	Rerank	Greedy	Rerank	Greedy	Rerank
Redundancy only	17.36	**20.22**	32.38	38.04	33.93	39.74
Coverage only	17.73	20.03	34.30	41.15	34.41	38.86
Lexicographic	**17.80**	20.07	**34.76**	**41.46**	**37.06**	**41.57**

Table 15: Graph-structure diagnostics for paired Vanilla and LGP indices on BRIGHT. Each dataset contributes equally to the reported averages, and all indices use a maximum degree of 16. Changed neighborhoods denote the percentage of nodes whose outgoing-neighbor sets differ between the paired indices.

Embedding	Mean out-degree		Changed neighborhoods (%)
	Vanilla	LGP	
Diver	15.60	15.66	88.5
Qwen3-Embedding	15.88	15.80	88.9

B.3 Design Analysis of LGP

Candidate Shortlisting. We ablate the shortlisting strategy while keeping the other designs and parameters of LGP fixed. We consider three alternative shortlisting methods: (i) *Proximity only* selects candidates closest to the anchor in embedding space after excluding those geometrically covered by its current neighbors; (ii) *Coverage only* ranks candidates by decreasing additional two-hop coverage; and (iii) *Separation only* ranks candidates by decreasing minimum embedding distance to the current neighbors. The default *Balanced* strategy in LGP combines candidates from all three rankings under the same shortlist budget. As shown in Table 13, the most effective shortlisting strategy varies across datasets, with coverage favored on Pony, balanced on Psychology, and separation on Biology. This variation exactly motivates our balanced strategy, which combines all three signals and maintains strong performance across datasets and retrieval settings. Thus, balanced shortlisting provides a robust default that generalizes across datasets without dataset-specific tuning.

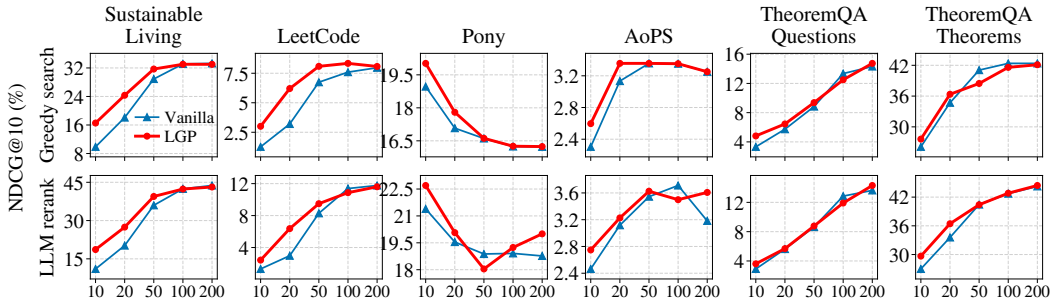

Figure 7: Search-width sweeps on the remaining six BRIGHT datasets using Diver embeddings (part 2 of 2).

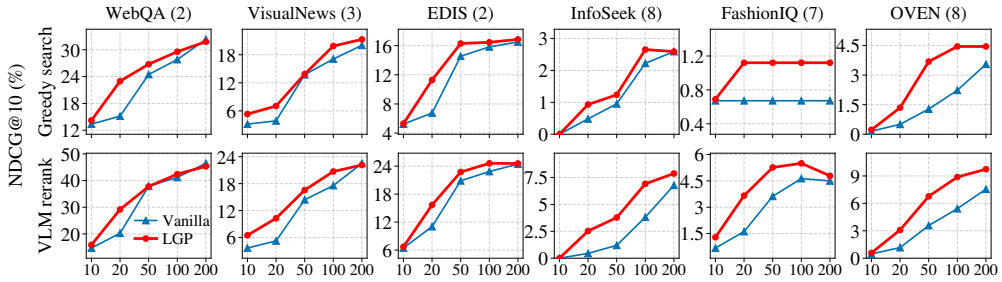

Figure 8: Search-width sweeps on 6 M-BEIR tasks using CLIP embeddings. LGP consistently matches or outperforms Vanilla under both greedy search and VLM reranking across search widths.

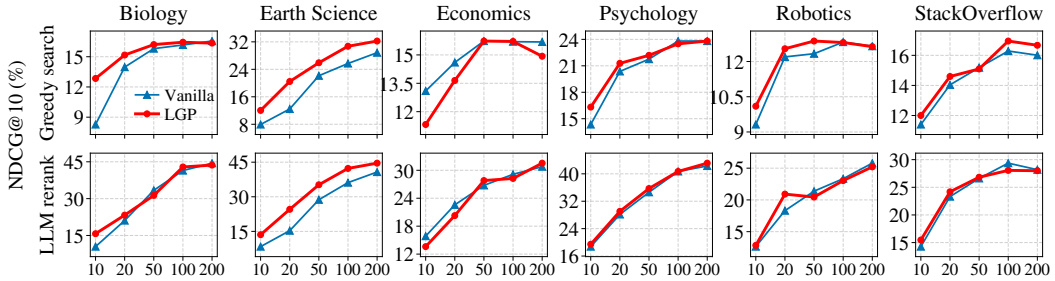

Figure 9: Search-width sweeps on six BRIGHT datasets using Qwen3-Embedding (part 1 of 2).

Low-Value Neighbor Identification. We ablate the criteria used to identify low-value existing neighbors for replacement while keeping all other components of LGP fixed. Besides our default *Lexicographic* rule, we consider two additional variants: (i) *Redundancy only*, which favors geometrically redundant neighbors, and (ii) *Coverage only*, which identifies neighbors with the smallest unique contribution to two-hop coverage. The default *Lexicographic* rule first prioritizes low unique coverage, then greater redundancy. As shown in Table 14, the lexicographic rule achieves the strongest greedy-search performance on all three datasets and the strongest reranked performance on Psychology and Biology, while remaining close to the best result on Pony. These results show that coverage and redundancy provide complementary signals for identifying low-value neighbors, making the lexicographic rule a robust default across datasets.

Graph Structure After Pruning. As shown in Table 15, LGP changes approximately 88–89% of neighborhoods across BRIGHT, while the mean out-degree remains nearly unchanged for both embedding models and stays below the maximum degree of 16. Thus, the retrieval gains of LGP come from reconfiguring graph connectivity rather than increasing graph density.

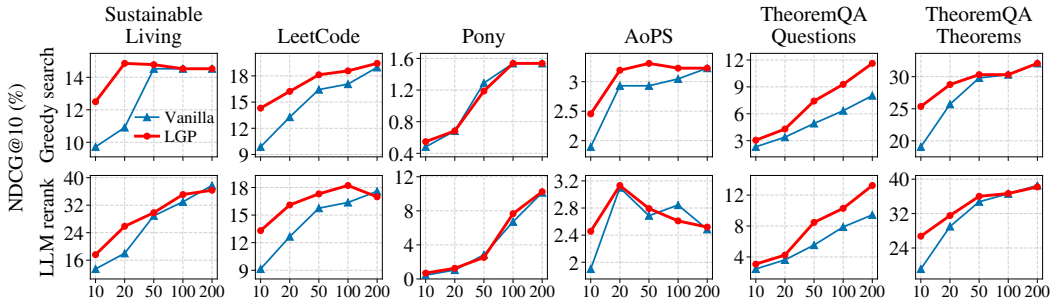

Figure 10: Search-width sweeps on the remaining six BRIGHT datasets using Qwen3-Embedding (part 2 of 2).

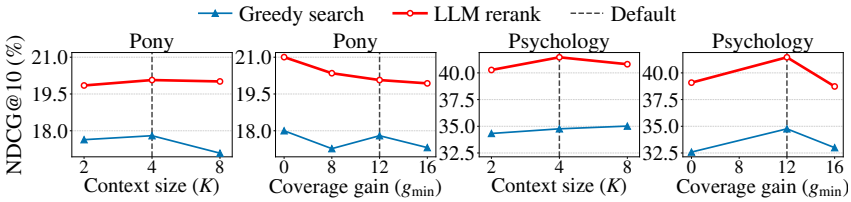

Figure 11: Additional hyperparameter ablations at search width 20. The left two panels show Pony and the right two show Psychology, with dashed lines indicating the default settings.

Table 16: Offline graph-construction cost using Diver embeddings and Qwen3-32B on a single NVIDIA H200. Build times are reported in minutes, token counts in millions, and per-node time as total LGP build time divided by the number of graph nodes. LLM calls and token counts refer to LGP.

Dataset	Nodes	Build time (min)		LGP time (s/node)	LLM calls	Tokens (M)	
		Vanilla	LGP			Input	Output
Pony	7,894	1.49	109.46	0.83	7,894	23.95	0.22
Psychology	52,835	11.84	691.40	0.79	52,835	124.56	1.50

B.4 Additional Ablation on Hyperparameters in LGP

Neighborhood Context Size K. We vary the neighborhood context size K, which controls the number of current neighbors provided to the LLM/VLM when selecting new neighbors that add complementary semantic coverage. As shown in Figure 11, the default $K = 4$ gives the strongest reranked NDCG@10 on both Pony and Psychology. Increasing the context to $K = 8$ provides no additional reranking benefit, suggesting that a small set of representative neighbors already captures most of the useful neighborhood context, while additional neighbors may introduce redundant information that makes semantic distinctions harder for the LLM/VLM.

Coverage Threshold $g_{\min}$. We vary the coverage threshold $g_{\min}$, which controls the minimum additional two-hop coverage required for an LLM-selected candidate to be accepted as a new neighbor. As shown in Figure 11, its effect varies across datasets: Pony performs best with the relaxed setting $g_{\min} = 0$, whereas Psychology performs best at the default $g_{\min} = 12$ under both greedy search and reranking. These results indicate that the preferred coverage constraint is dataset-dependent rather than uniformly favoring stronger coverage requirements. As shown in Figure 11, the default $g_{\min} = 12$ remains competitive on Pony and achieves the strongest performance on Psychology under both greedy search and reranking.

Table 17: Distance-computation-budget sweeps on BRIGHT and M-BEIR using Diver and CLIP embeddings, respectively. Results are averaged equally across datasets and tasks. Reranking uses the top 20 retrieved candidates at each budget. Green subscripts denote relative improvements over Vanilla, computed before rounding. Gray values denote average exact-kNN NDCG@10 / Recall@10 without reranking.

Benchmark (Exact kNN ref.)	Computation budget	Greedy search				Rerank			
		NDCG@10		Recall@10		NDCG@10		Recall@10	
		Vanilla	LGP	Vanilla	LGP	Vanilla	LGP	Vanilla	LGP
BRIGHT (28.50 / 34.18)	100	3.96	**5.02**$_{+26.9\%}$	2.96	**4.15**$_{+40.0\%}$	4.28	**5.40**$_{+26.1\%}$	3.07	**4.12**$_{+34.3\%}$
	200	10.50	**13.98**$_{+33.2\%}$	11.62	**15.60**$_{+34.3\%}$	11.67	**15.41**$_{+32.0\%}$	12.32	**16.40**$_{+33.1\%}$
	300	16.69	**19.80**$_{+18.6\%}$	19.18	**23.19**$_{+20.9\%}$	18.70	**22.46**$_{+20.1\%}$	20.35	**24.61**$_{+21.0\%}$
	500	22.92	**24.57**$_{+7.2\%}$	27.14	**28.83**$_{+6.2\%}$	26.04	**27.79**$_{+6.7\%}$	28.70	**30.82**$_{+7.4\%}$
	800	25.89	**26.61**$_{+2.8\%}$	30.66	**31.37**$_{+2.3\%}$	29.00	**29.92**$_{+3.2\%}$	32.59	**33.54**$_{+2.9\%}$
	1600	27.11	**27.27**$_{+0.6\%}$	32.27	**32.32**$_{+0.2\%}$	30.83	**30.95**$_{+0.4\%}$	34.63	**34.71**$_{+0.2\%}$
	3200	27.65	**27.74**$_{+0.3\%}$	32.90	**32.96**$_{+0.2\%}$	31.26	**31.28**$_{+0.1\%}$	35.15	**35.18**$_{+0.1\%}$
M-BEIR (16.16 / 22.87)	100	2.86	**2.97**$_{+3.8\%}$	3.88	**4.14**$_{+6.8\%}$	3.53	**3.69**$_{+4.4\%}$	4.21	**4.59**$_{+8.9\%}$
	200	7.60	**7.71**$_{+1.4\%}$	10.89	**10.97**$_{+0.8\%}$	10.13	**10.35**$_{+2.1\%}$	12.47	**12.56**$_{+0.7\%}$
	300	9.56	**10.04**$_{+5.0\%}$	13.61	**14.42**$_{+5.9\%}$	12.61	**13.49**$_{+7.0\%}$	15.32	**16.40**$_{+7.1\%}$
	500	11.60	**12.57**$_{+8.4\%}$	16.52	**17.69**$_{+7.1\%}$	15.33	**17.01**$_{+11.0\%}$	18.52	**20.41**$_{+10.2\%}$
	800	13.28	**13.70**$_{+3.2\%}$	18.50	**19.25**$_{+4.0\%}$	17.79	**18.58**$_{+4.5\%}$	21.46	**22.16**$_{+3.2\%}$
	1600	14.44	**14.98**$_{+3.7\%}$	20.10	**20.85**$_{+3.7\%}$	19.53	**20.26**$_{+3.7\%}$	23.51	**24.12**$_{+2.6\%}$
	3200	15.27	**15.51**$_{+1.6\%}$	21.17	**21.48**$_{+1.5\%}$	20.59	**20.80**$_{+1.0\%}$	24.59	**24.75**$_{+0.7\%}$

B.5 Pruning Cost Analysis

We profile the offline construction cost of Vanilla and LGP on Pony and Psychology using our Python DiskANN implementation with stored Diver embeddings. We use the default $R = 16$, $B = 2$, $M = 24$, $K = 4$, and $g_{\min} = 12$, with the full pruning budget $\beta = 1$. Each run uses eight CPU cores and one NVIDIA H200, with Qwen3-32B for semantic selection. As shown in Table 16, LGP constructs the Pony and Psychology indices in 109.46 and 691.40 minutes, compared with 1.49 and 11.84 minutes for Vanilla. More than 96% of the LGP construction time is spent waiting for LLM responses. At the full pruning budget, LGP makes one LLM call per node on average in both measured runs, with average graph-construction times of 0.83 and 0.79 seconds per node on Pony and Psychology, respectively. The additional cost is entirely offline: once constructed, the graph can be reused across queries without further calls to the semantic-selection model, while the query-time graph-search procedure remains unchanged.

B.6 Evaluation under Fixed Distance-Computation Budgets

Search width is the standard query-time control parameter for graph-based ANN indices such as DiskANN and HNSW. By controlling the size of the candidate set explored during graph search, it directly regulates search effort: with graph degree bounded by R, the number of distance computations scales roughly as $O(\text{SEARCH WIDTH} \times R)$. We therefore use search width as the primary query-time budget parameter throughout our main evaluation. Because the exact number of distance computations at a fixed width may vary with the graph topology, we additionally provide a complementary comparison of Vanilla and LGP under the same explicit query–document distance-computation budgets.

Specifically, we vary the distance-computation budget over $\{100, 200, 300, 500, 800, 1600, 3200\}$ and evaluate Vanilla and LGP on BRIGHT and M-BEIR using Diver and CLIP embeddings, respectively, with DiskANN. We report both greedy-search performance and performance after reranking the top 20 retrieved candidates. As shown in Table 17, LGP consistently outperforms Vanilla at every tested computation budget under both greedy search and reranking on both benchmarks. The gains are largest at smaller and intermediate budgets. For example, on BRIGHT at a budget of 200 distance computations, LGP improves greedy-search NDCG@10 and Recall@10 by **33.2%** and **34.3%**, while reranked NDCG@10 and Recall@10 improve by **32.0%** and **33.1%**, respectively. Moreover, on Economics from BRIGHT and OVEN Task 8 from M-BEIR, LGP outperforms Vanilla in NDCG@10 under both greedy search and reranking with only half the distance-computation budget (800 vs. 1,600 per query), corresponding to 50% fewer computations. As the computation budget increases and both methods approach exact retrieval, the gap naturally narrows. These results show that LGP

improves search efficiency by achieving higher retrieval quality under the same computation budget and comparable quality with substantially fewer distance evaluations.

C PROMPTS

C.1 PROMPT USED IN LLMSELECT

The LLMSELECT operation in our algorithm uses the prompt below.

System Prompt

```
You are editing a theorem retrieval graph at indexing time.
There is no real user query available, and you must not assume a
↪  particular test query.
Instead, privately simulate plausible latent user queries /
↪  problem intents for which the ANCHOR might be retrieved as a
↪  nearby theorem.

Your goal is to improve query-time routing from the ANCHOR to
↪  useful answers.
First, if helpful, optionally write a short latent query line
↪  beginning with QUERY: ; then select at most the best
↪  candidates for shortcut edges.

For each CANDIDATE, ask: would a user query that lands near the
↪  ANCHOR also reasonably need this CANDIDATE as the target
↪  theorem, a prerequisite theorem, a bridge theorem, a
↪  generalization/special case, or a theorem used in the same
↪  proof/problem-solving route?

Select a CANDIDATE only if adding an edge ANCHOR -> CANDIDATE
↪  would help query-time graph search route from the ANCHOR's
↪  local neighborhood toward useful answers that are not
↪  already covered by the ALREADY-SELECTED LOCAL NEIGHBORS.

Do NOT select based on wording similarity alone.
Do NOT select broad but unrelated facts.
If no candidate has a clear latent-query/use-case connection,
↪  return NONE.
Return at most {B} candidate numbers, best first, or NONE.
↪  Examples: [3] [1] or NONE
```

User Prompt

```
ANCHOR:
{anchor}

ALREADY-SELECTED LOCAL NEIGHBORS:
{keep_block}

CANDIDATE SHORTCUTS:
{cand_block}

Privately imagine the latent queries/use-cases, then choose
↪  which candidates should be connected to the ANCHOR for
↪  query-time routing. Return only candidate numbers or NONE.
If you include a latent query, start it with QUERY: on its own
↪  line.
```

C.2 TEXT RERANKING PROMPT

Text Reranking: System Prompt

```
You are an intelligent assistant that can rank answers based on their
↪  relevancy to the query.
I will provide you with {N} passages, each indicated by number
↪  identifier [].
Rank the answers based on their relevance to query: {QUERY}.
```

Text Reranking: User Prompt

```
[1] {passage_1}

...

[{N}] {passage_N}

Query: {QUERY}.
Rank the {N} passages above based on their relevance to the query.
The passages should be listed in descending order using identifiers.
The most relevant passages should be listed first.
The output format should be like [1] > [2] ... > [{N}].
Only response the ranking results, do not say any word or explain.
```

C.3 MULTIMODAL RERANKING PROMPT

Multimodal Reranking: System Prompt

```
You rank retrieval candidates for multimodal retrieval.
Given one QUERY and several CANDIDATES, rank the candidates from best
↪  to worst match.
Use all available evidence, including image and text when present.
Return only an ordering of candidate indices like [2] [1] [3].
```

Multimodal Reranking: User Content

```
QUERY:
Task instruction:
{task_instruction}
Query text:
{query_text}
<query image>

CANDIDATES:
[1]
Candidate 1 text:
{candidate_1_text}
<candidate 1 image>
...
[{N}]
Candidate {N} text:
{candidate_N_text}
<candidate N image>

Rank the CANDIDATES from best to worst match for the QUERY.
Use image and text jointly when available.
Return only candidate indices like [2] [1] [3].
```

D ALGORITHMS OF DISKANN AND HNSW

We provide the detailed algorithms of DiskANN in Algorithms 2 to 4 and HNSW in Algorithms 5 to 7. For HNSW, $N_\ell(i)$ denotes the outgoing neighborhood of v_i at layer ℓ, and the unlayered notation $N(i)$ used in our base-layer pruning refers to $N_0(i)$. We denote the construction search width by L_{build} and use a generic width argument L in the search routines below. DiskANN construction uses $L = L_{\text{build}}$. HNSW insertion uses $L = 1$ when descending through layers above the new node's maximum level, and $L = L_{\text{build}}$ when searching layers in which the node is inserted. At query time, DiskANN uses $L = L_{\text{search}}$. HNSW uses $L = 1$ for upper-layer descent and $L = ef_{\text{search}}$ for base-layer search.

Algorithm 2 DiskANN Indexing

Input Document embeddings $\{\boldsymbol{e}(d_i)\}_{i=1}^{|V|}$, degree budget R, construction search width L_{build}, pruning parameter α
Result Graph index $G = (V, E)$ with local neighborhoods $\{N(i)\}_{i=1}^{|V|}$
Initialize G as a random R-regular graph over $V = \{v_1, \ldots, v_{|V|}\}$
Let s be the vertex whose embedding is closest to the centroid of $\{\boldsymbol{e}(d_i)\}_{i=1}^{|V|}$
for $t = 1$ to 2 **do**
  Sample a random permutation σ of $\{1, \ldots, |V|\}$
  for $k = 1$ to $|V|$ **do**
    $i \leftarrow \sigma(k)$
    $U_i \leftarrow \text{GREEDYSEARCH}(s, v_i, L_{\text{build}})$
    $N(i) \leftarrow \text{ROBUSTPRUNING}(v_i, U_i, \alpha, R)$
    for each $v_j \in N(i)$ **do**
      $N(j) \leftarrow N(j) \cup \{v_i\}$
      if $|N(j)| > R$ **then**
        $N(j) \leftarrow \text{ROBUSTPRUNING}(v_j, N(j), \alpha, R)$

Algorithm 3 GREEDYSEARCH in DiskANN

Input Start point s, query point q, search width L
Result Visited vertex list U
$A \leftarrow \{s\}, \quad U \leftarrow \varnothing$
while $A \setminus U \neq \varnothing$ **do**
  $v_j \leftarrow \arg\min_{x \in A \setminus U} D(x, q)$
  $A \leftarrow A \cup N(j)$
  $U \leftarrow U \cup \{v_j\}$
  if $|A| > L$ **then**
    $A \leftarrow$ the L vertices in A closest to q
Sort U in increasing distance from q
return U

Algorithm 4 ROBUSTPRUNING in DiskANN

Input Vertex v_i, candidate neighbor set U, pruning parameter α, degree limit R
Result Updated $N(i)$, the set of out-neighbors of v_i
$U \leftarrow (U \cup N(i)) \setminus \{v_i\}$
$N(i) \leftarrow \varnothing$
while $U \neq \varnothing$ **and** $|N(i)| < R$ **do**
  $v_j \leftarrow \arg\min_{x \in U} D(x, v_i)$
  $N(i) \leftarrow N(i) \cup \{v_j\}$
  $U \leftarrow U \setminus \{v_j\}$
  $U \leftarrow \{v_k \in U : \alpha D(v_j, v_k) > D(v_i, v_k)\}$
return $N(i)$

Algorithm 5 HNSW Insertion

Input Vertex v_i, current entry point s, neighbor budget R, base-layer degree cap R_0, construction search width L_{build}, level-scale parameter m_ℓ, flags EXTENDCANDIDATES and KEEPPRUNED
Result Updated HNSW graph after inserting v_i
Sample the maximum level ℓ_i of v_i according to the level distribution parameterized by m_ℓ
Let $\ell_{\max}$ be the level of the current entry point
$S_{\text{entry}} \leftarrow \{s\}$
for $\ell = \ell_{\max}, \ell_{\max} - 1, \ldots, \ell_i + 1$ **do**
    $W \leftarrow$ SEARCH-LAYER$(v_i, S_{\text{entry}}, 1, \ell)$
    $S_{\text{entry}} \leftarrow \{\arg\min_{x \in W} D(x, v_i)\}$
for $\ell = \min(\ell_{\max}, \ell_i), \min(\ell_{\max}, \ell_i) - 1, \ldots, 0$ **do**
    $W \leftarrow$ SEARCH-LAYER$(v_i, S_{\text{entry}}, L_{\text{build}}, \ell)$
    $N_\ell(i) \leftarrow$ SELECT-NEIGHBORS$(v_i, W, R, \ell,$ EXTENDCANDIDATES, KEEPPRUNED$)$
    Add bidirectional connections between v_i and all nodes in $N_\ell(i)$ at layer ℓ
    for each $v_j \in N_\ell(i)$ **do**
        Let $N_\ell(j)$ be the current neighborhood of v_j at layer ℓ
        if $\ell = 0$ **and** $|N_\ell(j)| > R_0$ **then**
            $N_\ell(j) \leftarrow$ SELECT-NEIGHBORS$(v_j, N_\ell(j), R_0, \ell,$ EXTENDCANDIDATES, KEEPPRUNED$)$
        else if $\ell > 0$ **and** $|N_\ell(j)| > R$ **then**
            $N_\ell(j) \leftarrow$ SELECT-NEIGHBORS$(v_j, N_\ell(j), R, \ell,$ EXTENDCANDIDATES, KEEPPRUNED$)$
    $S_{\text{entry}} \leftarrow W$
if $\ell_i > \ell_{\max}$ **then**
    Set the entry point of the graph to v_i

Algorithm 6 SEARCH-LAYER in HNSW

Input Query q, entry-point set S_{entry}, search width L, layer ℓ
Result Candidate set W
$V_{\text{seen}} \leftarrow S_{\text{entry}}, \mathcal{C} \leftarrow S_{\text{entry}}, W \leftarrow S_{\text{entry}}$
while $\mathcal{C} \neq \varnothing$ **do**
    $v_j \leftarrow \arg\min_{x \in \mathcal{C}} D(x, q)$
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{v_j\}$
    $v_f \leftarrow \arg\max_{x \in W} D(x, q)$
    if $D(v_j, q) > D(v_f, q)$ **then**
        break
    for each $v_k \in N_\ell(j)$ **do**
        if $v_k \notin V_{\text{seen}}$ **then**
            $V_{\text{seen}} \leftarrow V_{\text{seen}} \cup \{v_k\}$
            $v_f \leftarrow \arg\max_{x \in W} D(x, q)$
            if $|W| < L$ **or** $D(v_k, q) < D(v_f, q)$ **then**
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{v_k\}$
                $W \leftarrow W \cup \{v_k\}$
                if $|W| > L$ **then**
                    $v_f \leftarrow \arg\max_{x \in W} D(x, q)$
                    $W \leftarrow W \setminus \{v_f\}$
return W

Algorithm 7 SELECT-NEIGHBORS in HNSW

Input Vertex v_i, candidate set $\mathcal{C}$, neighbor budget R, layer ℓ, flags EXTENDCANDIDATES and KEEPPRUNED
Result Selected neighbor set S
$S \leftarrow \varnothing$
$W \leftarrow \mathcal{C}$
if EXTENDCANDIDATES **then**
  for each $v_j \in \mathcal{C}$ **do**
    for each $v_k \in N_\ell(j)$ **do**
      if $v_k \notin W$ **then**
        $W \leftarrow W \cup \{v_k\}$
$W_{\text{discard}} \leftarrow \varnothing$
while $W \neq \varnothing$ **and** $|S| < R$ **do**
  $v_j \leftarrow \arg\min_{x \in W} D(x, v_i)$
  $W \leftarrow W \setminus \{v_j\}$
  if $D(v_j, v_i) \leq D(v_j, v_k),\ \forall v_k \in S$ **then**
    $S \leftarrow S \cup \{v_j\}$
  else
    $W_{\text{discard}} \leftarrow W_{\text{discard}} \cup \{v_j\}$
if KEEPPRUNED **then**
  while $W_{\text{discard}} \neq \varnothing$ **and** $|S| < R$ **do**
    $v_j \leftarrow \arg\min_{x \in W_{\text{discard}}} D(x, v_i)$
    $W_{\text{discard}} \leftarrow W_{\text{discard}} \setminus \{v_j\}$
    $S \leftarrow S \cup \{v_j\}$
return S

Algorithm 8 LLM-Guided Graph Pruning (LGP)

Input ANN graph G, current node v_i, neighborhood $N(i)$, and index-specific candidate set U_i
Parameters Degree cap R, shortlist budget M, per-node edit budget B, neighborhood context size K, and threshold $g_{\min}$
▷ /* Form a local replacement set */
$T_i \leftarrow \left(\bigcup_{v_j \in N(i)} N(j)\right) \setminus \{v_i\}$
$U_i \leftarrow (U_i \cup T_i) \setminus (N(i) \cup \{v_i\})$
▷ /* Shortlist local replacement candidates without LLM calls */
for each $v_c \in U_i$ **do**
  $g_i(c) \leftarrow |N(c) \setminus T_i|$
  $\nu_i(c) \leftarrow \min_{v_j \in N(i)} D(v_c, v_j)$
  $\delta_i(c) \leftarrow D(v_i, v_c)$
$L_g \leftarrow \textsc{Sort}_{\downarrow}(U_i; g_i)$
$L_\nu \leftarrow \textsc{Sort}_{\downarrow}(U_i; \nu_i)$
$L_\delta \leftarrow \textsc{Sort}_{\downarrow}(U_i; \delta_i)$
$m_{\text{src}} \leftarrow \lfloor M/3 \rfloor$
$U_i \leftarrow$ the first M distinct candidates in $L_\delta[1\!:\!m_{\text{src}}] \,\|\, L_g[1\!:\!m_{\text{src}}] \,\|\, L_\nu$
▷ /* Select semantically useful candidates with an LLM */
$\mathcal{K}_i \leftarrow$ the $\min(K, |N(i)|)$ nodes in $N(i)$ closest to v_i
$\mathcal{A}_i \leftarrow \textsc{LLMSelect}(v_i, \mathcal{K}_i, U_i, B)$
▷ /* Local edge pruning and replacement */
$N'(i) \leftarrow N(i)$
for each $v_c \in \mathcal{A}_i$ in selection order **do**
  if $v_c \in N'(i)$ **or** $g_i(c) < g_{\min}$ **then**
    continue
  if $|N'(i)| < R$ **then**
    $N'(i) \leftarrow N'(i) \cup \{v_c\}$
    continue
  if $N'(i) \cap N(i) = \varnothing$ **then**
    continue
  for each original neighbor $v_j \in N'(i) \cap N(i)$ **do**
    $\ell_i(j) \leftarrow \left| N(j) \setminus \left(\left(\bigcup_{v_k \in N'(i) \setminus \{v_j\}} N(k) \right) \setminus \{v_i\} \right) \right|$
    $\nu_i^{-j}(j) \leftarrow \min_{v_k \in N'(i) \setminus \{v_j\}} D(v_j, v_k)$
  $v_w \leftarrow \arg\min^{\text{lex}}_{v_j \in N'(i) \cap N(i)} \left(\ell_i(j), \nu_i^{-j}(j)\right)$
  $N'(i) \leftarrow (N'(i) \setminus \{v_w\}) \cup \{v_c\}$
return $N'(i)$

E Detailed LGP Algorithms and Index-Specific Instantiations

Algorithm 8 provides the detailed algorithm for LGP. We first describe its candidate construction, semantic selection, and edge-replacement steps, and then show how the same local update is instantiated in DiskANN and HNSW.

Efficient Local Candidate Set Construction. For each node v_i in the given ANN graph G, LGP forms a local set of possible replacement neighbors $\mathcal{C}_i$ rather than considering every node in the graph. Since most graph-based ANN methods already produce a candidate set when selecting or updating the outgoing neighborhood $N(i)$ of v_i, LGP reuses this set, denoted by U_i, as its initial replacement set and augments it with the nodes exposed through $N(i)$ (lines 3-5 in Algorithm 8):

$$T_i := \left(\bigcup_{v_j \in N(i)} N(j) \right) \setminus \{v_i\}, \qquad U_i := (U_i \cup T_i) \setminus (N(i) \cup \{v_i\}) \,. \tag{2}$$

Here, T_i contains the nodes reachable from v_i in two directed hops through its current neighbors, excluding v_i itself. After removing existing neighbors, the updated U_i therefore contains local non-neighbor alternatives drawn from both the index-specific candidate set and the directed two-hop neighborhood.

To bound the cost of LLM-guided selection and avoid excessively long contexts, LGP further reduces U_i to at most M candidates. (lines 6–15 in Algorithm 1). For each candidate node $v_c \in U_i$, LGP

Algorithm 9 DiskANN construction with LGP. Colored lines indicate operations introduced by LGP.

Input Document embeddings $\{\boldsymbol{e}(d_i)\}_{i=1}^{|V|}$, degree budget R, construction search width L_{build}, pruning parameter α, and LGP parameters $(B, M, K, g_{\min})$
Result Graph index $G = (V, E)$ with refined neighborhoods $\{N(i)\}_{i=1}^{|V|}$
Initialize G as a random R-regular graph and choose the centroid-nearest entry point s
for $t = 1$ to 2 **do**
  Sample a random permutation σ of $\{1, \ldots, |V|\}$
  for $k = 1$ to $|V|$ **do**
    $i \leftarrow \sigma(k)$
    $U_i \leftarrow \text{GREEDYSEARCH}(s, v_i, L_{\text{build}})$
    $N(i) \leftarrow \text{ROBUSTPRUNING}(v_i, U_i, \alpha, R)$
    $N_{\text{pre}}(i) \leftarrow N(i)$
    if $t = 2$ **then**
      $N(i) \leftarrow \text{LGP}(G, v_i, N(i), U_i, R, M, B, K, g_{\min})$
    for each $v_j \in N(i)$ **do**
      Insert v_i into $N(j)$ if absent
      if $|N(j)| > R$ **then**
        if $t = 2$ **and** $v_j \in N(i) \setminus N_{\text{pre}}(i)$ **then**
          $N(j) \leftarrow \text{ROBUSTPRUNING-PROTECTED}(v_j, N(j), \alpha, R, v_i)$
        else
          $N(j) \leftarrow \text{ROBUSTPRUNING}(v_j, N(j), \alpha, R)$
return G

considers three complementary ranking signals:

$$\delta_i(c) := D(v_i, v_c), \qquad g_i(c) := |N(c) \setminus T_i|, \qquad \nu_i(c) := \min_{v_j \in N(i)} D(v_c, v_j). \tag{3}$$

Here, $\delta_i(c)$ measures proximity to v_i, $g_i(c)$ measures the additional two-hop coverage introduced by v_c, and $\nu_i(c)$ is its distance to the closest current neighbor, with larger values indicating less geometric redundancy. Let L_g, L_ν, and L_δ denote the rankings of U_i by decreasing $g_i(c)$, decreasing $\nu_i(c)$, and increasing $\delta_i(c)$, respectively. LGP forms $\mathcal{P}_i$ by taking up to $M/3$ candidates from L_δ, up to $\lfloor M/3 \rfloor$ from L_g, and filling the remaining positions from L_ν.

LLM-Guided Semantic Selection. Given the compact candidate shortlist, LGP provides the LLM with the document represented by v_i, a compact neighborhood context $\mathcal{K}_i$ consisting of the $\min(K, |N(i)|)$ neighbors in $N(i)$ closest to v_i in embedding space, and the candidate documents in U_i. The LLM is then asked to select at most B candidates that extend rather than repeat the information already represented in $\mathcal{K}_i$, for example by providing missing supporting context, connecting v_i to a complementary concept, or extending its content toward a useful specialization or next step. The selected candidates form a proposal replacement set $\mathcal{A}_i \subseteq U_i$ (lines 16–18 in Algorithm 8).

Local Edge Replacement. Before incorporating an LLM-selected candidate, LGP applies a lightweight structural filter with threshold $g_{\min}$, retaining only candidates that provide sufficient additional two-hop coverage. Let $N'(i)$ denote the current working neighborhood of v_i. For each filtered candidate $v_c \in \mathcal{A}_i$, LGP adds it directly when $|N'(i)| < R$, and otherwise replaces a low-value original neighbor v_w (lines 19–33 in Algorithm 8).

For each original neighbor $v_j \in N'(i) \cap N(i)$, LGP measures its unique two-hop contribution and its redundancy with the remaining neighborhood:

$$\ell_i(j) := \left| N(j) \setminus \left(\left(\bigcup_{v_k \in N'(i) \setminus \{v_j\}} N(k) \right) \setminus \{v_i\} \right) \right|, \qquad \nu_i^{-j}(j) := \min_{v_k \in N'(i) \setminus \{v_j\}} D(v_j, v_k). \tag{4}$$

Here, $\ell_i(j)$ measures the unique two-hop contribution of v_j relative to the remaining neighborhood, while $\nu_i^{-j}(j)$ measures its distance to the closest remaining neighbor. LGP selects the neighbor v_w to remove by lexicographically minimizing $\big(\ell_i(j), \nu_i^{-j}(j)\big)$. This identifies neighbors with little unique local coverage and high redundancy as the lowest-value candidates for replacement.

Algorithm 10 Layer-0 HNSW pruning with LGP. Colored lines indicate operations introduced by LGP.

Input Completed Vanilla HNSW graph G, construction search width L_{build}, and LGP parameters $(B, M, K, g_{\min})$
Result Refined HNSW graph G with unchanged upper layers
Freeze all upper layers, node levels, and the entry point
Sample a random permutation σ of $\{1, \ldots, |V|\}$
for $k = 1$ to $|V|$ **do**
  $i \leftarrow \sigma(k)$
  $r_i \leftarrow |N_0(i)|$
  if $r_i = 0$ **then**
    continue
  $U_i \leftarrow$ the base-layer candidates reached by hierarchical search for v_i using width L_{build}
  $N_{\text{pre}}(i) \leftarrow N_0(i)$
  $N_0(i) \leftarrow$ LGP-LOCAL$(G, v_i, N_0(i), U_i, r_i, M, B, K, g_{\min})$
  Assert $|N_0(i)| = r_i$
  for each $v_c \in N_0(i) \setminus N_{\text{pre}}(i)$ such that $v_i \notin N_0(c)$ **do**
    $H_c \leftarrow$ SELECT-NEIGHBORS$(v_c, N_0(c) \cup \{v_i\}, |N_0(c)|, 0, \text{FALSE}, \text{FALSE})$
    if $v_i \in H_c$ **then**
      if $N_0(c) \setminus H_c \neq \varnothing$ **then**
        $v_z \leftarrow \arg\max_{v_j \in N_0(c) \setminus H_c} D(v_c, v_j)$
        $N_0(c) \leftarrow (N_0(c) \setminus \{v_z\}) \cup \{v_i\}$
Verify that all base-layer degrees and all upper layers are unchanged
return G

E.1 Instantiation for Representative ANN Indices

Algorithm 8 can be inserted into different graph-based ANN indices without changing their query-time search procedures. Algorithms 9 and 10 give the concrete instantiations for DiskANN and HNSW, respectively.

DiskANN. As shown in Algorithm 9, LGP is applied immediately after RobustPruning during the second DiskANN construction pass. The vertices visited by GreedySearch are reused as the candidate set U_i, after which LGP refines $N(i)$. Newly introduced edges follow DiskANN's usual reciprocal-insertion step. If a reciprocal insertion exceeds the degree bound, the newly inserted reciprocal edge is protected only during that immediate pruning operation. The protected pruning operation retains this edge while pruning the other candidates to satisfy the degree bound R. This protection does not persist beyond that pruning operation.

HNSW. As shown in Algorithm 10, LGP is applied as an offline pruning of layer 0 after the vanilla HNSW index has been constructed. For each node v_i, hierarchical search is reused to obtain the candidate set U_i, and LGP refines only its base-layer neighborhood $N_0(i)$ while preserving its original degree. For each newly introduced base-layer edge, HNSW's standard Select-Neighbors rule is applied to determine whether the corresponding reciprocal edge should be added. All upper layers, node levels, and the entry point remain unchanged.

F Extended Related Work

Graph-based Approximate Nearest Neighbor Search. Approximate nearest neighbor search (ANNS) has been studied through several indexing paradigms, including partition- and quantization-based methods (Jegou et al., 2011; Johnson et al., 2021) as well as graph-based methods (Malkov & Yashunin, 2020; Fu et al., 2019; Jayaram Subramanya et al., 2019), among which graph-based ANNS has emerged as one of the most effective and widely adopted approaches in practical tasks such document retrieval (Arslan et al., 2024), offering a strong balance between search quality and efficiency. These methods typically construct a proximity graph over high-dimensional vector representations and answer a query by greedily traversing the graph toward its nearest neighbors (Jayaram Subramanya et al., 2019).

Early graph-based approaches, such as FANNG (Harwood & Drummond, 2016) and EFANNA (Fu & Cai, 2016), accelerate search through neighbor expansion over approximate KNN graphs. HNSW later introduced a hierarchical navigable small-world graph that enables highly effective greedy search and has become a standard index for high-recall in-memory vector retrieval (Malkov & Yashunin, 2020). NSG further improves this line of work by explicitly optimizing graph connectivity, path length, and index size, yielding a sparse yet highly navigable graph structure (Fu et al., 2019). At larger scales, DiskANN and its Vamana graph construction algorithm (Jayaram Subramanya et al., 2019) extend graph-based ANNS to billion-scale, SSD-resident settings, demonstrating strong empirical performance in large-scale retrieval despite limited theoretical guarantees (Indyk & Xu, 2023). Building on this, FreshDiskANN supports real-time insertions and deletions (Singh et al., 2021), while Filtered-DiskANN further enables native support for filtered ANN queries (Gollapudi et al., 2023). Despite this progress, graph construction in existing ANNS methods remains fundamentally geometric and structural: edges are selected or pruned primarily based on vector proximity, graph sparsity, and navigability. As a result, these methods do not explicitly account for whether neighboring nodes are semantically redundant, complementary, or equally useful for downstream retrieval. Although stronger embedding models can improve the alignment between geometry and semantics (Chen et al., 2024; Meng et al., 2024; OpenAI, 2024), a gap still remains between geometric proximity and semantic informativeness for retrieval (Taghavi et al., 2026). Our work complements this by introducing an *offline* LLM-guided pruning paradigm on top of an existing vector-based ANN graph, where, instead of modifying query-time search, our method actively prunes local graph structure toward more semantically coherent neighborhoods while preserving the original navigation efficiency at test time.

LLM-Augmented Retrieval at Test Time. A separate line of work improves document retrieval by introducing stronger reasoning at inference time, especially for complex queries where embedding similarity is insufficient for realistic tasks that require nontrivial reasoning to identify the most relevant documents (SU et al., 2025). Existing LLM-augmented retrieval methods mainly improve retrieval quality in three directions. Some methods enhance the given queries by rewriting or reformulating the input before retrieval, with the goal of better aligning the search request with relevant documents (Ma et al., 2023; Chan et al., 2024). Another line of work strengthens the retriever itself by learning reasoning-aware retrieval representations. Representative examples include INF-X-Retriever (Yao et al., 2025), a single-stage dense retrieval framework for reasoning-intensive search that emphasizes intent-aligned retrieval, ReasonIR (Shao et al., 2025), which trains retrievers specifically for general reasoning tasks, and RaDeR (Das et al., 2025), which develops reasoning-aware dense retrieval models using synthesized supervision from reasoning trajectories. Furthermore, some approaches improve retrieval by introducing stronger reasoning at the reranking stage, where direct rerankers rescore an initial candidate set returned by a first-stage retriever (Sun et al., 2026; Zhang et al., 2025; Liu et al., 2025). For example, REARANK (Zhang et al., 2025) performs listwise reasoning-based reranking over retrieved candidates.

While they show strong empirical performance, these approaches primarily inject additional semantic reasoning at test time, which increases latency and serving cost while leaving the fundamental mismatch between geometry and semantics in the underlying graph unresolved. In contrast, our method uses LLM reasoning offline to refine local graph connectivity and address this structural misalignment at its root, thereby improving the retrieval index itself and reducing the need for further LLM assistance at test time.

Query-Time Graph Search Algorithms. Recent work has also explored improving graph-index retrieval from the query-time side by changing the search algorithm rather than the graph construction process. For example, Xu et al. (2026) builds the data structure using a cheap proxy metric but modifies the query procedure to approximate an expensive ground-truth metric, while reranker-guided search Xu & Chen (2026) alters query-time graph traversal to decide which candidates should be evaluated by a stronger reranker under a limited budget. These methods are complementary to ours: they keep the graph largely unchanged but introduce a more involved online search procedure. In contrast, our method uses LLM reasoning offline to refine the graph structure itself, without requiring any change to the query-time search algorithm. As a result, our refined graph remains compatible with standard greedy search and can also be combined with reranker-guided or bi-metric query-time search strategies.