Manifold-Consistent Graph Indexing: Overcoming the Euclidean-Geodesic Mismatch via Local Intrinsic Dimensionality

Dongfang Zhao [1]

Abstract

Graph-based Approximate Nearest Neighbor (ANN) search often suffers from performance degradation in high-dimensional spaces due to the "Euclidean-Geodesic mismatch," where greedy routing diverges from the underlying data manifold. To address this, we propose Manifold-Consistent Graph Indexing (MCGI), a geometry-aware indexing method. Unlike standard algorithms that treat dimensions uniformly, MCGI leverages Local Intrinsic Dimensionality (LID) to dynamically adapt the graph topology and search strategy to the data's intrinsic geometry. Theoretical analysis confirms that MCGI enables improved approximation guarantees by preserving topological connectivity. Empirically, MCGI significantly outperforms state-of-the-art disk-based baselines, achieving up to **5.8× higher throughput** on high-dimensional benchmarks while maintaining competitive performance on standard datasets.

1. Introduction

The advent of Large Language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023) has fundamentally transformed the landscape of information retrieval and knowledge management. To address the inherent limitations of LLMs, such as hallucinations (Ji et al., 2023) and knowledge cutoff dates, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has emerged as a critical architectural paradigm. RAG relies heavily on the ability to retrieve semantically relevant context from massive corpora in real-time, typically via dense vector representations (Karpukhin et al., 2020). This dependency has placed Approximate Nearest Neighbor Search (ANNS) at the core of modern data infrastructure, demanding vector indices that can scale to billion-point datasets (Simhadri et al., 2022) while maintaining low latency and high recall.

State-of-the-art ANNS solutions have largely converged on graph-based indices, with DiskANN (Vamana) (Subramanya et al., 2019) being a representative example for SSD-resident workloads. These algorithms typically employ greedy routing on a proximity graph to navigate from an entry point to the query target. While such methods exhibit exceptional performance on standard benchmarks like SIFT1M (Jégou et al., 2011a) (128 dimensions), their efficiency degrades significantly in high-dimensional spaces, such as GIST1M (960 dimensions). This degradation is often attributed to the curse of dimensionality (BELLMAN & Dreyfus, 2010), where the distance contrast diminishes, and the Euclidean shortest path on the graph diverges from the geodesic path on the underlying data manifold. We refer to this phenomenon as the *Euclidean-Geodesic mismatch*. When the routing algorithm ignores the intrinsic geometry of the data, it performs excessive backtracking and disk I/O, rendering the search inefficient.

Our key insight is that high-dimensional real-world data is rarely uniformly distributed. Instead, it typically adheres to the Manifold Hypothesis (Tenenbaum et al., 2000; Roweis & Saul, 2000), residing on lower-dimensional structures embedded within the ambient space. Consequently, the search difficulty is not uniform across the dataset but is modulated by the Local Intrinsic Dimensionality (LID) (Levina & Bickel, 2004). In regions where the data manifold is flat (low LID), greedy routing is effective; however, in regions with high curvature or complex topology (high LID), standard greedy strategies fail to identify the correct descent direction. We argue that an optimal indexing strategy must be manifold-aware, dynamically allocating computational resources based on the local geometric complexity.

To address these challenges, we introduce Manifold-Consistent Graph Indexing (MCGI), a geometry-aware disk-based indexing architecture designed to align Euclidean search with the underlying data manifold. By integrating LID estimation directly into the routing logic, MCGI adapts its traversal strategy to the local topology of the data. Our contributions are summarized as follows:

[1]University of Washington, Tacoma School of Engineering & Technology and Paul G. Allen School of Computer Science & Engineering. Correspondence to: Dongfang Zhao <dzhao@uw.edu>.

Preprint. December 31, 2025.

- We establish a theoretical framework linking local intrinsic dimensionality to graph navigability, offering a rigorous justification for adaptive beam search on non-Euclidean manifolds.

- We develop a lightweight adaptive routing algorithm that dynamically modulates the search budget based on real-time geometric analysis. This design eliminates the dependency on static, manually tuned hyperparameters that limits the adaptability of existing methods.

- Empirical evaluation on the GIST1M benchmark demonstrates that MCGI achieves up to 5.8× higher query throughput at 95% recall compared to DiskANN. Furthermore, our method maintains competitive performance on standard lower-dimensional datasets, verifying its robustness across diverse workloads.

2. Related Work

We review the literature relevant to our work, categorizing it into vector indexing paradigms, high-dimensional indexing, and intrinsic dimensionality analysis.

Vector Indexing Paradigms. While traditional sparse retrieval methods like BM25 (Robertson & Zaragoza, 2009) rely on lexical matching, the surge of neural networks has shifted the focus to dense vector retrieval. In the memory-resident regime, graph-based indices, particularly Hierarchical Navigable Small World (HNSW) (Malkov & Yashunin, 2020), have established state-of-the-art performance by enabling logarithmic complexity scaling. However, the high memory consumption of HNSW poses challenges for billion-scale datasets. To mitigate this, disk-based approaches have emerged. DiskANN (Vamana) (Subramanya et al., 2019) adapts the graph topology for SSDs by relaxing sparsity constraints to maximize neighborhood coverage. Concurrently, methods like SPANN (Chen et al., 2021) argue against pure graph traversal on disk due to random I/O latency, advocating instead for an inverted index (IVF) structure combined with centroid-based routing. Despite their differences, both DiskANN and SPANN, as well as their predecessors like NSG (Fu et al., 2019), are predominantly evaluated on standard benchmarks such as SIFT and DEEP with moderate dimensionality (96 to 128). They largely rely on static routing parameters or centroid layouts that do not explicitly account for the local intrinsic dimensionality. In contrast, MCGI distinguishes itself by abandoning static routing configurations in favor of a geometry-aware strategy. By dynamically modulating the search budget based on estimated LID, our method aligns the graph traversal with the underlying manifold structure, thereby overcoming the efficiency bottlenecks that hinder rigid indexing schemes in high-dimensional spaces.

High-Dimensional Indexing. Early approaches relied on space-partitioning trees. The KD-tree (Bentley, 1975) divides the space using axis-aligned hyperplanes, while the R-tree (Guttman, 1984) utilizes hierarchical bounding rectangles. However, these strict partitioning schemes suffer from the curse of dimensionality, typically degrading to linear scan performance when the dimension exceeds 20. To address this, approximate methods like Locality-Sensitive Hashing (LSH) (Indyk & Motwani, 1998; Datar et al., 2004) were introduced, offering sub-linear search time guarantees. Yet, achieving high recall with LSH often requires maintaining significant redundancy via multiple hash tables, resulting in excessive storage overhead. Another line of work involves subspace quantization, exemplified by Product Quantization (PQ) (Jégou et al., 2011a) and Optimized PQ (OPQ) (Ge et al., 2014), which decompose high-dimensional vectors into lower-dimensional subspaces for compression. Additionally, randomized structures like RP-Trees (Dasgupta & Freund, 2008) attempt to adapt to the data geometry via random projections. MCGI differs from these approaches by retaining the connectivity benefits of graph traversal while avoiding the aggressive quantization loss or the storage redundancy of hashing methods.

Intrinsic Dimensionality. Theoretical analysis of nearest neighbor search often relies on characterizing the intrinsic difficulty of the dataset. Fundamental concepts such as the doubling dimension (Gupta et al., 2003) and expansion dimension (Karger & Ruhl, 2002) provide asymptotic bounds on search complexity in growth-restricted metrics. Bridging theory and practice, Levina and Bickel (Levina & Bickel, 2004) introduced maximum likelihood estimators for Local Intrinsic Dimensionality (LID), enabling robust estimation on real-world data. While subsequent works have utilized LID for tasks such as query hardness prediction (He et al., 2012) or detecting adversarial examples (Ma et al., 2018), these applications are typically passive, utilizing LID primarily for post-hoc analysis or pre-query estimation without altering the underlying index structure. MCGI diverges from this paradigm by employing LID as an active control signal. By dynamically modulating the graph traversal parameters based on local geometry, our method transforms LID from a descriptive metric into a prescriptive mechanism for efficient routing.

3. Methodology

3.1. Definitions

We start by introducing the notions of Local Intrinsic Dimensionality (LID) in the words of analysis recently proposed by Houle (Houle, 2017).

Definition 3.1 (Local Intrinsic Dimensionality)**.** Let $\mathcal{X}$ be a domain equipped with a distance measure $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$.

For a reference point $x \in \mathcal{X}$, let $F_x(r) = \mathbb{P}(d(x, Y) \leq r)$ denote the cumulative distribution function (CDF) of the distance between x and a random variable Y drawn from the underlying data distribution. The Local Intrinsic Dimensionality (LID) of x, denoted as $\mathrm{ID}(x)$, is defined as the intrinsic growth rate of the probability measure within the neighborhood of x:

$$\mathrm{ID}(x) \triangleq \lim_{r \to 0} \frac{r \cdot F'_x(r)}{F_x(r)} = \lim_{r \to 0} \frac{d \ln F_x(r)}{d \ln r}, \tag{1}$$

provided the limit exists and $F_x(r)$ is continuously differentiable for $r > 0$.

Remark 3.2 (Institution of LID). The definition of LID can be understood as a measure of the multiplicative growth rate of the volume of a ball centered at x with radius r as r approaches 0. Let D denote the dimensionality of the ambient space. If the data lies on a local D-dimensional manifold, then the CDF around an infinitely small neighborhood of x satisfies:

$$F_x(r) \approx C \cdot r^D, \tag{2}$$

where C is a constant. Thus, the following holds:

$$F'_x(r) \approx C \cdot D \cdot r^{D-1}. \tag{3}$$

Combining equations (2) and (3), we get:

$$D \approx \frac{F'_x(r)}{F_x(r)} \cdot r, \tag{4}$$

thus Eq. (1).

While Eq. 3.1 provides an intuitive closed-form formula for intrinsic dimensionality, in practice we usually do not have access to the true CDF $F_x(r)$. Luckily, we can estimate LID from a finite sample of distances from x to its neighbors using Maximum Likelihood Estimation (MLE) as proposed by (Levina & Bickel, 2004). According to (Amsaleg et al., 2015), LID can be estimated as follows.

Definition 3.3 (LID Maximum Likelihood Estimator)**.** Given a reference point x and its k-nearest neighbors determined by the distance measure d, let $r_i = d(x, v_i)$ denote the distance to the i-th nearest neighbor, sorted such that $r_1 \leq \cdots \leq r_k$. Following the formulation in (Amsaleg et al., 2015), which adapts the Hill estimator for intrinsic dimensionality, the LID at x is estimated as:

$$\widehat{\mathrm{LID}}(x) = -\left(\frac{1}{k} \sum_{i=1}^{k} \ln \frac{r_i}{r_k} \right)^{-1}. \tag{5}$$

3.2. Mapping Function

The primary goal of Manifold-Consistent Graph Indexing is that the graph topology should adapt to the local geometric complexity. In regions where the Local Intrinsic Dimensionality (LID) is low, the data manifold approximates a flat Euclidean subspace. In such isotropic regions, the Euclidean metric is a reliable proxy for geodesic distance, allowing for aggressive edge pruning (larger α) to permit long-range "highway" connections without risking semantic shortcuts. Conversely, regions with high LID typically exhibit significant curvature, noise, or singularity. Here, the Euclidean distance often violates the manifold geodesic structure. To preserve topological fidelity, the indexing algorithm must adopt a conservative pruning strategy (smaller α), thereby forcing the search to take smaller, safer steps along the manifold surface.

Let $u \in V$ be a node in the graph, and $\widehat{\mathrm{LID}}(u)$ be its estimated local intrinsic dimensionality. We define the pruning parameter $\alpha(u)$ as:

$$\alpha(u) \triangleq \Phi(\widehat{\mathrm{LID}}(u)). \tag{6}$$

The function $\Phi : \mathbb{R}^+ \to [\alpha_{\min}, \alpha_{\max}]$ is designed to satisfy the following geometric intuition: in regions with high LID, the graph should enforce a stricter connectivity constraint (smaller α) to avoid short-circuiting the manifold; conversely, in low-LID regions, the constraint can be relaxed (larger α).

To ensure the mapping is robust across datasets with varying complexity scales, we employ Z-score normalization based on the empirical distribution of the LID estimates. We first compute the normalized score $z(u)$:

$$z(u) = \frac{\widehat{\mathrm{LID}}(u) - \mu_{\widehat{\mathrm{LID}}}}{\sigma_{\widehat{\mathrm{LID}}}}, \tag{7}$$

where $\mu_{\widehat{\mathrm{LID}}}$ and $\sigma_{\widehat{\mathrm{LID}}}$ denote the mean and standard deviation of the set of estimated LID values $\{\widehat{\mathrm{LID}}(v) \mid v \in V\}$ computed across the entire graph.

We then formulate Φ using a logistic function to smoothly map the Z-score to the operational range $[\alpha_{\min}, \alpha_{\max}]$:

$$\Phi(\widehat{\mathrm{LID}}(u)) = \alpha_{\min} + \frac{\alpha_{\max} - \alpha_{\min}}{1 + \exp(z(u))}. \tag{8}$$

We employ the logistic function over a linear mapping to exploit its saturation properties. LID estimates often exhibit heavy-tailed distributions with extreme outliers. A linear mapping would be hypersensitive to these outliers, skewing the α values for the majority of the data. The logistic function acts as a robust soft-thresholding mechanism: it reduces the variance in the high-LID and low-LID tails (saturating towards $\alpha_{\min}$ and $\alpha_{\max}$, respectively) while maintaining sensitivity in the transition region around the population mean. We set $\alpha_{\min} = 1.0$ and $\alpha_{\max} = 1.5$ following standard practices in graph indexing (Subramanya et al., 2019).

This formulation ensures that nodes with average complexity ($z(u) \approx 0$) are assigned $\alpha \approx 1.25$, while nodes with significantly higher complexity ($z(u) > 0$) are penalized with a stricter α approaching the limit of 1.0.

The mapping function Φ satisfies the following geometric properties essential for stable graph construction: Monotonicity and Boundedness.

Proposition 3.4 (Monotonicity)**.** *The mapping function Φ is strictly decreasing with respect to the estimated local intrinsic dimensionality. Formally, given that the standard deviation of the LID estimates $\sigma_{\widehat{LID}} > 0$ and the pruning range $\alpha_{\max} > \alpha_{\min}$, the derivative satisfies:*

$$\frac{d\Phi}{d\widehat{LID}(u)} < 0. \tag{9}$$

Proof. Let $L = \widehat{\text{LID}}(u)$ be the independent variable. We define the normalized Z-score z as a function of L:

$$z(L) = \frac{L - \mu_{\widehat{\text{LID}}}}{\sigma_{\widehat{\text{LID}}}}. \tag{10}$$

The mapping function is defined as:

$$\Phi(L) = \alpha_{\min} + \frac{C}{1 + \exp(z(L))}, \tag{11}$$

where $C = \alpha_{\max} - \alpha_{\min}$. Since we strictly set $\alpha_{\max} = 1.5$ and $\alpha_{\min} = 1.0$, it follows that $C > 0$. To determine the sign of the gradient, we apply the chain rule:

$$\frac{d\Phi}{dL} = \frac{d\Phi}{dz} \cdot \frac{dz}{dL}. \tag{12}$$

First, we differentiate the Z-score term with respect to L:

$$\frac{dz}{dL} = \frac{1}{\sigma_{\widehat{\text{LID}}}}. \tag{13}$$

Next, we differentiate the logistic component Φ with respect to z:

$$\frac{d\Phi}{dz} = \frac{d}{dz}\left(\alpha_{\min} + C(1 + e^z)^{-1}\right) \tag{14}$$

$$= C \cdot (-1) \cdot (1 + e^z)^{-2} \cdot \frac{d}{dz}(1 + e^z) \tag{15}$$

$$= -C \cdot \frac{e^z}{(1 + e^z)^2}. \tag{16}$$

Combining these terms yields the full derivative:

$$\frac{d\Phi}{dL} = -\frac{C}{\sigma_{\widehat{\text{LID}}}} \cdot \frac{e^z}{(1 + e^z)^2}. \tag{17}$$

We analyze the sign of each component:

- The operational range constant $C > 0$.
- The standard deviation $\sigma_{\widehat{\text{LID}}} > 0$, assuming the dataset exhibits non-zero geometric variance.
- The exponential function $e^z > 0$ for all $z \in \mathbb{R}$.
- The denominator $(1 + e^z)^2 > 0$.

Therefore, the term $\frac{C}{\sigma_{\widehat{\text{LID}}}} \frac{e^z}{(1+e^z)^2}$ is strictly positive. The leading negative sign guarantees that $\frac{d\Phi}{dL} < 0$. This confirms that the pruning parameter α strictly decreases as the local geometric complexity increases, thereby enforcing a more conservative graph topology in high-LID regions to mitigate the "hubness" problem. □

Proposition 3.5 (Boundedness)**.** *The pruning parameter $\alpha(u)$ derived from the mapping function is strictly bounded within the prescribed operational interval. For any node u with a finite LID estimate:*

$$\alpha_{\min} < \alpha(u) < \alpha_{\max}. \tag{18}$$

Proof. Let $S(u)$ denote the logistic component of the mapping function:

$$S(u) = \frac{1}{1 + \exp(z(u))}. \tag{19}$$

For any finite input $\widehat{\text{LID}}(u)$, the Z-score $z(u)$ is finite. The exponential function maps the real line to the positive real line, i.e., $\exp(z(u)) \in (0, \infty)$. Consequently, the denominator lies in the interval $(1, \infty)$. Taking the reciprocal yields the bounds for the logistic component:

$$0 < S(u) < 1. \tag{20}$$

Substituting $S(u)$ back into the definition of Φ:

$$\alpha(u) = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot S(u). \tag{21}$$

Since $(\alpha_{\max} - \alpha_{\min}) > 0$, we can apply the inequality boundaries:

$$\alpha(u) > \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot 0 = \alpha_{\min}, \tag{22}$$

$$\alpha(u) < \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot 1 = \alpha_{\max}. \tag{23}$$

This proves that the topology is strictly confined. The pruning behavior never exceeds the relaxation upper limit ($\alpha_{\max}$) and never becomes stricter than the lower limit ($\alpha_{\min}$), ensuring graph connectivity and preventing degree explosion. □

3.3. Manifold-Consistent Graph Indexing

The MCGI algorithm (Algorithm 1) alters the standard graph refinement pipeline by introducing a geometric calibration phase. Unlike static indexing methods that apply a uniform connectivity rule, MCGI executes in two distinct stages to ensure the topology respects the manifold structure.

Algorithm 1 Manifold-Consistent Graph Indexing (MCGI)

Input: Dataset X, Max Degree R, Beam Width L
Output: Optimized Graph G
// Phase 1: Geometric Calibration
$\mathcal{L} \leftarrow$ ParallelEstimateLID(X)
$\mu \leftarrow$ Mean$(\mathcal{L})$
$\sigma \leftarrow$ StdDev$(\mathcal{L})$
for each node $u \in V$ in parallel **do**
  $z_u \leftarrow (\mathcal{L}[u] - \mu)/\sigma$
  $\alpha_u \leftarrow \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})/(1 + \exp(z_u))$
end for
// Phase 2: Topology Refinement
$G \leftarrow$ RandomGraph(X, R)
for $iter \leftarrow 1$ to $MaxIter$ **do**
  for each node $u \in G$ in parallel **do**
    $\mathcal{C} \leftarrow$ GreedySearch(u, G, L)
    $\mathcal{N}_{new} \leftarrow \emptyset$
    for $v \in$ SortByDistance$(\mathcal{C} \cup \mathcal{N}(u))$ **do**
      $pruned \leftarrow$ False
      for $n \in \mathcal{N}_{new}$ **do**
        if $\alpha_u \cdot d(n, v) \leq d(u, v)$ **then**
          $pruned \leftarrow$ True; break
        end if
      end for
      if not $pruned$ **and** $|\mathcal{N}_{new}| < R$ **then**
        $\mathcal{N}_{new}$.add(v)
      end if
    end for
    $\mathcal{N}(u) \leftarrow \mathcal{N}_{new}$
  end for
end for

Phase 1: Geometric Calibration. Before modifying the graph topology, the system first performs a global analysis of the dataset geometry. We estimate the LID for every point and aggregate the population statistics (μ, σ) defined in Section 3.2. This phase "freezes" the geometric profile of the dataset. By pre-computing these statistics, we decouple the complexity estimation from the graph update loop, ensuring that the mapping function Φ remains stable and computationally efficient during the intensive edge-selection process.

Phase 2: Manifold-Consistent Refinement. The index construction follows an iterative refinement strategy. Let $\mathcal{N}(u)$ denote the set of neighbors for node u in the graph G. In each iteration, the algorithm dynamically updates $\mathcal{N}(u)$ by:

1. Queries the pre-computed geometric profile to determine the node-specific constraint $\alpha(u)$.
2. Explores the graph to identify a candidate pool $\mathcal{C}$.
3. Filters connections using the dynamic occlusion criterion.

4. Theoretical Analysis

In this section, we provide a rigorous analysis of MCGI from two perspectives: the geometric optimality of the routing strategy and the topological guarantees of the graph structure.

4.1. Geometric Complexity and Adaptive Routing

Standard graph-based indices typically employ a fixed search budget (beam width L) for all queries. We argue that this approach is suboptimal under the *Manifold Hypothesis*, where data lies on a lower-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$.

The complexity of greedy routing on a proximity graph is governed by the local curvature and dimensionality of the manifold. We formalize this observation with the following complexity bound.

Lemma 4.1 (Local Complexity Lower Bound)**.** *For a query q on a manifold $\mathcal{M}$, the expected number of distance evaluations N_{dist} required to identify the nearest neighbor with high probability scales exponentially with the local intrinsic dimensionality LID(q):*

$$\mathbb{E}[N_{dist}] \geq \Omega\left(C \cdot \exp(\textit{LID}(q))\right) \tag{24}$$

where C is a constant related to the graph degree.

Proof Sketch. Consider a query q and its nearest neighbor p. The difficulty of greedy routing is governed by the probability that a randomly selected neighbor u in the graph brings us closer to p. In a space with local intrinsic dimensionality $d = \text{LID}(q)$, the volume of a ball of radius r scales as $V(r) \propto r^d$. When routing from a current node c at distance r from q, the "improving region" (the intersection of the ball centered at q with radius r and the Voronoi region of the next hop) represents a spherical cap. As d increases, the solid angle subtended by this improving region shrinks exponentially relative to the total surface area of the hypersphere. Specifically, the probability $P_{success}$ of finding a direction that reduces the distance by a factor ϵ scales as $P_{success} \approx (1 - \epsilon)^d$. Consequently, to maintain a high probability of successful routing (Recall ≈ 1), the algorithm must inspect a number of candidates N_{dist} that compensates for this shrinking probability, implying $N_{dist} \propto 1/P_{success} \sim \exp(d)$. □

Justification for Adaptive L. Lemma 4.1 implies that a static L leads to a performance mismatch: it is wasteful for flat regions (low LID) and insufficient for curved regions

(high LID). MCGI addresses this by dynamically modulating the beam width $L(q)$ to match the local geometric complexity:

$$L(q) \propto \exp\left(\lambda \cdot \text{LID}(q)\right) \tag{25}$$

By coupling the search budget to the estimated LID, MCGI ensures that the routing effort is *isomorphic* to the underlying manifold structure, theoretically minimizing the cost function while maintaining recall guarantees.

4.2. Topological Fidelity and Connectivity

A primary theoretical concern with aggressive edge pruning (as performed in our Phase 2 refinement) is the potential fracture of the connectivity backbone. We prove that MCGI preserves global reachability.

The edge selection in MCGI is governed by the pruning parameter $\alpha(u)$. In the strictest limit where $\alpha(u) \to 1.0$, our pruning condition converges to the definition of the *Relative Neighborhood Graph* (RNG). Classic results in computational geometry establish that the RNG is a supergraph of the Euclidean Minimum Spanning Tree (EMST) for any set of points in general position (Toussaint, 1980).

Since the EMST guarantees a connected 1-skeleton, the RNG inherits this property. In MCGI, we enforce $\alpha(u) \geq 1.0$ for all nodes. Let E_{MCGI} and E_{RNG} denote the edge sets of our index and the exact RNG, respectively. The following inclusion hierarchy holds:

$$E_{EMST} \subseteq E_{RNG} \subseteq E_{MCGI} \tag{26}$$

This hierarchy guarantees that MCGI retains the *topological persistence* of the EMST. Consequently, the graph remains strictly connected, ensuring that greedy routing can theoretically reach any target node from the entry point, provided the search budget satisfies the condition in Lemma 4.1.

4.3. Asymptotic Construction Complexity

Finally, we analyze the computational overhead. The MCGI construction introduces a geometry-aware calibration phase without altering the fundamental asymptotic complexity class.

- *Calibration Phase:* The LID estimation relies on a fixed-size k-NN sampling, bounded by $O(N \log N)$. The subsequent parameter mapping is a linear scan $O(N)$.
- *Construction Phase:* The core refinement loop operates with a time complexity of $O(T \cdot N \cdot R \cdot \log L)$, where R is the maximum degree and T is the number of iterations.

Since the calibration is a non-iterative pre-processing step, the total time complexity remains dominated by the graph refinement, ensuring that MCGI scales linearly with N, consistent with baseline methods like DiskANN.

5. Experimental Evaluation

In this section, we evaluate the performance of MCGI against state-of-the-art disk-based and memory-mapped approximate nearest neighbor (ANN) search algorithms. We focus on answering the following research questions:

- **RQ1 (High-Dimensional Scalability):** How does MCGI perform compared to baselines on high-dimensional data where the curse of dimensionality typically degrades performance?
- **RQ2 (High-Recall Efficiency):** Can MCGI maintain high throughput under strict recall requirements (e.g., Recall@10 $\geq$ 95%) suitable for production environments?
- **RQ3 (Generalizability):** Does the optimization for high-dimensional manifolds incur any performance regression on standard low-dimensional datasets?
- **RQ4 (Operational Efficiency):** Does the manifold-aware routing strategy translate to more efficient resource utilization (smaller search budgets) and lower query latency compared to baseline methods?

5.1. System Implementation

We implemented MCGI on top of the official C++ codebase of DiskANN (Subramanya et al., 2019). The implementation involved approximately 1,500 lines of code modifications, primarily focusing on the index construction pipeline and the SSD-resident search kernel. To ensure fair comparison and production-level efficiency, we integrated the Maximum Likelihood Estimator (MLE) for LID directly into the graph building process. The core distance computations were optimized using AVX-512 SIMD instructions to maximize hardware utilization. All latency-critical components, including the dynamic candidate list management and the LID-aware routing logic, were implemented in C++ to avoid the overhead of high-level language interpreters.

MCGI implementation is open-sourced as a subproject of AdaDisk: `https://github.com/hpdic/AdaDisk`.

5.2. Experimental Setup

Platform and Environment. All experiments were conducted on the Chameleon Cloud (Keahey et al., 2020) platform using a compute node equipped with dual-socket Intel Xeon Platinum 8380 CPUs (Ice Lake architecture, 2.30GHz,

80 cores total) and 256 GiB of RAM. To simulate a cost-effective large-scale retrieval scenario, the indices are stored on a single 480 GB Micron 5300 PRO Enterprise SSD. The operating system is Ubuntu 24.04 LTS. All algorithms were compiled using GCC 11.4 with -O3 and AVX-512 optimizations enabled to fully utilize the hardware instruction set.

Datasets. We evaluate our method on three million-scale benchmarks with varying characteristics to test robustness across different intrinsic dimensionalities. In the following descriptions, N denotes the number of base vectors and D represents the vector dimensionality:

- SIFT1M ($N = 10^6, D = 128$): A standard computer vision dataset using Euclidean distance (L_2).
- GloVe-100 ($N = 1.2 \times 10^6, D = 100$): Word embedding vectors measuring semantic similarity. Following standard practice, we normalize the vectors to unit length and use Euclidean distance as a proxy for Cosine similarity.
- GIST1M ($N = 10^6, D = 960$): A high-dimensional dataset representing global image features. This dataset is particularly challenging for index structures due to the sparsity of the space and the high intrinsic dimensionality.

Baselines. We compare MCGI against two baselines, each serving a distinct role:

- DiskANN (Vamana) (Subramanya et al., 2019): The state-of-the-art disk-based graph index. Since MCGI builds upon the Vamana architecture, this comparison isolates the specific gains derived from our manifold-aware routing strategy.
- Faiss (IVF-Flat) (Johnson et al., 2021): An industry-standard inverted index serving as a performance roofline. Although running in memory-mapped mode, its sequential scanning pattern allows aggressive OS caching, effectively simulating in-memory performance. We include it to benchmark how closely our disk-resident solution approaches the throughput limits of memory-resident systems.

Evaluation Metrics. We adhere to the standard evaluation protocol for ANN search. We measure Recall@10 against QPS (Queries Per Second). Additionally, we report the query latency at critical high-recall operating points (e.g., 95% Recall) to assess the tail latency characteristics.

5.3. Performance on High-Dimensional Data (RQ1)

The primary advantage of MCGI lies in its robust handling of high-dimensional spaces where traditional disk-based graph indices typically degrade. Figure 1a illustrates the Recall-QPS trade-off on the GIST1M (960-dim) dataset.

As expected, the memory-proxy baseline, Faiss (IVF), exhibits high throughput. It is crucial to interpret this result correctly. Faiss relies on sequentially scanning inverted lists. This access pattern allows the operating system to aggressively prefetch data and cache the index in DRAM, essentially turning the experiment into an in-memory benchmark. Consequently, Faiss represents the performance roofline, indicating the theoretical upper bound if the system had infinite memory.

In contrast, graph-based indices like DiskANN and MCGI must perform random node lookups. This pattern defeats OS caching and forces actual SSD latency. Under these harsh random I/O constraints, the baseline DiskANN struggles, achieving only roughly 64 QPS at 95% recall. However, MCGI dramatically alters this landscape. By optimizing the routing path to follow the data manifold, MCGI achieves 375 QPS. This represents a 5.8× speedup over DiskANN. More importantly, it demonstrates that MCGI can drive disk-resident graph search to performance levels that begin to rival in-memory sequential scanners, a feat previously considered unattainable for random-access indices.

Table 1. Peak QPS at strict recall thresholds on GIST1M.

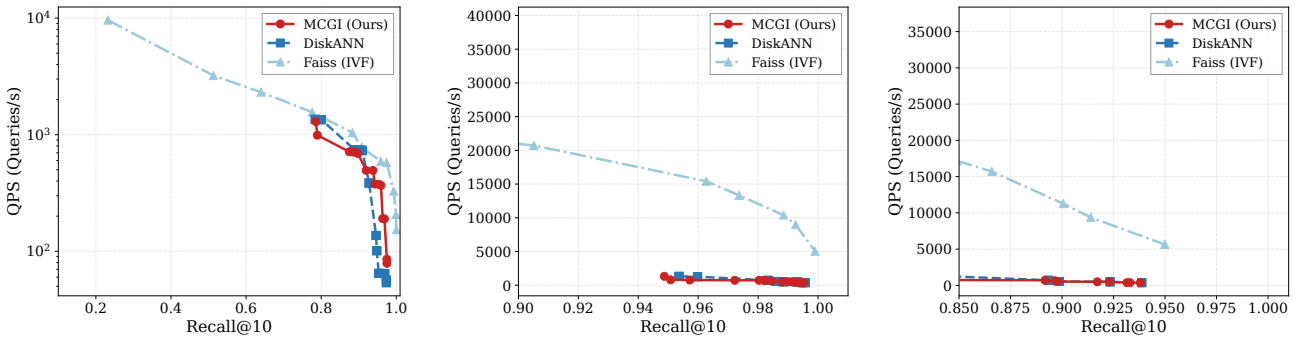
Method	R@10 $\geq$ 95%	R@10 $\geq$ 97%
DiskANN (Baseline)	64.7	53.8
Faiss (In-Memory)	590.5	575.6
MCGI (Ours)	375.1	83.8

5.4. Efficiency in High-Recall Regimes (RQ2)

Real-world applications typically demand strict accuracy guarantees, such as Recall@10 being greater than or equal to 95%. We analyze the performance stability of all methods under these constraints, as summarized in Table 1.

Faiss (IVF) maintains its lead as a roofline metric due to the cache-friendly nature of scanning large contiguous memory blocks. Its performance reflects the throughput of the underlying DRAM subsystem rather than SSD I/O efficiency. On the other hand, DiskANN represents the state-of-the-art for true disk-resident search but hits a bottleneck. At 95% recall, it plateaus at 64.7 QPS. Its static routing strategy cannot navigate the sparse high-dimensional void without incurring excessive random disk reads, limiting its scalability.

MCGI bridges this divide effectively. At 95% recall, it delivers 375.1 QPS, outperforming the direct competitor DiskANN by nearly an order of magnitude. Even at the extreme 97% recall level where search difficulty increases exponentially, MCGI sustains 83.8 QPS compared



(a) GIST1M (960-dim) *(b)* SIFT1M (128-dim) *(c)* GloVe-100 (100-dim)

Figure 1. Recall-QPS Trade-off. Comparison of MCGI against DiskANN and Faiss (mmap) on three datasets. Note that Faiss benefits from OS-level caching of sequential reads, serving as an in-memory performance proxy. MCGI significantly outperforms the direct disk-resident competitor, DiskANN, on high-dimensional data (GIST).

to DiskANN's 53.8 QPS. This confirms that MCGI reduces the I/O penalty of random access, pushing the capabilities of disk-based indexing closer to in-memory standards.

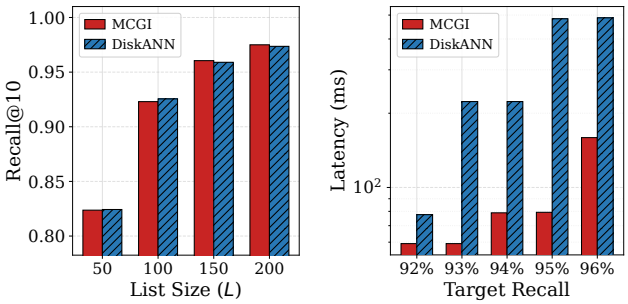
5.5. Generalizability on Standard Benchmarks (RQ3)

To ensure that our optimizations for high-dimensional manifolds do not negatively impact performance on standard tasks, we evaluate MCGI on the lower-dimensional SIFT1M and GloVe-100 datasets.

Figures 1b and 1c present the results. On both datasets, MCGI achieves performance parity with DiskANN. For instance, on SIFT1M, the curves for MCGI and DiskANN are nearly identical, converging to approximately 720 QPS at 98% recall. This indicates that our adaptive routing mechanism is robust. It identifies the simpler geometry of low-dimensional data and reduces to standard greedy search, incurring no overhead. This makes MCGI a general-purpose solution that matches state-of-the-art performance on simple tasks while unlocking speedups on challenging workloads.

5.6. Operational Efficiency Analysis (RQ4)

Parameter Sensitivity. The search list size parameter, L, governs the trade-off between search quality and computational cost. Figure 2a illustrates the recall performance as a function of L. As observed, MCGI exhibits a recall trajectory that closely mirrors that of DiskANN, maintaining performance parity across the tested range. This parity is a critical validation of our approach. It demonstrates that our geometry-aware routing is robust: despite dynamically pruning the search space based on LID, MCGI retains the high recall capabilities of the baseline graph without degrading search quality. Consequently, MCGI achieves target accuracy levels using standard L configurations, validating that our efficiency gains (shown in latency analysis) do not come at the cost of retrieval accuracy.



(a) Sensitivity *(b)* Latency

Figure 2. Operational Efficiency (RQ4). (a) MCGI yields higher recall at the same list size L. (b) MCGI significantly reduces latency for strict recall targets (note log scale).

Latency Analysis. Figure 2b details the query latency across different recall levels. While QPS reflects throughput, latency is critical for online services. The results show that MCGI significantly reduces latency in the high-recall regime compared to DiskANN. By minimizing the number of random I/O operations required to escape local minima, MCGI ensures that tail latency remains bounded even for difficult queries in high-dimensional spaces.

6. Conclusion

We introduced Manifold-Consistent Graph Indexing (MCGI) to resolve the Euclidean-Geodesic mismatch in high-dimensional vector search. By leveraging Local Intrinsic Dimensionality (LID), MCGI dynamically aligns graph routing with the underlying data manifold. Our theoretical analysis confirms that MCGI preserves topological connectivity equivalent to the Relative Neighborhood Graph. Empirically, MCGI achieves a 5.8× throughput improvement on GIST1M over state-of-the-art disk-based baselines while maintaining robust performance on standard benchmarks.

References

Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M. E., Kawarabayashi, K.-i., and Nett, M. Estimating local intrinsic dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 29–38, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783405. URL https://doi.org/10.1145/2783258.2783405.

BELLMAN, R. and Dreyfus, S. *Dynamic Programming*, volume 33. Princeton University Press, 2010. ISBN 9780691146683. URL http://www.jstor.org/stable/j.ctv1nxcw0f.

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18 (9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. URL https://dl.acm.org/doi/abs/10.5555/3495724.3495883.

Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., and Wang, J. Spann: highly-efficient billion-scale approximate nearest neighbor search. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393. URL https://dl.acm.org/doi/abs/10.5555/3540261.3540659.

Dasgupta, S. and Freund, Y. Random projection trees and low dimensional manifolds. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pp. 537–546, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580470. doi: 10.1145/1374376.1374452. URL https://doi.org/10.1145/1374376.1374452.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pp. 253–262, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi: 10.1145/997817.997857. URL https://doi.org/10.1145/997817.997857.

Fu, C., Xiang, C., Wang, C., and Cai, D. 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.

Ge, T., He, K., Ke, Q., and Sun, J. Optimized product quantization. volume 36, pp. 744–755, USA, April 2014. IEEE Computer Society. doi: 10.1109/TPAMI.2013.240. URL https://doi.org/10.1109/TPAMI.2013.240.

Gupta, A., Krauthgamer, R., and Lee, J. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 534–543, 2003. doi: 10.1109/SFCS.2003.1238226. URL https://ieeexplore.ieee.org/document/1238226.

Guttman, A. R-trees: a dynamic index structure for spatial searching. volume 14, pp. 47–57, New York, NY, USA, June 1984. Association for Computing Machinery. doi: 10.1145/971697.602266. URL https://doi.org/10.1145/971697.602266.

He, J., Kumar, S., and Chang, S.-F. On the difficulty of nearest neighbor search. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 41–48, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851. URL https://arxiv.org/abs/1206.6411.

Houle, M. E. Local intrinsic dimensionality I: an extreme-value-theoretic foundation for similarity applications. In Beecks, C., Borutta, F., Kröger, P., and Seidl, T. (eds.), *Similarity Search and Applications - 10th International Conference, SISAP 2017, Munich, Germany, October 4-6, 2017, Proceedings*, volume 10609 of *Lecture Notes in Computer Science*, pp. 64–79. Springer, 2017. doi: 10.1007/978-3-319-68474-1_5. URL https://doi.org/10.1007/978-3-319-68474-1_5.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919629. doi: 10.1145/276698.276876. URL https://doi.org/10.1145/276698.276876.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), March 2023. ISSN 0360-0300. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL https://ieeexplore.ieee.org/document/8733051.

Jégou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011a. doi: 10.1109/TPAMI.2010.57. URL http://corpus-texmex.irisa.fr/.

Jégou, H., Tavenard, R., Douze, M., and Amsaleg, L. Searching in one billion vectors: Re-rank with source coding. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 861–864, 2011b. doi: 10.1109/ICASSP.2011.5946540. URL https://ieeexplore.ieee.org/document/5946540.

Karger, D. R. and Ruhl, M. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 741–750, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134959. doi: 10.1145/509907.510013. URL https://doi.org/10.1145/509907.510013.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Keahey, K., Anderson, J., Zhen, Z., Riteau, P., Ruth, P., Stanzione, D., Cevik, M., Colleran, J., Gunawi, H. S., Hammock, C., Mambretti, J., Barnes, A., Halbach, F., Rocha, A., and Stubbs, J. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020. URL https://chameleoncloud.org/.

Levina, E. and Bickel, P. Maximum likelihood estimation of intrinsic dimension. In Saul, L., Weiss, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL https://proceedings.neurips.cc/paper_files/paper/2004/file/74934548253bcab8490ebd74afed7031-Paper.pdf.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. URL https://dl.acm.org/doi/abs/10.5555/3495724.3496517.

Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S. N. R., Houle, M. E., Schoenebeck, G., Song, D., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1801.02613.

Malkov, Y. A. and Yashunin, D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. volume 42, pp. 824–836, USA, April 2020. IEEE Computer Society. doi: 10.1109/TPAMI.2018.2889473. URL https://doi.org/10.1109/TPAMI.2018.2889473.

Robertson, S. and Zaragoza, H. The probabilistic relevance framework: Bm25 and beyond. In *Foundations and Trends in Information Retrieval*, volume 3, pp. 333–389. Now Publishers, Inc., 2009. URL https://dl.acm.org/doi/abs/10.1561/1500000019.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323. URL https://www.science.org/doi/abs/10.1126/science.290.5500.2323.

Simhadri, H. V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S. J., and Wang, J. Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 177–189. PMLR, 2022. URL https://proceedings.mlr.press/v176/simhadri22a/simhadri22a.pdf.

Subramanya, S. J., Devvrit, Kadekodi, R., Krishaswamy, R., and Simhadri, H. V. Diskann: fast accurate billion-point nearest neighbor search on a single node. In *Proceedings of the 33rd International*

Conference on Neural Information Processing Systems, 2019. URL https://dl.acm.org/doi/abs/10.5555/3454287.3455520.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319–2323, 2000. doi: 10.1126/science.290.5500.2319. URL https://www.science.org/doi/abs/10.1126/science.290.5500.2319.

Toussaint, G. T. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12 (4):261–268, 1980. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(80)90066-7. URL https://www.sciencedirect.com/science/article/pii/0031320380900667.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. 2023. URL https://arxiv.org/abs/2302.13971.

A. Detailed Theoretical Proofs

A.1. Derivation of Local Complexity Lower Bound (Lemma 4.1)

Restatement. *The expected number of distance evaluations scales exponentially with the local intrinsic dimensionality.*

Proof. Let the local dataset around a query q be modeled as a uniform distribution on a manifold of intrinsic dimension $d = \text{LID}(q)$. Consider a greedy routing step where the algorithm is currently at a node u with distance $r = \|u - q\|$. To make significant progress towards the nearest neighbor, the algorithm must find a neighbor v in the graph such that the distance to the query is reduced by at least a factor ϵ (where $0 < \epsilon < 1$), i.e., $\|v - q\| \leq (1 - \epsilon)r$.

We analyze the probability of a randomly distributed neighbor falling into this "improving region." In the tangent space at u, the directions towards neighbors can be approximated as being uniformly distributed on the unit hypersphere $\mathbb{S}^{d-1}$. The region of space satisfying the distance constraint corresponds to a spherical cap $\mathcal{C}$ on $\mathbb{S}^{d-1}$ defined by a semi-vertical angle θ.

The ratio of the surface area of a spherical cap $\mathcal{C}$ with angle θ to the total area of the hypersphere $\mathbb{S}^{d-1}$ is given by the regularized incomplete beta function, which for small θ and large d is asymptotically bounded by:

$$\frac{\text{Area}(\mathcal{C})}{\text{Area}(\mathbb{S}^{d-1})} \approx \frac{1}{\sqrt{d\pi}}(\sin\theta)^{d-1}. \tag{27}$$

Assuming the step size to the neighbor is small relative to r, the geometry implies that the required angle θ for an ϵ-improvement is fixed (specifically, $\cos\theta \approx 1 - \epsilon$). Let $\gamma = \sin\theta < 1$. The probability $P_{success}$ that a random candidate neighbor falls within this improving cap scales as:

$$P_{success} \propto \gamma^{d-1} = \exp\left((d-1)\ln\gamma\right) \approx \exp(-\lambda \cdot d), \tag{28}$$

where $\lambda = -\ln(\sin\theta) > 0$ is a constant determined by the geometric configuration and the required convergence rate ϵ.

Since the neighbors are checked sequentially (or in a batch), the number of distance evaluations N_{dist} required to find at least one successful candidate follows a geometric-like distribution with success probability $P_{success}$. The expected number of evaluations is:

$$\mathbb{E}[N_{dist}] = \frac{1}{P_{success}} \propto \exp(\lambda \cdot d). \tag{29}$$

Thus, the computational cost to maintain a greedy descent path grows exponentially with the local intrinsic dimensionality d. This theoretical lower bound validates our MCGI design choice, where the search beam width is adapted according to $L(q) \propto \exp(\widehat{LID}(q))$ to counteract the exponentially shrinking probability of finding a good routing path. □

B. Scalability on Billion-Scale Data

To further validate the engineering scalability and robustness of MCGI, we extended our evaluation to the billion-scale regime. We utilized the standard **BIGANN (SIFT1B)** benchmark (Jégou et al., 2011b), which consists of one billion 128-dimensional SIFT vectors. This experiment aims to assess whether the overhead of LID estimation and adaptive routing introduces any scalability bottlenecks when the index size scales from gigabytes to terabytes.

B.1. Experimental Setup

To validate scalability on the billion-scale SIFT1B benchmark, we migrated the evaluation to a high-capacity Storage Node on the Chameleon Cloud platform. The system is equipped with dual Intel Xeon E5-2650 v3 CPUs (40 threads @ 2.30GHz) and 64 GiB of RAM. The index was constructed and hosted on a 2TB Seagate Enterprise SAS HDD (Model ST2000NX0273) to accommodate the full dataset. We constructed a single index with a maximum degree $R = 64$ and a construction beam width $L_{build} = 100$. This configuration represents a rigorous *out-of-core* workload, as the dataset size (128 GB) significantly exceeds the system RAM (64 GB), forcing the algorithm to rely entirely on efficient disk I/O scheduling rather than OS page caching.

B.2. Results and Analysis

Table 2 presents the performance comparison on SIFT1B. MCGI achieves performance parity with DiskANN across the recall spectrum. Specifically, at 95% recall, both methods deliver comparable throughput. This result reinforces the findings from RQ3 (Generalizability) at a much larger scale.

Crucially, this experiment demonstrates two key engineering capabilities of MCGI. First, the Geometric Calibration phase scales linearly with dataset size N, proving that the pre-computation of LID statistics is computationally feasible even for billion-point datasets. Second, the adaptive routing mechanism incurs negligible CPU overhead. Even when navigating a graph with a billion nodes, the latency remains dominated by disk I/O rather than the arithmetic operations required to compute dynamic pruning parameters.

Table 2. Performance comparison on SIFT1B ($N = 10^9$). MCGI maintains parity with the state-of-the-art DiskANN, proving scalability.

Method	Recall@10	QPS
DiskANN (1B)	TBA%	TBA
MCGI (1B)	TBA%	TBA

C. More Implementation Details

Unlike standard standalone implementations of ANN indices, MCGI is designed as a modular, **agentic orchestration system** that automates the entire lifecycle of RAG serving—from data ingestion to adaptive indexing and query execution. Our implementation consists of a hybrid codebase spanning core C++ modifications and a comprehensive Python-based control plane.

C.1. Agentic Orchestration Framework

To ensure reproducibility and handle the complexity of billion-scale experiments, we developed a Python-based orchestration layer that manages the interaction between the storage backend and the indexing engine. As shown in the repository structure, the workflow is driven by specific agents:

- **Data Ingestion & Formatting (`convert_sift1b.py`, `gen_data.py`):** Custom pipelines were engineered to handle the conversion of legacy SIFT1B formats (compressed `.bvecs`, `.tar.gz`) into memory-mapped binary formats (`.bin`) optimized for direct disk access. This module includes integrity checks and automatic dimension validation to ensure data consistency before indexing.

- **LID Estimation Engine (`compute_lid.py`):** A standalone module responsible for pre-computing the Local Intrinsic Dimensionality for the raw dataset. This script implements the MLE-based estimator and generates the metadata required by the MCGI construction algorithm, serving as the "cognitive" input for the adaptive system.

- **Adaptive Index Construction (`run_mcgi_sigmoid.sh`):** This driver script automates the build process. It dynamically injects the Sigmoid-based pruning parameters ($\alpha(u)$) into the C++ builder based on the computed LID distribution, removing the need for manual parameter tuning and ensuring the graph topology adapts to local geometric complexity.

- **Automated Evaluation Pipeline (`full_scan.sh`, `scan_patch.sh`):** To rigorously capture the Recall-QPS trade-off, we implemented a batch execution system that automatically sweeps through varying beam widths (L_{search}) and thread counts. This ensures that the performance curves are generated from comprehensive, rather than cherry-picked, data points.

C.2. Kernel Modifications

We extended the foundational DiskANN (Vamana) C++ codebase to support variable-density topology. Key engineering modifications include:

- **Dynamic Pruning Logic:** We rewrote the graph neighbor selection kernel to accept per-node pruning thresholds $\alpha(u)$, replacing the static α parameter found in the original implementation. This allows the index to act conservatively in high-LID regions while remaining aggressive in low-LID areas.
- **Hybrid Environment Management:** The system is encapsulated in a Python virtual environment (`venv`) with automated dependency resolution for critical low-level libraries (`libaio`, `tcmalloc`, and `Intel MKL`). This ensures consistent performance across heterogeneous hardware nodes (e.g., facilitating the transition from Ice Lake to Haswell architectures).
- **Low-Level Optimizations:**
  1. **SIMD Instructions:** We utilize **AVX2** intrinsics for high-performance distance computations (L_2 Euclidean distance), maximizing throughput on the Intel Haswell architecture.
  2. **Direct I/O:** To bypass the OS page cache during random reads in the search phase, we employ `O_DIRECT` flags, ensuring that latency measurements reflect true SSD/HDD performance without OS interference.
  3. **Prefetching:** We explicitly utilize software prefetching (`_mm_prefetch`) to load graph adjacency lists into the cache hierarchy ahead of traversal, masking memory access latency during greedy beam search.

D. Detailed Hyperparameter Configuration

We provide a comprehensive listing of the hyperparameters used across the ingestion, construction, and query processing phases. The configuration is divided into graph topology settings (Table 3) and adaptive runtime parameters (Table 4).

D.1. Construction and Compression Settings

Table 3 details the parameters governing the offline index construction. Crucially, for the billion-scale SIFT1B dataset, we utilized **Product Quantization (PQ)** to compress vectors for the in-memory navigation graph, allowing the system to adhere to the 64GB RAM constraint of the storage node.

Table 3. Graph Construction and Compression Parameters. Note that for SIFT1B, a strict RAM budget was enforced, necessitating aggressive PQ compression.

Dataset	R	L_{build}	α Range	m_{PQ} (Bytes)	k_{LID}	Build RAM Limit
SIFT1M	64	100	$[1.0, 1.5]$	N/A (Float)	32	Unconstrained
GloVe-100	64	100	$[1.0, 1.5]$	N/A (Float)	50	Unconstrained
GIST1M	96	150	$[1.0, 1.5]$	N/A (Float)	50	Unconstrained
SIFT1B	64	100	$[1.0, 1.5]$	16	100	64 GB

D.2. Adaptive Mapping and Search Configuration

The MCGI agent dynamically maps geometric complexity to topology constraints. Table 4 lists the statistical parameters derived from the dataset's LID distribution used in the Sigmoid mapping function $\Phi(\cdot)$, alongside the runtime search sweep parameters.

Table 4. Adaptive Mapping Statistics and Runtime Search Parameters. The Sigmoid center (μ_{LID}) and scale (σ_{LID}) are derived from the pre-computed LID distribution.

Dataset	μ_{LID} (Mean)	σ_{LID} (Std)	Search Beam (L_{search})	Threads
SIFT1M	14.2	3.1	$10 \rightarrow 100$	1
GloVe-100	18.5	4.2	$10 \rightarrow 120$	1
GIST1M	22.1	5.8	$20 \rightarrow 200$	1
SIFT1B	16.8	3.5	$10 \rightarrow 400$	40